

Åke Björck

Linköping University

Royal Institute of Technology

**Numerical Methods
in
Scientific Computing**

Volume II

Working copy, August 28, 2009

siam

©This material is the property of the authors and is for the sole and exclusive use of the students enrolled in specific courses. It is not to be sold, reproduced, or generally distributed.

Contents

Preface	xiii
7 Direct Methods for Linear System	1
7.1 Linear Algebra and Matrix Computations	1
7.1.1 Matrix Algebra	2
7.1.2 Submatrices and Block Matrices	8
7.1.3 Permutations and Determinants	14
7.1.4 Vectors and Matrix Norms	19
7.1.5 The Singular Value Decomposition	25
7.1.6 Numerical Rank	28
7.1.7 Matrix Multiplication	30
7.1.8 Floating-Point Arithmetic	32
7.1.9 Complex Arithmetic in Matrix Computations	35
Review Questions	38
Problems	39
7.2 Gaussian Elimination	42
7.2.1 Solving Triangular Systems	42
7.2.2 Gaussian Elimination	44
7.2.3 LU Factorization	49
7.2.4 Pivoting and Stability	55
7.2.5 Computational Variants of LU Factorization	61
7.2.6 Computing the Inverse Matrix	65
Review Questions	68
Problems	69
7.3 Symmetric Matrices	69
7.3.1 Symmetric Positive Definite Matrices	69
7.3.2 Cholesky Factorization	75
7.3.3 Inertia of Symmetric Matrices	79
7.3.4 Symmetric Indefinite Matrices	81
Review Questions	85
Problems	85
7.4 Banded Linear Systems	86
7.4.1 Band Matrices	86
7.4.2 Multiplication of Band Matrices	88

7.4.3	LU Factorization of Band Matrices	89
7.4.4	Tridiagonal Linear Systems	94
7.4.5	Inverses of Band Matrices	98
7.4.6	Envelope Methods	99
Review Questions		100
Problems		101
7.5	Perturbation and Error Analysis	103
7.5.1	Perturbation Analysis for Linear Systems	103
7.5.2	Backward Error Bounds	111
7.5.3	Estimating Condition Numbers	114
7.5.4	Rounding Error Analysis of Gaussian Elimination .	117
7.5.5	Scaling of Linear Systems	122
7.5.6	Iterative Refinement of Solutions	125
7.5.7	Interval Matrix Computations	128
Review Questions		132
Problems		132
7.6	Block Algorithms	133
7.6.1	BLAS and Linear Algebra Software	133
7.6.2	Block and Partitioned Algorithms	136
7.6.3	A Recursive Fast Matrix Multiply	142
7.6.4	Recursive Cholesky and LU Factorizations . . .	144
7.6.5	Kronecker Structure and Linear Systems	147
Review Questions		149
Problems		149
7.7	Sparse Linear Systems	150
7.7.1	Introduction	150
7.7.2	Storage Schemes for Sparse Matrices	153
7.7.3	Graphs and Sparse Matrices.	155
7.7.4	The Elimination Tree	158
7.7.5	Orderings Algorithms for Cholesky Factorization .	160
7.7.6	LU Factorization of Sparse Unsymmetric Matrices .	168
7.7.7	Nonzero Diagonal and Block Triangular Form . .	171
Review Questions		172
Problems		173
7.8	Structured Linear Equations	174
7.8.1	Toepplitz and Hankel Matrices	174
7.8.2	Cauchy-Like Matrices	177
7.8.3	Vandermonde systems	180
7.8.4	Semiseparable Matrices	181
Review Questions		183
Problems		183
8	Linear Least Squares Problems	189
8.1	Preliminaries	189
8.1.1	The Least Squares Principle	189
8.1.2	The Gauss-Markov Model	193

8.1.3	Orthogonal and Oblique Projections	197
8.1.4	Generalized Inverses and the SVD	198
8.1.5	Matrix Approximation and the SVD	204
8.1.6	Principal Angles and Distance Between Subspaces .	207
8.1.7	The CS Decomposition	209
Review Questions		211
Problems		212
8.2	The Method of Normal Equations	213
8.2.1	Forming and Solving the Normal Equations	213
8.2.2	Computing the Covariance Matrix.	216
8.2.3	Perturbation Bounds for Least Squares Problems .	219
8.2.4	Stability and Accuracy with Normal Equations . .	222
8.2.5	Backward Error Analysis	225
8.2.6	The Peters–Wilkinson method	227
Review Questions		229
Problems		229
8.3	Orthogonal Factorizations	232
8.3.1	Elementary Orthogonal Matrices	232
8.3.2	Householder QR Factorization	239
8.3.3	Least Squares Problems by QR Factorization . .	246
8.3.4	Gram–Schmidt QR Factorization	249
8.3.5	Loss of Orthogonality in GS and MGS	253
8.3.6	MGS as a Householder Method	256
8.3.7	Error Estimation and Iterative Refinement . . .	261
Review Questions		264
Problems		265
8.4	Rank Deficient Problems	267
8.4.1	Rank Revealing QR Factorizations	267
8.4.2	Complete QR Factorizations	269
8.4.3	Column Subset Selection Problem	271
8.4.4	Modified Least Squares Problems	274
8.4.5	Golub–Kahan Bidiagonalization	279
8.4.6	Regularization by TSVD and Partial Least Squares	281
Review Questions		288
Problems		288
8.5	Structured and Sparse Least Squares Problems . .	290
8.5.1	Banded Least Squares Problems	290
8.5.2	Blocked Form of QR Factorization	293
8.5.3	Block Angular Least Squares Problems	297
8.5.4	Block Triangular Form	300
8.5.5	Sparse Least Squares Problems	303
8.5.6	Multifrontal QR decomposition.	307
8.5.7	Kronecker Product Problems	311
8.5.8	Tensor Computations	313
Review Questions		314
Problems		314

8.6	Some Generalized Least Squares Problems	314
8.6.1	Least Squares for General Linear Models	314
8.6.2	Indefinite Least Squares	319
8.6.3	Linear Equality Constraints	321
8.6.4	Quadratic Constraints and Tikhonov Regularization	323
8.6.5	Linear Orthogonal Regression	329
8.6.6	The Orthogonal Procrustes Problem	332
	Review Questions	334
	Problems	334
8.7	The Total Least Squares Problem	335
8.7.1	Total Least Squares and the SVD	335
8.7.2	Conditioning of the TLS Problem	338
8.7.3	Some Generalized TLS Problems	339
8.7.4	Bidiagonalization and TLS Problems.	342
8.7.5	Solving Overdetermined Systems in the l_p Sense.	344
	Review Questions	347
	Problems and Computer Exercises	347
9	Nonlinear and Constrained Problems	353
9.1	Systems of Nonlinear Equations	353
9.1.1	Calculus in Vector Spaces	354
9.1.2	The Contraction Mapping Theorem	358
9.1.3	Newton's Method and Its Variants	361
9.1.4	Derivative Free Methods	368
9.1.5	Parametrized Nonlinear Systems	372
	Review Questions	374
	Problems	375
9.2	Nonlinear Least Squares Problems	377
9.2.1	Methods for Unconstrained Optimization	378
9.2.2	Nonlinear Least Squares Problems	382
9.2.3	Gauss–Newton and Newton Methods	384
9.2.4	Separable Problems	391
9.2.5	Nonlinear Orthogonal Regression	394
9.2.6	Fitting of Circles and Ellipses.	397
	Review Questions	402
	Problems	403
9.3	Linear Programming	405
9.3.1	Optimality for Linear Inequality Constraints	405
9.3.2	Standard Form for LP	409
9.3.3	The Simplex Method	412
9.3.4	Duality	418
9.3.5	Barrier Functions and Interior Point Methods	420
	Review Questions	421
	Problems	421
9.4	Least Squares Problems with Inequality Constraints	422
9.4.1	Classification of Problems.	422

9.4.2	Basic Transformations of Problem LSI	425
9.4.3	Active Set Algorithms for Problem LSI	426
9.4.4	An Active Set Algorithm for BLS	430
Review Questions		431
Problems		431
10	Matrix Eigenvalue Problems	433
10.1	Basic Properties	433
10.1.1	Introduction	433
10.1.2	Basic Theory	434
10.1.3	The Jordan Canonical Form	440
10.1.4	The Schur Decomposition	443
10.1.5	Sylvester's Equation and Spectral Decomposition .	449
10.1.6	Nonnegative Matrices	451
Review Questions		453
Problems		453
10.2	Perturbation Theory and Eigenvalue Bounds	456
10.2.1	Geršgorin's Theorem	456
10.2.2	General Perturbation Theorems	459
10.2.3	Hermitian Matrices	463
10.2.4	Invariant Subspaces and Newton-Based Methods .	469
Review Questions		475
Problems		476
10.3	The Power Method and Its Generalizations	477
10.3.1	The Simple Power Method	477
10.3.2	Deflation of Eigenproblems	480
10.3.3	Inverse Iteration	482
10.3.4	Rayleigh Quotient Iteration	485
10.3.5	Subspace Iteration	487
Review Questions		490
Problems		490
10.4	The QR Algorithm	491
10.4.1	The Basic QR Algorithm	491
10.4.2	Reduction to Hessenberg Form	495
10.4.3	The Hessenberg QR Algorithm	498
10.4.4	Balancing an Unsymmetric Matrix	503
Review Questions		505
Problems		505
10.5	The Hermitian QR Algorithm	506
10.5.1	Reduction to Symmetric Tridiagonal Form	506
10.5.2	The Real Symmetric QR Algorithm	508
10.5.3	The QR–SVD Algorithm	512
10.5.4	Skew-Symmetric and Unitary Matrices	521
Review Questions		523
Problems		524
10.6	Some Alternative Algorithms	526

10.6.1	A Divide and Conquer Algorithm	526
10.6.2	Spectrum Slicing	531
10.6.3	Jacobi Methods	535
10.6.4	Jacobi Methods for Computing the SVD	538
Review Questions		542
Problems		542
10.7	Generalized Eigenvalue Problems	544
10.7.1	Canonical Forms	544
10.7.2	Reduction to Standard Form	547
10.7.3	Methods for Generalized Eigenvalue Problems	549
10.7.4	Quadratic and Structured Eigenvalue Problems	553
10.7.5	Generalized Singular Value Decomposition	555
10.7.6	Pseudospectra of Nonnormal Matrices	556
Review Questions		559
Problems		559
10.8	Matrix Functions	560
10.8.1	Convergence of Matrix Power Series	560
10.8.2	Analytic Matrix Functions	562
10.8.3	Matrix Exponential and Logarithm	566
10.8.4	Matrix Square Root and the Polar Decomposition	571
10.8.5	The Matrix Sign Function	576
10.8.6	Estimating Matrix Functionals	578
10.8.7	Finite Markov Chains	579
Review Questions		583
Problems and Computer Exercises		584
11	Iterative Methods	591
11.1	Classical Iterative Methods	591
11.1.1	Introduction	591
11.1.2	A Model Problem	592
11.1.3	Stationary Iterative Methods	594
11.1.4	Convergence of Stationary Iterative Methods	597
11.1.5	Successive Overrelaxation Methods	602
11.1.6	Chebyshev Acceleration	610
11.1.7	Effects of Nonnormality and Finite Precision	613
Review Questions		617
Problems and Computer Exercises		618
11.2	Projection Methods	620
11.2.1	Principles of Projection Methods	620
11.2.2	The One-Dimensional Case	622
11.2.3	The Conjugate Gradient Method	626
11.2.4	Convergence Properties of CG	631
11.2.5	The Lanczos Process	634
11.2.6	Symmetric Indefinite Systems	638
Review Questions		640
Problems and Computer Exercises		640

11.3	Least Squares Problems	641
11.3.1	Introduction	641
11.3.2	Some Classical Methods	643
11.3.3	Krylov Subspace Methods	648
11.3.4	Lanczos Bidiagonalization and LSQR	651
11.3.5	Regularization by Iterative Methods	655
	Review Questions	659
	Problems and Computer Exercises	659
11.4	Unsymmetric Problems	660
11.4.1	Arnoldi's Method and GMRES	660
11.4.2	The Bi-Lanczos and Bi-CG Methods	665
11.4.3	Transpose-Free Methods	670
	Review Questions	672
	Problems and Computer Exercises	673
11.5	Preconditioned Iterative Methods	673
11.5.1	Preconditioned Krylov Subspace Methods	675
11.5.2	Preconditioners from Matrix Splittings	679
11.5.3	Incomplete Factorization Methods	681
11.5.4	Sparse Approximate Inverses	684
11.5.5	Block Tridiagonal Matrices	686
11.5.6	Constant Coefficient Approximation	688
11.5.7	Preconditioners for Toeplitz Systems	690
	Review Questions	693
	Problems and Computer Exercises	694
11.6	Large-Scale Eigenvalue Problems	695
11.6.1	The Rayleigh–Ritz Procedure	696
11.6.2	The Arnoldi Algorithm	698
11.6.3	The Simple Lanczos Method	703
11.6.4	Lanczos Method in Finite Precision	705
11.6.5	Convergence Analysis	710
11.6.6	Simultaneous Iteration for Hermitian Matrices	713
11.6.7	Spectral Transformation	714
	Review Questions	716
	Problems	717

Bibliography	721
--------------	-----

Index	757
-------	-----

*Dedicated to the memory of
Germund Dahlquist and Gene H. Golub*

List of Figures

7.1.1	Singular values of a numerically singular matrix.	29
7.2.1	Illustration of rook pivoting in a 5×5 matrix with positive integer entries as shown. The (2, 4) element 9 is chosen as pivot.	58
7.2.2	Computations in the k th step of Doolittle's method.	62
7.2.3	Computations in the k th step of the bordering method.	64
7.2.4	Computations in the k th step of the sweep methods. Left: The column sweep method. Right: The row sweep method.	65
7.3.1	Computations in the k th step of the Cholesky methods. Left: The row sweep method. Right: The column sweep method.	76
7.4.1	Structure of A (left) and $L + L^T$ (right) for the Laplace equation, $n = 20$ with rowwise ordering of the unknowns.	93
7.5.1	The solution set (solid line) and its enclosing hull (dashed line) for the linear system in Example 7.6.5.	130
7.6.1	Binary tree with nodes in preorder.	143
7.7.1	Nonzero pattern of a matrix W and its LU factors.	151
7.7.2	Nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor L^T ; $nnz(A_L) = 4015$; $nnz(L) = 30,141$	161
7.7.3	Nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor after reverse Cuthill–McKee reordering; $nnz(A) = 4015$; $nnz(L) = 23,596$	163
7.7.4	Nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor after minimum degree reordering. $nnz(A) = 4015$; $nnz(L) = 8,911$. 164	164
7.7.5	A 3×3 element in a Regular grid.	166
7.7.6	Nonzero pattern of the matrix W west0479 and its LU factors after reordering by increasing column count; $nnz(W) = 1887$; $nnz(L + U) = 6604$	169
7.7.7	Nonzero pattern of the matrix west0479 and its LU factors after reordering by and its LU factors after column minimum degree ordering; $nnz(W) = 1887$; $nnz(L + U) = 5904$	170
8.1.1	Geometric characterization of the least squares solution.	190
8.3.1	A Householder reflection of the vector a	233
8.4.1	Relative error $\ x_k - x\ _2/\ x\ _2$ and residual $\ Ax_k - b\ _2/\ b\ _2$ for TSVD solutions truncated after k steps.	282

8.4.2	Relative error $\ x_k - x\ _2/\ x\ _2$ and residual $\ Ax_k - b\ _2/\ b\ _2$ for PLS solutions after k steps.	287
8.5.1	The QR factorization of a banded rectangular matrix A	292
8.5.2	One and two levels of dissection of a region.	300
8.5.3	The coarse block triangular decomposition of A	301
8.5.4	Circled elements \otimes in R_{k-1} are involved in the elimination of a_k^T ; fill elements are denoted by \oplus	306
8.5.5	Nonzero pattern of A and its R factor after reordering using MATLAB's colperm.	307
8.5.6	Nonzero pattern of A and its R factor after reordering using MATLAB's colamd.	308
8.5.7	A matrix A corresponding to a 3×3 mesh.	308
8.5.8	The graph $G(A^T A)$ and a nested dissection ordering.	309
8.5.9	The reduced matrix after the first elimination stage.	309
9.1.1	A rotating double pendulum.	376
9.2.1	Geometry of the data fitting problem for $m = 2$, $n = 1$	386
9.2.2	Orthogonal distance regression.	395
9.2.3	Ellipse fits for triangle and shifted triangle data: (a) SVD constraint; (b) Linear constraint $\lambda_1 + \lambda_2 = 1$; (c) Bookstein constraint $\lambda_1^2 + \lambda_2^2 = 1$	399
9.3.1	Geometric illustration of a linear programming problem.	406
9.3.2	Feasible points in a degenerate case.	415
10.8.1	$\ e^{tA}\ $ as a function of t for the matrix in Example 10.8.3.	567
11.1.1	$f_\omega(\lambda)$ and $g(\lambda, \mu)$ as functions of λ ($\mu = 0.99$, $\omega = \omega_b = 1.7527$).	606
11.1.2	The spectral radius $\rho(B_\omega)$ as a function of ω ($\rho = 0.99$, $\omega_b = 1.7527$).	607
11.1.3	$\ x^{(n)}\ _2$, where $x^{(k+1)} = Bx^{(k)}$, and $x^{(0)} = (1, 1, \dots, 1)^T$	615
11.2.1	Convergence of the error norm for a 32 by 32 Laplace problem.	632
11.3.1	Structure of A (left) and $A^T A$ (right) for a model problem.	642
11.3.2	Relative error $\ x_k - x\ _2/\ x\ _2$ and residual $\ Ax_k - b\ _2/\ b\ _2$ for LSQR solutions after k steps.	657
11.6.1	Eigenvalues (+) and Ritz values (x) for a 100×100 matrix with normally distributed entries after 30 steps of Arnoldi.	700

List of Tables

7.1.1	Numbers γ_{pq} such that $\ A\ _p \leq \gamma_{pq}\ A\ _q$, where $A \in \mathbf{C}^{m \times n}$ and $\text{rank}(A) = r$	23
7.5.1	Condition numbers of Hilbert matrices of order ≤ 12	106
8.3.1	Loss of orthogonality and CGS and MGS.	254
10.5.1	Approximate flop counts for the QR–SVD algorithm.	518
11.1.1	Number of iterations needed to reduce the initial error by a factor of 10^{-3} for the model problem, as a function of $n = 1/h$	607

Preface

This is the second volume of a textbook intended to replace the book by Dahlquist and Björck, *Numerical Methods*, published in 1974 by Prentice–Hall. As mentioned in the preface to the first volume my mentor, friend, and coauthor Germund Dahlquist died in 2005, before even the first volume was published. It has been a great privilege for me trying to finish the first two volumes. which I hope reflects some of Germund's enthusiasm for the subject.

The development of methods of matrix computations reflects the tremendous increase in the use of sophisticated mathematical models in the last decades. Advanced mathematical models and methods are today used more and more in science and engineering. Today, experiment and theory, are supplemented in many areas by computations that are an equally important component. More complex and less simplified problems are now treated through massive amounts of numerical calculations.

Matrix computations have become of increasing importance during the last decades and are now fundamental and ubiquitous in modern scientific computing. Although there is a link between matrix computations and linear algebra as taught in department of mathematics there are also several important differences. In mathematics one tends to think of matrices as coordinate representations of linear operators with respect to some basis. However, matrices can also represent other things, such as bilinear forms, graph structures, images, DNA measurements, and so on. Many of these make little sense when viewed as an operator. In numerical linear algebra one works in the field of real or complex numbers. Concepts such as ill-conditioning, norms, and orthogonality, which do not extend to arbitrary fields, are central to matrix computations. This allows the use of powerful tools like QR factorization and singular value decomposition.

This volume is suitable for use in a course on matrix computations at graduate levels in applied mathematics. Great care has been taken in giving additional references that gives an up-to-date picture of the subject, in the hope that the volume can be used also as a reference in applied sciences. A necessary prerequisite is an introductory course in linear algebra.

A key feature of this book is that it aims to cover both direct methods (full and sparse) as well as iterative methods for linear systems and least squares problems as well as eigenvalue problems. Except for a few exceptions, other books available treat only linear systems or eigenvalue problems; or only direct or iterative methods. Treating both linear systems and eigenvalue problems in the same text has many

advantages. Much of the theory and methods used for these two problems are closely intertwined—it suffices to mention Krylov subspace methods. Hence a lot of duplication is avoided.

As in the first volume an important feature of this book is a collection of review questions, problems and computer exercises included after each section. Many of these draws from the authors' experience in teaching. It is highly recommended that a modern interactive system such as MATLAB is available to the reader for working out these assignments.

A comprehensive and up-to-date bibliography is an important part of the book. A Notes and References section containing historical comments and additional references concludes each chapter. Both review papers and recent research papers of importance are included. As in the first volume many short biographical notes on mathematicians who have made significant contributions to matrix computations are given. These 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). Many of these biographies are fascinating to read in full.

I am very grateful for the encouragement received from the Dahlquist family, who graciously allowed me to access computer files from Germund Dahlquist's personal computer.

Many people read early drafts at various stages of the evolution of this book and contributed many corrections and constructive comments. I am particularly grateful to Nick Higham and anonymous referees whose suggestions led to several major improvements. The book was typeset in L^AT_EX the references were prepared in BIB^LT_EX and the index with MAKEINDEX. These are all wonderful tools and my thanks goes to Donald Knuth for his gift to Mathematics.

It is a pleasure to thank Elizabeth Greenspan, Sarah Murphy, and other staff at SIAM for their cheerful and professional support during all phases of the acquisition and production of the book.

Åke Björck
Linköping, September 2009

List of Conventions

Besides the generally accepted mathematical abbreviations and notations (see e.g., James and James, *Mathematics Dictionary* [1985, pp. 467–471]), the following notations are used in the book:

MATLAB® has been used for this book in testing algorithms. We also use its notations for array operations and the convenient colon notation.

$.*$	$A . * B$ element-by-element product $A(i,j)B(i,j)$
$./$	$A ./ B$ element-by-element division $A(i,j)/B(i,j)$
$i : k$	same as $i, i + 1, \dots, k$ and empty if $i > k$
$i : j : k$	same as $i, i + j, i + 2j, \dots, k$
$A(:,k), A(i,:)$	the k th column, i th row of A , respectively
$A(i:k)$	same as $A(i), A(i+1), \dots, A(k)$
$\text{fl}(x+y)$	floating-point operations
$\{x_i\}_{i=0}^n$	denotes the set $\{x_0, x_1, \dots, x_n\}$
$[a, b]$	closed interval ($a \leq x \leq b$)
(a, b)	open interval ($a < x < b$)
$\text{sign}(x)$	+1 if $x \geq 0$, else -1
$f(n) = O(g(n)), n \rightarrow \infty$	$ f(n)/g() $ is bounded as $n \rightarrow \infty$
$f(n) = o(g(n)), n \rightarrow \infty$	$\lim_{n \rightarrow \infty} f(n)/g(n) = 0$
$k \leq i, j \leq n$	means $k \leq i \leq n$ and $k \leq j \leq n$
\mathcal{P}_n	the set of polynomials of degree <i>less than</i> n
(f, g)	scalar product of functions f and g
$\ \cdot\ _p$	p -norm in a linear vector or function space;

Matrices and vectors are generally denoted by Roman letters A and b . A^T and b^T denote the transpose of the matrix A and the vector b , respectively. (A, B) means a partitioned matrix;

Chapter 7

Direct Methods for Linear System

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.

—A. M. Turing, Rounding-off errors in matrix processes. Quart. J. Mech. Appl. Math., 1948.

7.1 Linear Algebra and Matrix Computations

The numerical solution of a system of linear equations enters at some stage in almost all applications. Even though the mathematical theory is simple and algorithms have been known for centuries, decisive progress has been made in the last few decades. It is important to note that some methods which are perfectly acceptable for theoretical use, may be useless for solving systems with several thousands or even million of unknown. Algorithms for solving linear systems are the most widely used in scientific computing and it is of great importance that they are both efficient and reliable. They have also to be adopted continuously as computer architectures change. Critical details in the algorithms can influence the efficiency and accuracy in a way the beginner can hardly expect. It is strongly advisable to use efficient and well-tested software available.

Two quite different classes of methods for solving systems of linear equations $Ax = b$ are of interest: **direct** and **iterative** methods. In direct methods, typified by Gaussian elimination, the matrix A is transformed by a sequence of elementary transformations into a system of simpler form, e.g., triangular or diagonal form, which can be solved in an elementary way. Disregarding rounding errors, direct methods give the exact solution after a finite number of arithmetic operations. Direct methods are inadequate To handle the very large linear systems that appear in many applications. A characteristic of iterative methods are that the matrix A is never transformed, but only used in matrix-vector operations such as Ax . Typically, less than 100 such operations are needed to give a sufficiently good approximate

solution.

Since the 1950s and early 1960s the so-called **decompositional approach** to matrix computation has come into favor. A prime example is Gaussian elimination, which is interpreted as a factorization of a matrix A into the product of a lower triangular matrix L and an upper triangular matrix U , the LU factorization. This approach extends to many other factorizations such as Cholesky and QR, the singular value and eigenvalue decompositions. The decompositional approach has also influenced the development of iterative methods. It has been named as one of the ten algorithms with most influence on science and engineering in the 20th century; see Stewart [550].

An important aspect of matrix computation is to take advantage of any special structure of the matrix that can speed up the algorithms. A matrix A for which only a small fraction of the elements are nonzero is called **sparse**. A simple case is when A has a banded structure, where the nonzero elements are close to the main diagonal. This is simple to treat, but many important applications areas give more random sparsity patterns. In the last decades there has been great progress in handling such problems, which without exploiting sparsity would be intractable!

Some classes of matrices are dense but **structured**. One example is Vandermonde systems, which are related to polynomial interpolation. Other important examples of structured matrices are Toeplitz, Hankel, and Cauchy matrices. In all these instances the n^2 elements in the matrix are derived from only $O(n)$ quantities. Solution algorithms exists which only use $O(n^2)$ or less arithmetic operations.

7.1.1 Matrix Algebra

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

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

The term “matrix” was first coined in 1848 by J.J. Sylvester.¹ Much of the terminology in linear algebra is due to Sylvester, e.g., “canonical form”, “minor”, and “nullity”. Cayley² made the fundamental discovery that such an array of numbers can be conceived as one single algebraic quantity $A = (a_{ij})$, with which certain algebraic operations can be performed. His paper [91] published in 1858 is the first paper on matrix algebra.

¹James Joseph Sylvester English mathematician (1814–1893) studied at St. John’s College, Cambridge. Because of his Jewish faith Sylvester had trouble in finding an adequate research position in England. His most productive period was in 1877–1884, when he held a chair at Johns Hopkins university, Baltimore, in USA.

²Arthur Cayley (1821–1895) English mathematician studied at Trinity College, Cambridge. He worked as a lawyer before being appointed Sadlerian Professor of Pure Mathematics at Cambridge in 1863. Besides developing the algebra of matrices his most important work was in geometry and group theory. His work on matrices served as a foundation for quantum mechanics developed by Heisenberg in 1925.

We write $A \in \mathbf{R}^{m \times n}$, where $\mathbf{R}^{m \times n}$ denotes the set of all real $m \times n$ matrices. If $m = n$, then the matrix A is said to be square and of order n . An empty matrix is a matrix with at least one dimension equals 0, Empty matrices are convenient to use as place holders.

A **column vector** is a matrix consisting of just one column and we write $x \in \mathbf{R}^m$ instead of $x \in \mathbf{R}^{m \times 1}$. Similarly, a **row vector** is a matrix consisting of just one row.

In this book we will follow a notational convention introduced by Householder³ and use uppercase letters (e.g., A, B) to denote matrices. The corresponding lowercase letters with subscripts ij then refer to the (i, j) component of the matrix (e.g., a_{ij}, b_{ij}). Greek letters α, β, \dots are used to denote scalars. Column vectors are usually denoted by lower case letters (e.g., x, y).

Basic Operations

The two fundamental 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 commutative law of multiplication.

The addition of two matrices A and B in $\mathbf{R}^{m \times n}$ is a simple operation. The sum, only defined if A and B have the same dimension, equals

$$C = A + B, \quad c_{ij} = a_{ij} + b_{ij}. \quad (7.1.1)$$

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

$$B = \alpha A, \quad b_{ij} = \alpha a_{ij}. \quad (7.1.2)$$

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

$$y_i = \sum_{j=1}^n a_{ij} x_j, \quad i = 1 : m,$$

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 column vector x . This definitions means that the linear system

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, \quad (7.1.3)$$

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

The general case now follows from the requirement that if $z = By$ and $y = Ax$, then substituting we should obtain $z = BAx = Cx$, where $C = BA$. Clearly the

³Alston S. Householder (1904–1993) American mathematician at Oak Ridge National Laboratory and University of Tennessee. He pioneered the use of matrix factorization and orthogonal transformations in numerical linear algebra.

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

$$C = AB \in \mathbf{R}^{m \times p}, \quad c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}. \quad (7.1.4)$$

Matrix multiplication is associative and distributive,

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

but not *not commutative*. The product BA is not even defined unless $p = m$. Then the matrices $AB \in \mathbf{R}^{m \times m}$ and $BA \in \mathbf{R}^{n \times n}$ are both square, but if $m \neq n$ of different orders. In general, $AB \neq BA$ even when $m = n$. If $AB = BA$ the matrices are said to **commute**.

The **transpose** A^T of a matrix $A = (a_{ij})$ is the matrix whose rows are the columns of A , i.e., if $C = A^T$, then $c_{ij} = a_{ji}$. Row vectors are obtained by transposing column vectors (e.g., x^T, y^T). If A and B are matrices of the same dimension then clearly $(A + B)^T = B^T + A^T$. For the transpose of a product we have the following result, the proof of which is left as an exercise for the reader.

Lemma 7.1.1.

If A and B are matrices such that the product AB is defined, then it holds that

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

that is, the product of the transposed matrices in reverse order.

A square matrix $A \in \mathbf{R}^{n \times n}$ is called **symmetric** if $A^T = A$ and **skew-symmetric** if $A^T = -A$. An arbitrary matrix can always be represented uniquely in the form $A = S + K$, where S is symmetric and K is skew-symmetric and

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

It is called **persymmetric** if it is symmetric about its antidiagonal, i.e.

$$a_{ij} = a_{n-j+1, n-i+1}, \quad 1 \leq i, j \leq n.$$

The matrix $A \in \mathbf{R}^{n \times n}$ is called **normal** if

$$AA^T = A^T A.$$

The **standard inner product** of two vectors x and y in \mathbf{R}^n is given by

$$x^T y = \sum_{i=1}^n x_i y_i = y^T x, \quad (7.1.5)$$

and the **Euclidian length** of the vector x is

$$\|x\|_2 = (x^T x)^{1/2} = \left(\sum_{i=1}^n |x_i|^2 \right)^{1/2}. \quad (7.1.6)$$

The **outer product** of $x \in \mathbf{R}^m$ and $y \in \mathbf{R}^n$ is the matrix

$$xy^T = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} (y_1 \ y_2 \ \cdots \ y_n) = \begin{pmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{pmatrix} \in \mathbf{R}^{m \times n}. \quad (7.1.7)$$

The absolute value of a matrix A and vector b is interpreted elementwise and defined by

$$|A|_{ij} = (|a_{ij}|), \quad |b|_i = (|b_i|).$$

The partial ordering “ \leq ” for matrices A, B and vectors x, y is to be interpreted componentwise⁴

$$A \leq B \iff a_{ij} \leq b_{ij}, \quad x \leq y \iff x_i \leq y_i.$$

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 holds for matrix-vector multiplication.

It is useful to define some other **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

$$C = A .* B, \quad c_{ij} = a_{ij} * b_{ij} \quad (7.1.8)$$

is the **Hadamard product**. Similarly, if $B > 0$ the matrix $C = A ./ B$ is the matrix with elements $c_{ij} = a_{ij} / b_{ij}$. (Note that for $+, -$ array operations coincides with matrix operations so no distinction is necessary.)

Any matrix D for which $d_{ij} = 0$ if $i \neq j$, is called a **diagonal matrix**. Hence, a square diagonal matrix has the form

$$D = \begin{pmatrix} d_{11} & 0 & \dots & 0 \\ 0 & d_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_{nn} \end{pmatrix}.$$

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

The **unit matrix** of order n is the matrix

$$I_n = \text{diag}(1, 1, \dots, 1) = (e_1, e_2, \dots, e_n), \quad (7.1.9)$$

⁴Note that when A and B are square symmetric matrices $A \leq B$ in other contexts can mean that the matrix $B - A$ is positive semidefinite.

The elements of the unit matrix are δ_{ij} , where δ_{ij} is the **Kronecker symbol** defined by

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \quad (7.1.10)$$

For all square matrices of order n , it holds that

$$A I_n = I_n A = A.$$

Usually the dimension if the size of the unit matrix will be clear from the context. Then we delete the subscript and just write I . The vector e_k denotes the k -th column of the unit matrix. 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

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

with all elements equal to one. These vectors will also be used without special reference to their dimension, if it is clear from the context.

Vector Spaces

We will be concerned with the vector spaces \mathbf{R}^n and \mathbf{C}^n , that is the set of real or complex n -tuples with $1 \leq n < \infty$. Let v_1, v_2, \dots, v_k be vectors and $\alpha_1, \alpha_2, \dots, \alpha_k$ scalars. The vectors are said to be **linearly independent** if none of them is a linear combination of the others, that is

$$\sum_{i=1}^k \alpha_i v_i = 0 \Rightarrow \alpha_i = 0, \quad i = 1 : k.$$

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 all vectors $v \in \mathcal{V}$ 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 every system of linearly independent vectors of \mathcal{V} has at most k elements and any other basis of \mathcal{V} has the same number k of elements. The number k is called the **dimension** of \mathcal{V} and denoted by $\dim(\mathcal{V})$.

The linear space of column vectors $x = (x_1, x_2, \dots, x_n)^T$, where $x_i \in \mathbf{R}$ is denoted \mathbf{R}^n ; if $x_i \in \mathbf{C}$, then it is denoted \mathbf{C}^n . The dimension of this space is n , and the unit vectors e_1, e_2, \dots, e_n , where

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

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 \mathbf{C}^n to \mathbf{C}^m is a function f such that

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

for all $\alpha, \beta \in \mathbf{K}$ and $u, v \in \mathbf{C}^n$. Let x and y be the column vectors representing the vectors v and $f(v)$, respectively, using the standard basis of the two spaces. Then, there is a unique matrix $A \in \mathbf{C}^{m \times n}$ 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 $A \in \mathbf{R}^{m \times n}$, 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**.

A square matrix $A \in \mathbf{R}^{n \times n}$ is **nonsingular** if and only if $\text{rank}(A) = n$. Then there exists an **inverse matrix** denoted by A^{-1} with the property that

$$A^{-1}A = AA^{-1} = I.$$

By A^{-T} we will denote the matrix $(A^{-1})^T = (A^T)^{-1}$. If A and B are nonsingular and the product AB defined, then

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

where the product of the inverse matrices are to be taken *in reverse order*.

Orthogonality

Recall that two vectors v and w in \mathbf{R}^n are said to be **orthogonal** with respect to the Euclidian inner product if $(v, w) = 0$. A set of vectors v_1, \dots, v_k in \mathbf{R}^n is called orthogonal if

$$v_i^T v_j = 0, \quad i \neq j,$$

and **orthonormal** if also $\|v_i\|_2 = 1, i = 1 : k$. An orthogonal set of vectors is linearly independent. More generally, a collection of subspaces S_1, \dots, S_k of \mathbf{R}^n are mutually orthogonal if for all $1 \leq i, j \leq k, i \neq j$,

$$x \in S_i, \quad y \in S_j \quad \Rightarrow \quad x^T y = 0.$$

The **orthogonal complement** S^\perp of a subspace $S \in \mathbf{R}^n$ is defined by

$$S^\perp = \{y \in \mathbf{R}^n \mid x^T y = 0, x \in S\}.$$

Let q_1, \dots, q_k be an orthonormal basis for a subspace $S \subset \mathbf{R}^n$. Such a basis can always be extended to a full orthonormal basis q_1, \dots, q_n for \mathbf{R}^n , and then $S^\perp = \text{span}\{q_{k+1}, \dots, q_n\}$.

Let $q_1, \dots, q_n \in \mathbf{R}^m, m \geq n$, be orthonormal. Then the matrix $Q = (q_1, \dots, q_n) \in \mathbf{R}^{m \times n}$, is orthogonal and $Q^T Q = I_n$. If Q is square ($m = n$) then $Q^{-1} = Q^T$, and hence $QQ^T = I_n$.

7.1.2 Submatrices and Block Matrices

A matrix formed by the elements at the intersection of a set of rows and columns of a matrix 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 submatrices of A . The second submatrix is called a contiguous submatrix since it is formed by contiguous elements of A .

Definition 7.1.2.

A **submatrix** of $A = (a_{ij}) \in \mathbf{R}^{m \times n}$ is a matrix $B \in \mathbf{R}^{p \times q}$ formed by selecting p rows and q columns of A ,

$$B = \begin{pmatrix} a_{i_1 j_1} & a_{i_1 j_2} & \cdots & a_{i_1 j_q} \\ a_{i_2 j_1} & a_{i_2 j_2} & \cdots & a_{i_2 j_q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i_p j_1} & a_{i_p j_2} & \cdots & a_{i_p j_q} \end{pmatrix},$$

where

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

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 .

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.,

$$A = \begin{array}{c} q_1 & q_2 & \cdots & q_N \\ p_1 \{ & \left(\begin{array}{cccc} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M1} & A_{M2} & \cdots & A_{MN} \end{array} \right), & p_2 \{ & \left(\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_M \end{array} \right), & (7.1.11) \\ p_2 \{ & & & & \\ \vdots & & & & \\ p_M \{ & & & & \end{array}$$

where A_{IJ} is a matrix of dimension $p_I \times q_J$. We call such a matrix 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

$$C_{IJ} = \sum_{K=1}^N A_{IK} B_{KJ}, \quad 1 \leq I \leq M, \quad 1 \leq J \leq P. \quad (7.1.12)$$

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 $[A(:, j), A(:, j + 1), \dots, A(:, k)]$,
$A(:)$	is all the elements of the matrix A regarded as a single column.

The various special forms of matrices have analogue block forms. For example a matrix R is block upper triangular if it has the form

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

Example 7.1.1.

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

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}. \quad (7.1.13)$$

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

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

Note that the product is again block upper triangular and its block diagonal simply equals the products of the diagonal blocks of the factors.

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

Definition 7.1.3.

A matrix $A \in \mathbf{R}^{n \times n}$, $n \geq 2$ is said to be **reducible** if for some permutation matrix P , $P^T AP$ has the block triangular form

$$P^T AP = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix}, \quad (7.1.14)$$

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

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

Block Elimination

Partitioning is a powerful tool for deriving algorithms and proving theorems. In this section we derive some useful formulas for solving linear systems where the matrix has been modified by a matrix of low rank.

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

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}, \quad U = \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}, \quad (7.1.15)$$

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

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{21}^{-1}L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}, \quad U^{-1} = \begin{pmatrix} U_{11}^{-1} & -U_{11}^{-1}U_{12}U_{22}^{-1} \\ 0 & U_{22}^{-1} \end{pmatrix}. \quad (7.1.16)$$

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

Consider a linear system $Mx = b$, written in block 2×2 form

$$\begin{aligned} Ax_1 + Bx_2 &= b_1, \\ Cx_1 + Dx_2 &= b_2. \end{aligned}$$

where A and D are square matrices. If A is 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. This is equivalent to **block Gaussian elimination** using the matrix A as pivot. The reduced system for x_2 becomes

$$(D - CA^{-1}B)x_2 = b_2 - CA^{-1}b_1. \quad (7.1.17)$$

If this system is solved for x_2 , we then obtain x_1 from $Ax_1 = b_1 - Bx_2$. The matrix

$$S = D - CA^{-1}B \quad (7.1.18)$$

is called the **Schur complement** of A in M .⁵

⁵Issai 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.

The elimination step can also be effected by premultiplying the system by the 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 in a product of a block lower and a block upper triangular matrix,

$$M = \begin{pmatrix} I & 0 \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ 0 & S \end{pmatrix}, \quad S = D - CA^{-1}B. \quad (7.1.19)$$

From $M^{-1} = (LU)^{-1} = U^{-1}L^{-1}$ using the formulas (7.1.16) for the inverses of 2×2 block triangular matrices we deduce the **Banachiewicz** inversion formula⁶

$$\begin{aligned} M^{-1} &= \begin{pmatrix} A^{-1} & -A^{-1}BS^{-1} \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix} \\ &= \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}. \end{aligned} \quad (7.1.20)$$

Similarly, assuming that D is nonsingular, we can factor M into a product of a block upper and a block lower triangular matrix

$$M = \begin{pmatrix} I & BD^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} T & 0 \\ C & D \end{pmatrix}, \quad T = A - BD^{-1}C, \quad (7.1.21)$$

where T is the Schur complement of D in M . (This is equivalent to block Gaussian elimination in reverse order.) From this factorization an alternative expression of M^{-1} can be derived,

$$M^{-1} = \begin{pmatrix} T^{-1} & -T^{-1}BD^{-1} \\ -D^{-1}CT^{-1} & D^{-1} + D^{-1}CT^{-1}BD^{-1} \end{pmatrix}. \quad (7.1.22)$$

If A and D are both nonsingular, then both triangular factorizations (7.1.19) and (7.1.21) exist.

Modified Linear Systems

It is well known that any matrix in $E \in \mathbf{R}^{n \times n}$ of rank p can be written as a product $E = BD^{-1}C$, where $B \in \mathbf{R}^{n \times p}$ and $C \in \mathbf{R}^{p \times n}$. (The third factor $D \in \mathbf{R}^{p \times p}$ has been added for convenience.) The following formula gives an expression for the inverse of a matrix A after it has been modified by a matrix of rank p .

⁶Tadeusz Banachiewicz (1882–1954) was a Polish astronomer and mathematician. In 1919 he became director of Cracow Observatory. In 1925 he developed a special kind of matrix algebra for “cracovians” which brought him international recognition.

Theorem 7.1.4.

Let A and D be square nonsingular matrices and let B and C be matrices of appropriate dimensions. If $(A - BD^{-1}C)$ exists and is nonsingular, then

$$(A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}, \quad (7.1.23)$$

which is the **Woodbury formula**.

Proof. The result follows directly by equating the $(1, 1)$ blocks in the inverse M^{-1} in (7.1.20) and (7.1.22). \square

The identity (7.1.23) appeared in several papers before Woodbury's report [617]. For a review of the history, see [317].

The Woodbury formula is very useful in situations where $p \ll n$. Frequently it is required to solve a linear system, where the matrix has been modified by a correction of low rank

$$(A - BD^{-1}C)\hat{x} = b, \quad B, C^T \in \mathbf{R}^{n \times p}, \quad (7.1.24)$$

with $D \in \mathbf{R}^{p \times p}$ nonsingular. Let $x = A^{-1}b$ be the solution to the unmodified system. Then, using the Woodbury formula, we have

$$(A - BD^{-1}C)^{-1}b = x + A^{-1}B(D - CA^{-1}B)^{-1}Cx. \quad (7.1.25)$$

This formula first requires computing the solution W of the linear system $AW = B$ with p right hand sides. The correction is then obtained by solving the linear system of size $p \times p$

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

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

In the special case that $D = \sigma \neq 0$ is a nonzero scalar the Woodbury formula (7.1.23) becomes

$$\left(A - \frac{1}{\sigma}uv^T\right)^{-1} = A^{-1} + \alpha A^{-1}uv^TA^{-1}, \quad \alpha = \frac{1}{\sigma - v^TA^{-1}u}, \quad (7.1.26)$$

where $u, v \in \mathbf{R}^n$. This is also known as the **Sherman–Morrison formula**. 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 linear system $Ax = b$ and \hat{x} the solution to the modified system

$$\left(A - \frac{1}{\sigma}uv^T\right)\hat{x} = b. \quad (7.1.27)$$

Using the Sherman–Morrison formula the solution of the can be written

$$\hat{x} = x + \beta w, \quad \beta = \frac{v^Tx}{\sigma - v^Tw}, \quad (7.1.28)$$

where $Aw = u$. Hence, w and \hat{x} can be computed without factorizing the modified matrix.

For a survey of applications of the Woodbury and the Sherman–Morrison formulas, see Hager [301]. Note that these should be used with caution since they can not be expected to be numerically stable in all cases. In particular, accuracy can be lost when the initial problem is more ill-conditioned than the modified one.

For some problems it is more relevant and convenient to work with complex vectors and matrices. For example, a real unsymmetric matrix in general has complex eigenvalues and eigenvectors. We denote by $\mathbf{C}^{m \times n}$ the vector space of all complex $m \times n$ matrices whose components are complex numbers. Most concepts introduced here carry over from the real to the complex case in a natural way. Addition and multiplication of vectors and matrices follow the same rules as before.

The complex inner product of two vectors x and y in \mathbf{C}^n is defined as

$$(x, y) = x^H y = \sum_{k=1}^n \bar{x}_k y_k, \quad x^H = (\bar{x}_1, \dots, \bar{x}_n), \quad (7.1.29)$$

and \bar{x}_i denotes the complex conjugate of x_i . It follows that $(x, y) = \overline{(y, x)}$. The Euclidean length of a vector $x \in \mathbf{C}^n$ is

$$\|x\|_2 = (x^H x)^{1/2} = \sqrt{\sum_{k=1}^n |x_k|^2}.$$

Two vectors x and y in \mathbf{C}^n are called orthogonal if $x^H y = 0$.

The Hermitian inner product leads to modifications in the definition of symmetric and orthogonal matrices. If $A = (a_{ij}) \in \mathbf{C}^{m \times n}$ the **adjoint** of A is denoted by $A^H \in \mathbf{C}^{n \times m}$ since

$$(x, A^H y) = (Ax, y).$$

By using coordinate vectors for x and y it follows that $A^H = (\bar{a}_{ji})$, that is, A^H is the conjugate transpose of A . For example,

$$A = \begin{pmatrix} 0 & i \\ 2i & 0 \end{pmatrix}, \quad A^H = \begin{pmatrix} 0 & -2i \\ -i & 0 \end{pmatrix}.$$

It is easily verified that $(AB)^H = B^H A^H$. In particular, if α is a scalar $\alpha^H = \bar{\alpha}$.

A complex matrix $A \in \mathbf{C}^{n \times n}$ is called **Hermitian** if $A^H = A$ and **skew-Hermitian** if $A^H = -A$. It is called self-adjoint or **Hermitian** if $A^H = A$. A Hermitian matrix has analogous properties to a real symmetric matrix. If A is Hermitian, then $(x^H Ax)^H = x^H Ax$ is real, and A is called positive definite if

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

Note that in every case, the new definitions coincides with the old when the vectors and matrices are real.

A square matrix U for which $U^H U = I$ is called **unitary**, and has the property that

$$(Ux)^H Uy = x^H U^H Uy = x^H y,$$

From (7.1.29) it follows that unitary matrices preserve the Hermitian inner product. In particular, the Euclidian length of a vector is invariant under unitary transformations, i.e., $\|Ux\|_2^2 = \|x\|_2^2$. Note that when the vectors and matrices are real the definitions for the complex case are consistent with those made for the real case.

7.1.3 Permutations and Determinants

A **permutation matrix** $P \in \mathbf{R}^{n \times n}$ is a matrix whose columns are a permutation of the columns of the unit matrix, that is,

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

where p_1, \dots, p_n is a permutation of $1 : n$. A permutation matrix contains just one unit element in every row and every column. The transpose P^T of a permutation matrix is again a permutation matrix. The product of two permutations p and q is the composition defined by

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

and the corresponding permutation matrix is the matrix product PQ .

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

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

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, but this decomposition is not unique. Thus, any permutation matrix can be expressed as a product of transposition matrices $P = I_{i_1,j_1} I_{i_2,j_2} \cdots I_{i_k,j_k}$. Since $I_{i_p,j_p}^{-1} = I_{i_p,j_p}$, we have

$$P^{-1} = I_{i_k,j_k} \cdots I_{i_2,j_2} I_{i_1,j_1} = P^T,$$

and

$$P^T P = P P^T = I. \quad (7.1.30)$$

Thus, permutation matrices are orthogonal and P^T effects the reverse permutation. If P is a permutation matrix then PA is the matrix A with its rows permuted and AP^T performs the same permutation on the columns of A .

A permutation matrix P can be uniquely represented by the integer vector $p = (p_1, \dots, p_n)$, and it need not be explicitly stored. For example, the vector $p = (2, 4, 1, 3)$ represents the permutation matrix

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

Example 7.1.2.

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 rows of a matrix A as $A(p, :)$. Similarly, $AP' = A(:, p)$

performs the same permutation on the columns of A . The permutation matrix P corresponding to p is $\mathbf{I}(p, :)$ and P^T is $\mathbf{I}(:, p)$. Also p is $(P * (1 : n)')'$ or $(1 : n) * P'$.

The classical definition of the **determinant**⁷ 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 pair α_r, α_s , $r < s$, is said to form an inversion in the permutation if $\alpha_r > \alpha_s$. For example, in the permutation $\{2, \dots, n, 1\}$ there are $(n-1)$ inversions $(2, 1), (3, 1), \dots, (n, 1)$. A permutation α is said to be even and $\text{sign}(\alpha) = 1$ if it contains an even number of inversions; otherwise the permutation is odd and $\text{sign}(\alpha) = -1$.

Lemma 7.1.5.

A transposition τ of a permutation will change the number of inversions in the permutation by an odd number and thus $\text{sign}(\tau) = -1$.

Proof. If τ interchanges two adjacent elements α_r and α_{r+1} in the permutation $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$, this will not affect inversions in other elements. Hence, the number of inversions increases by 1 if $\alpha_r < \alpha_{r+1}$ and decreases by 1 otherwise. Suppose now that τ interchanges α_r and α_{r+q} . This can be achieved by first successively interchanging α_r with α_{r+1} , then with α_{r+2} , and finally with α_{r+q} . This takes q steps. Next the element α_{r+q} is moved in $q-1$ steps to the position which α_r previously had. In all it takes an *odd number* $2q-1$ of transpositions of adjacent elements, in each of which the sign of the permutation changes. \square

Definition 7.1.6.

The determinant of a square matrix $A \in \mathbf{R}^{n \times n}$ is the scalar

$$\det(A) = \sum_{\alpha \in S_n} \text{sign}(\alpha) a_{1,\alpha_1} a_{2,\alpha_2} \cdots a_{n,\alpha_n}, \quad (7.1.31)$$

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

Note that there are $n!$ terms in (7.1.31) and each term contains exactly one factor from each row and each column in A . For example, if $n = 2$ there are two terms, and

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

From the definition it follows easily that

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

⁷Determinants were first introduced by Leibniz (1693) and Cayley (1841). Determinants arise in many parts of mathematics, such as combinatorial enumeration, graph theory, representation theory, statistics, and theoretical computer science. 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).

IF we collect all terms in (7.1.31) that contains the element a_{rs} these can be written as $a_{rs}A_{rs}$, where A_{rs} is called the complement of a_{rs} . Since the determinant contains only one element from row r and column s the complement A_{rs} does not depend on any elements in row r and column s . Since each product in (7.1.31) contains precisely one element of the elements $a_{r1}, a_{r2}, \dots, a_{rn}$ in row r it follows that

$$\det(A) = a_{r1}A_{r1} + a_{r2}A_{r2} + \cdots + a_{rn}A_{rn}. \quad (7.1.32)$$

This is called to expand the determinant after the row r . It is not difficult to verify that

$$A_{rs} = (-1)^{r+s}D_{rs}, \quad (7.1.33)$$

where D_{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 we can similarly expand $\det(A)$ after a column.

Another scalar-valued function of a matrix is the **permanent**. Its definition is similar to that of the determinant, but in the sum (7.1.31) all terms are to be taken with positive sign; see Marcus and Minc [423, Sec. 2.9]. The permanent has no easy geometric interpretation and is used mainly in combinatorics. The permanent is more difficult to compute than the determinant, which can be computed in polynomial time.

The direct use of the definition (7.1.31) to evaluate $\det(A)$ would require about $nn!$ operations, which rapidly becomes infeasible as n increases. A much more efficient way to compute $\det(A)$ is by repeatedly using the following properties:

Theorem 7.1.7.

- (i) *The value of the $\det(A)$ 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 is upper triangular*

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

- (iii) *If two rows (columns) in A are interchanged the value of $\det(A)$ is multiplied by (-1) .*

- (iv) *The product rule $\det(AB) = \det(A)\det(B)$.*

If Q is a square orthogonal matrix then $Q^T Q = I$, and using (iv) it follows that

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

Hence, $\det(Q) = \pm 1$. If $\det(Q) = 1$, then Q represents a rotation.

Theorem 7.1.8.

The matrix A is nonsingular if and only if $\det(A) \neq 0$. If the matrix A is nonsingular, then the solution of the linear system $Ax = b$ can be expressed as

$$x_j = \det(B_j)/\det(A), \quad j = 1 : n. \quad (7.1.34)$$

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

Proof. Form the linear combination

$$a_{1j}A_{1r} + a_{2j}A_{2r} + \cdots + a_{nj}A_{nr} = \begin{cases} 0 & \text{if } j \neq r, \\ \det(A) & \text{if } j = r. \end{cases} \quad (7.1.35)$$

with elements from column j and the complements of column r . If $j = r$ this is an expansion after column r of $\det(A)$. If $j \neq r$ the expression the expansion of the determinant of a matrix equal to A except that column r is equal to column j . Such a matrix has determinant equal to 0.

Now 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 (7.1.35) 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 (7.1.34). \square

The expression (7.1.34) is known as **Cramer's rule**.⁸ Although elegant, it is both computationally expensive and numerically unstable even for $n = 2$; see Higham [328, p. 13].

Let U be an upper block triangular matrix with square diagonal blocks $U_{II}, I = 1 : N$. Then

$$\det(U) = \det(U_{11})\det(U_{22}) \cdots \det(U_{NN}). \quad (7.1.36)$$

and 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.

Example 7.1.3.

For the 2×2 block matrix M in (7.1.19) and (7.1.21) it follows using (7.1.36) that

$$\det(M) = \det(A - BD^{-1}C)\det(D) = \det(A)\det(D - CA^{-1}B).$$

In the special case that $D^{-1} = \lambda$, $B = x$, and $B = y$, this gives

$$\det(A - \lambda xy^T) = \det(A)(1 - \lambda y^T A^{-1}x). \quad (7.1.37)$$

⁸Named after the Swiss mathematician Gabriel Cramer 1704–1752.

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

The following determinant inequality, which relates the determinant to a volume is due to Hadamard.⁹

Theorem 7.1.9 (*Hadamard's Determinant Inequality*).

Let $a_j = Ae_j$, $j = 1 : n$, be the j th column of a matrix $A \in \mathbf{R}^{n \times n}$. Then

$$|\det(A)| \leq \prod_{j=1}^n \|a_j\|_2, \quad (7.1.38)$$

with equality only if $A^T A$ is a diagonal matrix or A has a zero column.

Proof. Let $A = QR$ be the QR factorization of A . (see Section 8.3). Since the determinant of an orthogonal matrix equals ± 1 , and the determinant of a triangular matrix is the product of the diagonal elements, we have

$$|\det(A)| = |\det(Q)| \cdot |\det(R)| = |\det(R)| = \prod_{j=1}^n |r_{jj}|.$$

The inequality now follows from $|r_{jj}| \leq \|r_j\|_2 = \|a_j\|_2$, where $r_j = Re_j$ is the j th column of R . If $\det(A) > 0$, equality holds if and only if $|r_{jj}| = \|r_j\|_2$, $j = 1 : n$. This implies that R is diagonal, i.e., the columns in A are mutually orthogonal. If $\det(A) = 0$, then the right hand side equals zero if and only if at least one column in A is zero. \square

Another determinant identity that is a useful tool in many algebraic manipulations is the following:

Theorem 7.1.10 (*Sylvester's Determinant Identity*).

Let $\hat{A} \in \mathbf{R}^{n \times n}$, $n \geq 2$, be partitioned

$$\begin{aligned} \hat{A} &= \begin{pmatrix} \alpha_{11} & a_1^T & \alpha_{12} \\ \hat{\alpha}_{11} & A & \hat{\alpha}_2 \\ \alpha_{21} & a_2^T & \alpha_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & * \\ * & \alpha_{22} \end{pmatrix} = \begin{pmatrix} * & A_{12} \\ \alpha_{21} & * \end{pmatrix} \\ &= \begin{pmatrix} * & \alpha_{12} \\ A_{21} & * \end{pmatrix} = \begin{pmatrix} \alpha_{11} & * \\ * & A_{21} \end{pmatrix}. \end{aligned}$$

Then we have the identity

$$\det(A) \cdot \det(\hat{A}) = \det(A_{11}) \cdot \det(A_{22}) - \det(A_{12}) \cdot \det(A_{21}). \quad (7.1.39)$$

⁹ Jacques Salomon Hadamard (1865–1963) was a French mathematician active at the Sorbonne, Collège de France and École Polytechnique in Paris. He made important contributions to geodesics of surfaces and functional analysis. He gave a proof of the result that the number of primes $\leq n$ tends to infinity as $n/\ln n$.

Proof. See [132, p. 341]. \square

7.1.4 Vectors 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 \mathbf{R} .

Definition 7.1.11.

A norm on a vector space $\mathbf{V} \in \mathbf{C}^n$ is a function $\mathbf{V} \rightarrow \mathbf{R}$ denoted by $\|\cdot\|$ that satisfies the following three conditions:

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

The triangle inequality is often used in the form (see Problem 7.1.12)

$$\|x \pm y\| \geq |\|x\| - \|y\||.$$

The most common vector norms are special cases of the family of Hölder norms or p -norms

$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}, \quad 1 \leq p < \infty. \quad (7.1.40)$$

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

$$\begin{aligned} \|x\|_1 &= |x_1| + \cdots + |x_n|, \\ \|x\|_2 &= (|x_1|^2 + \cdots + |x_n|^2)^{1/2} = (x^H x)^{1/2}, \\ \|x\|_\infty &= \max_{1 \leq i \leq n} |x_i|. \end{aligned} \quad (7.1.41)$$

The vector 2-norm is also called the Euclidean norm. It is invariant under unitary (unitary) transformations since

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

if Q is unitary.

The proof that the triangle inequality is satisfied for the p -norms depends on the following inequality. Let $p > 1$ and q satisfy $1/p + 1/q = 1$. Then it holds that

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

Indeed, let x and y be any real number and λ satisfy $0 < \lambda < 1$. Then by the convexity of the exponential function it holds that

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

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

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

$$|x^H y| \leq \|x\|_p \|y\|_q, \quad \frac{1}{p} + \frac{1}{q} = 1, \quad p \geq 1. \quad (7.1.42)$$

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

$$|x^H y| = \left| \sum_{i=1}^n x_i^H y_i \right| \leq \sum_{i=1}^n |x_i^H y_i| \leq \max_i |y_i| \sum_{i=1}^n |x_i| = \|x\|_1 \|y\|_\infty. \quad (7.1.43)$$

Definition 7.1.12.

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

$$\|x\|_D = \max_{y \neq 0} |x^H y| / \|y\|. \quad (7.1.44)$$

The vectors in the set

$$\{y \in \mathbf{C}^n \mid \|y\|_D \|x\| = y^H x = 1\}. \quad (7.1.45)$$

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 [552, Theorem II.1.12]). It follows from the Hölder inequality that the dual of the p -norm is the q -norm, where

$$1/p + 1/q = 1.$$

The dual of the 2-norm can be seen to be itself. It can be shown to be the only norm with this property (see [338, Theorem 5.4.16]).

The vector 2-norm can be generalized by taking

$$\|x\|_{2,G}^2 = (x, Gx) = x^H G x, \quad (7.1.46)$$

where G is a Hermitian positive definite matrix. It can be shown that the unit ball $\{x : \|x\| \leq 1\}$ corresponding to this norm is an ellipsoid, and hence they are also called **elliptic norms**. Other generalized norms are the **scaled p -norms** defined by

$$\|x\|_{p,D} = \|Dx\|_p, \quad D = \text{diag}(d_1, \dots, d_n), \quad d_i \neq 0, \quad i = 1 : n. \quad (7.1.47)$$

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

$$\frac{1}{c}\|x\|' \leq \|x\| \leq c'\|x\|' \quad \forall x \in \mathbf{C}^n. \quad (7.1.48)$$

In particular, it can be shown that for the p -norms we have

$$\|x\|_q \leq \|x\|_p \leq n^{(1/p-1/q)}\|x\|_q, \quad 1 \leq p \leq q \leq \infty. \quad (7.1.49)$$

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

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

We now consider **matrix norms**. Given any vector norm, we can construct a matrix norm by defining

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|. \quad (7.1.50)$$

This norm is called the **operator norm**, or the matrix norm **subordinate** to the vector norm. From the definition it follows directly that

$$\|Ax\| \leq \|A\| \|x\| \quad \forall x \in \mathbf{C}^n. \quad (7.1.51)$$

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

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

$$4. \quad \|AB\| \leq \|A\| \|B\|$$

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

Theorem 7.1.13.

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

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|, \quad (7.1.52)$$

$$\|A\|_2 = \max_{\|x\|=1} (x^H A^H A x)^{1/2} = \sigma_1(A), \quad (7.1.53)$$

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|. \quad (7.1.54)$$

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 $\|Ax\|_1 \leq \max_{1 \leq j \leq n} \|a_j\|_1 = \|a_k\|_1$, 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 (7.1.52). The formula (7.1.54) for the matrix ∞ -norm is proved in a similar fashion. The expression for the 2-norm follows from the extremal property

$$\max_{\|x\|=1} \|Ax\|_2 = \max_{\|x\|=1} \|U\Sigma V^H x\|_2 = \sigma_1(A)$$

of the singular vector $x = v_1$. \square

For $p = 1$ and $p = \infty$ the matrix subordinate norms are easily computable. Note that the 1-norm is the maximal column sum and the ∞ -norm is the maximal row sum of the magnitude of the elements. It follows that $\|A\|_1 = \|A^H\|_\infty$.

The 2-norm, also called the **spectral norm**, equals the largest singular value $\sigma_1(A)$ of A . It has the drawback that it is expensive to compute, but is a useful analytical tool. Since the nonzero eigenvalues of $A^H A$ and AA^H are the same it follows that $\|A\|_2 = \|A^H\|_2$. An upper bound for the matrix 2-norm is

$$\|A\|_2 \leq (\|A\|_1 \|A\|_\infty)^{1/2}. \quad (7.1.55)$$

The proof of this bound is given as an exercise in Problem 7.1.18.

A vector norm is called **absolute** if

$$\|x\| = \||x|\|.$$

Clearly, the vector p -norms are absolute for all $1 \leq p < \infty$. A vector norm is called **monotone** if

$$|x| \leq |y| \Rightarrow \|x\| \leq \|y\|.$$

The matrix norm subordinate to an absolute vector norm has the property

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

It can be shown that these three properties are equivalent; see Householder [343, Section 2.3].

Another way to proceed in defining norms for matrices is to regard $\mathbf{C}^{m \times n}$ as an mn -dimensional vector space and apply a vector norm over that space.

Definition 7.1.14.

The **Frobenius norm**¹⁰ is derived from the vector 2-norm

$$\|A\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{1/2} \quad (7.1.56)$$

¹⁰Ferdinand George Frobenius (1849–1917), German mathematician, professor at ETH Zürich (1875–1892) before he succeeded Weierstrass at Berlin University.

The Frobenius norm is submultiplicative, but is often larger than necessary, e.g., $\|I_n\|_F = n^{1/2}$. This tends to make bounds derived in terms of the Frobenius norm not as sharp as they might be. From (7.1.61) it follows that

$$\frac{1}{\sqrt{n}} \|A\|_F \leq \|A\|_2 \leq \|A\|_F, \quad k = \min(m, n). \quad (7.1.57)$$

Note that $\|A^H\|_F = \|A\|_F$.

The **trace** of a matrix $A \in \mathbf{R}^{n \times n}$ is defined by

$$\text{trace}(A) = \sum_{i=1}^n a_{ii}. \quad (7.1.58)$$

Useful properties are of this function are

$$\text{trace}(S^{-1}AS) = \text{trace}(A), \quad (7.1.59)$$

$$\text{trace}(AB) = \text{trace}(BA). \quad (7.1.60)$$

An alternative characterization of the Frobenius norm is

$$\|A\|_F^2 = \text{trace}(A^H A) = \sum_{i=1}^k \sigma_i^2(A), \quad k = \min(m, n), \quad (7.1.61)$$

where $\sigma_i(A)$ are the nonzero singular values of A . Of the matrix norms the 1 - ∞ - and the Frobenius norm are absolute, but for the 2 -norm the best result is

$$\| |A| \|_2 \leq n^{1/2} \|A\|_2.$$

Table 7.1.1. Numbers γ_{pq} such that $\|A\|_p \leq \gamma_{pq} \|A\|_q$, where $A \in \mathbf{C}^{m \times n}$ 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

The Frobenius norm shares with the 2 -norm the property of being invariant with respect to unitary (orthogonal) transformations

$$\|QAP\| = \|A\|. \quad (7.1.62)$$

Such norms are called **unitarily invariant**.

One use of norms is in the study of *limits of sequences of vectors and matrices* (see Section 9.2.4). Consider an infinite sequence of vectors in \mathbf{C}^n

$$x_k = (\xi_1^{(k)} \quad \xi_2^{(k)} \cdots \xi_n^{(k)})^T, \quad k = 1, 2, 3, \dots$$

Then it is natural to say that the vector sequence converges to a vector $x = (\xi_1 \ \xi_2 \cdots \xi_n)$ if each component converges

$$\lim_{k \rightarrow \infty} \xi_i^{(k)} = \xi_i, \quad i = 1 : n.$$

Another useful way of defining convergence is to use a norm $\|\cdot\|$ on \mathbf{C}^n . The sequence is said to converge and we write $\lim_{k \rightarrow \infty} x_k = x$ if

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

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 \mathbf{C}^n is equivalent to convergence of the n sequences of scalars formed by the components of the vectors. By considering matrices in $\mathbf{C}^{m \times n}$ as vectors in \mathbf{C}^{mn} the same conclusion holds for matrices. We define the limit of a sequence of matrices as follows.

Definition 7.1.15.

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

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

From the equivalence of norms in a finite dimensional vector space it follows that convergence is independent of the choice of norm. The particular choice $\|\cdot\|_\infty$ shows that convergence of vectors in \mathbf{R}^n is equivalent to convergence of the n sequences of scalars formed by the components of the vectors. By considering matrices in $\mathbf{R}^{m \times n}$ as vectors in \mathbf{R}^{mn} the same conclusion holds for matrices.

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 $z \in \mathbf{C}$.

Theorem 7.1.16.

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

Example 7.1.4.

An approximative 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**

$$(I - B)^{-1} = I + B + B^2 + B^3 + \cdots, \quad (7.1.63)$$

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**

$$(I - B)^{-1} = (I + B)(I + B^2)(I + B^4) \cdots. \quad (7.1.64)$$

By induction it follows that

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

7.1.5 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 diagonal form of a real (or complex) matrix A under an orthogonal (unitary) equivalence transformation. Although substantially 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. The history of the SVD goes back more than a century. Its use in numerical computations was not practical until an algorithm to stably compute it was developed in the late 1960s; see Section 10.8.6.

Theorem 7.1.17. (Singular Value Decomposition.)

Every matrix $A \in \mathbf{C}^{m \times n}$ of rank r can be written

$$A = U\Sigma V^H = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^H \\ V_2^H \end{pmatrix}, \quad (7.1.65)$$

where $U \in \mathbf{C}^{m \times m}$ and $V \in \mathbf{C}^{n \times n}$ are unitary matrices, $\Sigma_1 = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$, and

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0.$$

(Note that if $r = n$ and/or $r = m$, some of the zero submatrices in Σ disappear.) The σ_i are called the **singular values** of A and if we write

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

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

Proof. Let $f(x) = \|Ax\|_2 = (x^H A^H A x)^{1/2}$ be the Euclidian length of the vector $y = Ax$, and consider the problem

$$\sigma_1 := \max_x \{f(x) \mid x \in \mathbf{C}^n, \|x\|_2 \leq 1\}.$$

Here $f(x)$ is a real-valued convex function¹¹ defined on a convex, compact set. It is well known (see, e.g., Ciarlet [110, Sec. 7.4]) that the maximum σ_1 is then attained on an extreme point of the set. Let v_1 be such a point with $\sigma_1 = \|Av_1\|$, $\|v_1\|_2 = 1$. If $\sigma_1 = 0$ then $A = 0$, and (7.1.65) holds with $\Sigma = 0$, and U and V arbitrary unitary matrices. Therefore, assume that $\sigma_1 > 0$, and set $u_1 = (1/\sigma_1)Av_1 \in \mathbf{C}^m$, $\|u_1\|_2 = 1$. Let the matrices

$$V = (v_1, V_1) \in \mathbf{C}^{n \times n}, \quad U = (u_1, U_1) \in \mathbf{C}^{m \times m}$$

be unitary. (Recall that it is always possible to extend an unitary set of vectors to a unitary basis for the whole space.) Since $U_1^H Av_1 = \sigma_1 U_1^H u_1 = 0$ it follows that $U^H A V$ has the following structure:

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

where $w^H = u_1^H A V_1$ and $B = U_1^H A V_1 \in \mathbf{C}^{(m-1) \times (n-1)}$.

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

We have $U A_1 y = A V y = A x$ and, since U and V are unitary, it follows that

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

and hence,

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

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

The SVD can be written in the **compact form**

$$A = U_1 \Sigma_1 V_1^H = \sum_{i=1}^r \sigma_i u_i v_i^H, \tag{7.1.66}$$

where the last expression expresses A as a sum of matrices of rank one.

¹¹A 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 $f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$.

Definition 7.1.18.

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

$$\mathcal{R}(A) = \{y \in \mathbf{C}^m \mid y = Ax, x \in \mathbf{C}^n\}. \quad (7.1.67)$$

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

$$\mathcal{N}(A) = \{x \in \mathbf{C}^n \mid Ax = 0\}. \quad (7.1.68)$$

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

$$\mathcal{R}(A) = \mathcal{R}(U_1), \quad \mathcal{N}(A^H) = \mathcal{R}(U_2), \quad (7.1.69)$$

$$\mathcal{R}(A^H) = \mathcal{R}(V_1), \quad \mathcal{N}(A) = \mathcal{R}(V_2). \quad (7.1.70)$$

For the sake of generality we have formulated the SVD for a complex matrix although in most applications A is real. If $A \in \mathbf{R}^{m \times n}$, then U and V are real orthogonal matrices. The geometrical significance of this theorem is as follows. The rectangular matrix A represents a mapping from \mathbf{C}^n to \mathbf{C}^m . The theorem shows that, there is an unitary basis in each of these two spaces, with respect to which this mapping is represented by a generalized diagonal matrix Σ .

The singular values of A are uniquely determined. The singular vector v_j , $j \leq r$, is unique (up to a factor ± 1) if σ_j^2 is a *simple* singular value. For multiple singular values the corresponding singular vectors can be chosen as any orthonormal basis for the unique subspace that they span. 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, using

$$Av_j = \sigma_j u_j, \quad A^H u_j = \sigma_j v_j, \quad j = 1, \dots, r. \quad (7.1.71)$$

Norms like the 2-norm and the Frobenius norm, which are invariant with respect to unitary (orthogonal) transformations have an interesting history; see Stewart and Sun [552, Sec. II.3]. Then can be characterized as follows.

Theorem 7.1.19.

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

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

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

Unitarily invariant norms were characterized by von Neumann [449], who showed that the converse of Theorem 7.1.19 is true: A function $\Phi(\sigma_1, \dots, \sigma_n)$

which is symmetric in its arguments and satisfies the three properties in the Definition 7.1.11 of a vector norm defines a unitarily invariant matrix norm. In this connection such functions are called **symmetric gauge functions**. Two examples are

$$\|A\|_2 = \max_i \sigma_i, \quad \|A\|_F = \left(\sum_{i=1}^n \sigma_i^2 \right)^{1/2}.$$

7.1.6 Numerical Rank

Inaccuracy of data and rounding errors made during the computation usually perturb the ideal matrix A . In this situation the *mathematical* notion of rank may not be appropriate. For example, let A be a matrix of rank $r < n$, whose elements are perturbed by a matrix E of small random errors. Then it is most likely that the perturbed matrix $A + E$ has full rank n . However, $A + E$ is close to a rank deficient matrix, and should be considered as *numerically rank deficient*.

In solving linear systems and linear least squares problems failure to detect ill-conditioning and possible rank deficiency in A can lead to a meaningless solution of very large norm, or even to breakdown of the numerical algorithm.

Clearly 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 is the following:

Definition 7.1.20.

A matrix $A \in \mathbf{R}^{m \times n}$ has numerical δ -rank equal to k ($k \leq \min\{m, n\}$) if

$$\sigma_1 \geq \dots \geq \sigma_k > \delta \geq \sigma_{k+1} \geq \dots \geq \sigma_n,$$

where σ_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 8.1.16, 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 7.1.20 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. However, it may occur 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!

If $r < n$ then the system is *numerically underdetermined*. Note that this can be the case even when $m > n$.

Failure to detect possible rank deficiency can be catastrophic, and may lead to a meaningless solution of very large norm, or even to failure of an algorithm.

Example 7.1.5.

Consider an example based on the integral equation of the first kind

$$\int_0^1 k(s, t)f(s) ds = g(t), \quad k(s, t) = e^{-(s-t)^2}, \quad (7.1.72)$$

on $-1 \leq t \leq 1$. In order to solve this equation numerically it must first be discretized. We introduce a uniform mesh for s and t on $[-1, 1]$ with step size $h = 2/n$, $s_i = -1 + ih$, $t_j = -1 + jh$, $i, j = 0 : n$. Approximating the integral with the trapezoidal rule gives

$$h \sum_{i=0}^n w_i k(s_i, t_j) f(t_i) = g(t_j), \quad j = 0 : n.$$

where $w_i = 1$, $i \neq 0, n$ and $w_0 = w_n = 1/2$. These equations form a linear system

$$Kf = g, \quad K \in \mathbf{R}^{(n+1) \times (n+1)}, \quad f, g \in \mathbf{R}^{n+1}.$$

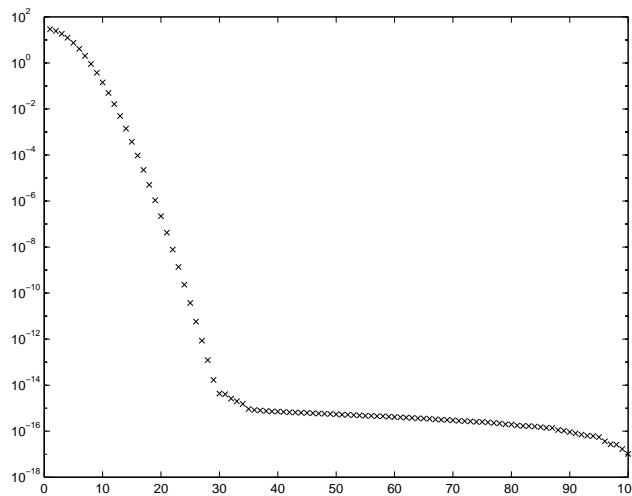


Figure 7.1.1. Singular values of a numerically singular matrix.

For $n = 100$ the singular values σ_k of the matrix K computed in IEEE double precision are displayed in logarithmic scale in Figure 8.4.1. Note that for $k > 30$ all σ_k are close to roundoff level, so the numerical rank of K certainly is smaller than 30. This means that the linear system $Kf = g$ is *numerically under-determined* and has a meaningful solution only for special right-hand sides g .

The equation (7.1.72) is a **Fredholm integral equation** of the first kind. It is known that such equations are **ill-posed** in the sense that the solution f does not depend continuously on the right hand side g . This example illustrate how this inherent difficulty in the continuous problem carry over to the discretized problem!

The choice of the parameter δ in Definition 7.1.20 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 7.1.20 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.)

7.1.7 Matrix Multiplication

It is important to know roughly how much work is required by different matrix algorithms. As an example we take matrix multiplication. If $A \in \mathbf{R}^{m \times n}$ and $B \in \mathbf{R}^{n \times p}$, then by inspection of (7.1.4) it is seen that computing the mp elements c_{ij} of the product $C = AB$ can be made in mnp additions and the same number of multiplications.

In a product of more than two matrices is to be computed the number of operations will depend on the ordering of the products. If $C \in \mathbf{R}^{p \times q}$, then the triple product $M = ABC \in \mathbf{R}^{m \times q}$ 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 $x \in \mathbf{R}^n$ a column vector, then computing $(AB)x$ requires $n^3 + n^2$ multiplications whereas $A(Bx)$ only requires $2n^2$ multiplications. When $n \gg 1$ this makes a great difference!

In older textbooks a **flop** usually 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. In more recent textbooks (e.g., Golub and Van Loan [277]) a flop is defined as one floating-point operation doubling the older flop counts.¹² It is usually ignored that on many computers a division is 5–10 times slower than a multiplication. Using the new convention multiplication $C = AB$ of two square matrices of order n requires $2n^3$ flops. The matrix-vector multiplication $y = Ax$, where $A \in \mathbf{R}^{n \times n}$ and $x \in \mathbf{R}^n$, requires $2mn$ flops.¹³

Operation counts like these are meant only as a rough appraisal of the work and one should not assign too much meaning to their precise value. Usually one only considers the highest-order term(s). On modern computer architectures the rate of transfer of data between different levels of memory often limits the actual performance and the data access patterns are very important. Some times the execution times of algorithms with the same flop count can differ by an order of magnitude. An operation count still provides useful information, and can serve as an initial basis of comparison of different algorithms. It implies that the running time for multiplying two square matrices on a computer roughly will increase cubically with the dimension n . Thus, doubling n will approximately increase the work by a

¹²Stewart [549, p. 96] uses **flop** (floating-point addition and multiplication) to denote an “old” flop.

¹³To add to the confusion, in computer literature flops means floating-point operations per second.

factor of eight.

When implementing a matrix algorithm such as (7.2.1) or (7.2.2) on a computer, the *order of operations* in matrix algorithms may be important. One reason for this is the economizing of storage, since even matrices of moderate dimensions have a large number of elements. When the initial data is not needed for future use, computed quantities may overwrite data. To resolve 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 either use MATLAB or an informal programming language, which is sufficiently precise for our purpose but allows the suppression of cumbersome details.

Let $A \in \mathbf{R}^{m \times p}$ be partitioned into columns and $B \in \mathbf{R}^{p \times n}$ into rows. Then the matrix product $C = AB \in \mathbf{R}^{m \times n}$ can be written as

$$C = AB = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{pmatrix} (b_1 \ b_2 \ \cdots \ b_n) = (c_{ij}), \quad c_{ij} = a_i^T b_j. \quad (7.1.73)$$

with $a_i, b_j \in \mathbf{R}^p$. Here each element c_{ij} is expressed as an inner product. A MATLAB script expressing this can be formulated as follows:

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

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

$$C = AB = (a_1 \ a_2 \ \cdots \ a_p) \begin{pmatrix} b_1^T \\ b_2^T \\ \vdots \\ b_p^T \end{pmatrix} = \sum_{k=1}^p a_k b_k^T, \quad (7.1.74)$$

where each term in the sum of (7.1.74) is an *outer product*. A code expressing this is

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

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

A faster method for matrix multiplication would give more efficient algorithms for many linear algebra problems such as inverting matrices, solving linear systems

and eigenvalue problems. An intriguing question is whether it is possible to multiply two matrices $A, B \in \mathbf{R}^{n \times n}$ (or solve a linear system of order n) in less than n^3 (scalar) multiplications. The answer is yes! Strassen [558] developed a fast algorithm for matrix multiplication, which if used recursively to multiply two square matrices of dimension $n = 2^k$, the number of multiplications is reduced from n^3 to $n^{\log_2 7} = n^{2.807\dots}$; see Section 7.6.3.

It is still an open (difficult!) question what the minimum exponent ω is such that matrix multiplication can be done in $O(n^\omega)$ operations. The fastest known algorithm, devised in 1987 by Coppersmith and Winograd [118], 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 [504]. (Note that for many of the theoretically “fast” methods large constants are hidden in the O notation.)

7.1.8 Floating-Point Arithmetic

The IEEE 754 standard (see [209]) for floating-point arithmetic is now used for all personal computers and workstations. It specifies basic and extended formats for floating-point numbers, elementary operations and rounding rules available, conversion between different number formats, and binary-decimal conversion. The handling of exceptional cases like exponent overflow or underflow and division by zero are also specified.

Two main basic formats, single and double precision are defined, using 32 and 64 bits respectively. In **single precision** a floating-point number a is stored as a sign s (one bit), the exponent e (8 bits), and the mantissa m (23 bits). In **double precision** of the 64 bits 11 are used for the exponent, and 52 bits for the mantissa. The value v of a is in the normal case

$$v = (-1)^s(1.m)_2 2^e, \quad -e_{\min} \leq e \leq e_{\max}. \quad (7.1.75)$$

Note that the digit before the binary point is always 1 for a normalized number. A biased exponent is stored and no sign bit used for the exponent. In single precision $e_{\min} = -126$ and $e_{\max} = 127$ and $e + 127$ is stored.

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 is also supported as well as directed rounding to ∞ and to $-\infty$. The latter mode simplifies the implementation of interval arithmetic.

In a floating-point number system every real number in the floating-point range of F can be represented with a relative error, which does not exceed the **unit roundoff** u . For IEEE floating point arithmetic the unit roundoff equals

$$u = \begin{cases} 2^{-24} \approx 5.96 \cdot 10^{-8}, & \text{in single precision;} \\ 2^{-53} \approx 1.11 \cdot 10^{-16} & \text{in double precision.} \end{cases}$$

The largest number that can be represented is approximately $2.0 \cdot 2^{127} \approx 3.4028 \times 10^{38}$ in single precision and $2.0 \cdot 2^{1023} \approx 1.7977 \times 10^{308}$ in double precision. The

smallest normalized number is $1.0 \cdot 2^{-126} \approx 1.1755 \times 10^{-38}$ in single precision and $1.0 \cdot 2^{-1022} \approx 2.2251 \times 10^{-308}$ in double precision. For more details on the IEEE floating-point standard and floating point arithmetic, see Volume I, Section 2.3.

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 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, it holds that

$$fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad |\delta| \leq u, \quad (7.1.76)$$

where u is the unit roundoff and “op” stands for one of the four elementary operations $+$, $-$, \cdot , and $/$. Further,

$$fl(\sqrt{x}) = \sqrt{x}(1 + \delta), \quad |\delta| \leq u, \quad (7.1.77)$$

Bounds for roundoff errors for basic vector and matrix operations can easily be derived using the following basic result: (see Wilkinson [610, pp. 23–25], [611, pp. 114–118]):

Lemma 7.1.21. [Higham [328, Lemma 3.1]]

Let $|\delta_i| \leq u$, $\rho_i = \pm 1$, $i = 1:n$, and

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

If $nu < 1$, then $|\theta_n| < \gamma_n$, where $\gamma_n = nu/(1 - nu)$.

If we make the realistic assumption that $nu < 0.1$, then it holds that

$$\epsilon < 1.06nu; \quad (7.1.78)$$

To simplify the result of an error analysis we will often make use of the the following convenient notation

$$\bar{\gamma}_k = \frac{cku}{1 - cku/2}, \quad (7.1.79)$$

where c denotes a small integer constant.

Consider the inner product $x^T y$, where $x, y \in \mathbf{R}^m$. Since the errors depends on the order of evaluation we assume it is computed from left to right. Then we have

$$fl(x^T y) = x_1 y_1 (1 + \delta_1) + x_2 y_2 (1 + \delta_2) + \cdots + x_n y_n (1 + \delta_n)$$

where

$$|\delta_1| < \gamma_n, \quad |\delta_r| < \gamma_{n+2-i}, \quad i = 2 : n.$$

It follows that for any order of evaluation we have the backward error bounds

$$fl(x^T y) = (x + \Delta x)^T y = x^T(y + \Delta y), \quad |\Delta x| \leq \gamma_n |x|, \quad |\Delta y| \leq \gamma_n |y|. \quad (7.1.80)$$

The corresponding forward error bound becomes

$$|fl(x^T y) - x^T y| < \sum_{i=1}^n \gamma_{n+2-i} |x_i| |y_i| < \gamma_n |x^T| |y|, \quad (7.1.81)$$

where $|x|$, $|y|$ denote vectors with elements $|x_i|$, $|y_i|$. This bound is independent of the summation order and is valid also for floating-point computation with no guard digit rounding.

For the outer product xy^T of two vectors $x, y \in \mathbf{R}^n$ it holds that $fl(x_i y_j) = x_i y_j (1 + \delta_{ij})$, $\delta_{ij} \leq u$, and so

$$|fl(xy^T) - xy^T| \leq u |xy^T|. \quad (7.1.82)$$

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 $fl(xy^T) = (x + \Delta x)(y + \Delta y)^T$. This shows that matrix multiplication in floating-point arithmetic is not backward stable!

From the error analysis of inner products error bounds for matrix-vector and matrix-matrix multiplications can easily be obtained. Let $A \in \mathbf{R}^{m \times n}$, $x \in \mathbf{R}^n$, and $y = Ax$. Then $y_i = a_i^T x$, where a_i^T is the i th row of A . From (7.1.80) 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

$$fl(Ax) = (A + \Delta A)x, \quad |\Delta A| \leq \gamma_n |A|. \quad (7.1.83)$$

Now consider matrix multiplications, $C = AB$, where $A \in \mathbf{R}^{m \times n}$, $B \in \mathbf{R}^{n \times p}$. Then by (7.1.83), for the j th column c_j of C , we have

$$fl(AB_j) = (A + \Delta_j A)b_j, \quad |\Delta_j A| \leq \gamma_n |A|.$$

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 . However, we have the forward error bound

$$|fl(AB) - AB| < \gamma_n |A| |B|. \quad (7.1.84)$$

where the inequality is to be interpreted elementwise. Often we shall need bounds for some norm of the error matrix. From (7.1.84) it follows that

$$\|fl(AB) - AB\| < \gamma_n \|A\| \|B\|. \quad (7.1.85)$$

Hence, for the 1-norm, ∞ -norm and the Frobenius norm, we have

$$\|fl(AB) - AB\| < \gamma_n \|A\| \|B\|. \quad (7.1.86)$$

but unless A and B have nonnegative elements, we have for the 2-norm only the weaker bound

$$\|fl(AB) - AB\|_2 < n\gamma_n \|A\|_2 \|B\|_2. \quad (7.1.87)$$

In many matrix algorithms there repeatedly occurs 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 in Sec. 2.4.1 (cf. Problem 2.3.7) shows that if the term c is added last, then the computed \bar{y} satisfies

$$\bar{y}d(1 + \delta_k) = c - \sum_{i=1}^{k-1} a_i b_i(1 + \delta_i), \quad (7.1.88)$$

where

$$|\delta_1| \leq \gamma_{k-1}, \quad |\delta_i| \leq \gamma_{k+1-i}, \quad i = 2 : k-1, \quad |\delta_k| \leq \gamma_2. \quad (7.1.89)$$

and $\gamma_k = ku/(1 - ku)$ and u is the unit roundoff. 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

$$\left| \bar{y}d - c + \sum_{i=1}^{k-1} a_i b_i \right| \leq \gamma_k \left(|\bar{y}d| + \sum_{i=1}^{k-1} |a_i||b_i| \right), \quad (7.1.90)$$

and this inequality holds independent of the summation order.

7.1.9 Complex Arithmetic in Matrix Computations

Complex arithmetic can be reduced to real arithmetic. Let $x = a + ib$ and $y = c + id$ be two complex numbers, where $y \neq 0$. Then we have:

$$\begin{aligned} x \pm y &= a \pm c + i(b \pm d), \\ x \times y &= (ac - bd) + i(ad + bc), \\ x/y &= \frac{ac + bd}{c^2 + d^2} + i \frac{bc - ad}{c^2 + d^2}, \end{aligned} \quad (7.1.91)$$

Using the above formula complex addition (subtraction) needs two real additions. Multiplying two complex numbers requires four real multiplications and two real additions.

Lemma 7.1.22.

Assume that the standard model (7.1.76) for floating point arithmetic holds. Then, provided that no overflow or underflow occurs, no denormalized numbers are produced, the complex operations computed according to (7.1.91) satisfy

$$\begin{aligned} \text{fl}(x \pm y) &= (x \pm y)(1 + \delta), \quad |\delta| \leq u, \\ \text{fl}(x \times y) &= x \times y(1 + \delta), \quad |\delta| \leq \sqrt{5}u, \\ \text{fl}(x/y) &= x/y(1 + \delta), \quad |\delta| \leq \sqrt{2}\gamma_4, \end{aligned} \quad (7.1.92)$$

where δ is a complex number and $\gamma_n = nu/(1 - nu)$.

Proof. See Higham [328, Sec. 3.6]. The result for complex multiplication is due to Brent et al. [79]. \square

The square root of a complex number $u + iv = \sqrt{x + iy}$ is given by

$$u = \left(\frac{r+x}{2} \right)^{1/2}, \quad v = \left(\frac{r-x}{2} \right)^{1/2}, \quad r = \sqrt{x^2 + y^2}. \quad (7.1.93)$$

When $x > 0$ there will be cancellation when computing v , which can be severe if also $|x| \gg |y|$ (cf. Volume I, Section 2.3.4). To avoid this we note that

$$uv = \sqrt{r^2 - x^2}/2 = y/2,$$

so v can be computed from $v = y/(2u)$. When $x < 0$ we instead compute v from (7.1.93) and set $u = y/(2v)$.

Almost all published rounding error results are formulated for real arithmetic. This applies also to those given in this book. Since the bounds in Lemma 7.1.22 are of the same form as the standard model for real arithmetic, results for the real case are valid for complex arithmetic provided the constants in the bounds are increased appropriately.

Some programming languages, e.g., C, does 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 matrix

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + b \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \quad (7.1.94)$$

Thus, the real number 1 and the imaginary unit are represented by

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

respectively. The sum and product of two such matrices is again of this form. Multiplication of two matrices of this form is commutative

$$\begin{pmatrix} a_1 & -b_1 \\ b_1 & a_1 \end{pmatrix} \begin{pmatrix} a_2 & -b_2 \\ b_2 & a_2 \end{pmatrix} = \begin{pmatrix} a_1a_2 - b_1b_2 & -(a_1b_2 + b_1a_2) \\ a_1b_2 + b_1a_2 & a_1a_2 - b_1b_2 \end{pmatrix}. \quad (7.1.95)$$

and 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 (7.1.94) are therefore afield isomorphic to the field of complex numbers. Note that the complex scalar $u = \cos \theta + i \sin \theta = e^\theta$ on the unit circle corresponds to the orthogonal matrix

$$\tilde{u} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}.$$

This represents a counter-clockwise rotation of θ ; see Section 8.3.1.

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

$$\tilde{c}_{ij} = \begin{pmatrix} a_{ij} & -b_{ij} \\ b_{ij} & a_{ij} \end{pmatrix}. \quad (7.1.96)$$

As we have seen in Section 7.1.2 the operations of addition and multiplication can be performed on block matrices by the general rules of matrix treating the blocks as scalars. It follows that these operation as well as inversion can be performed by instead operating on the real representations of the complex matrices.

An obvious drawback of the outlined scheme is that the representation (7.1.96) is redundant and doubles the memory requirement, since the real and imaginary parts are stored twice. To some extent this drawback can easily be circumvented. Consider the multiplication of two complex numbers in (7.1.95). Clearly we only need to compute the first column in the product

$$\begin{pmatrix} a_1 & -b_1 \\ b_1 & a_1 \end{pmatrix} \begin{pmatrix} a_2 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 a_2 - b_1 b_2 \\ a_1 b_2 + b_1 a_2 \end{pmatrix},$$

the second block column can be constructed from the first. This extends to sums and products of matrices of this form. In a product only the first matrix C_1 needs to be expanded into full size. This trick can be useful when efficient subroutines for real matrix multiplication are available.

Example 7.1.6.

An alternative representation of complex matrices is as follows. 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

$$\begin{pmatrix} A & -B \\ B & A \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

Here we have associated a complex matrix $C = A + iB$ with the real matrix

$$C = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}, \quad (7.1.97)$$

This matrix is related to the previous matrix \tilde{C} by a permutation of rows and columns. Note that the structure of the product of two such matrices is again a block matrix with 2×2 blocks of this structure. In (7.1.96) the real part corresponds to the diagonal (symmetric) matrix $a_{ij}I_2$ the imaginary part corresponds to the skew symmetric part.

Review Questions

1.1 Define the concepts:

- (i) Real symmetric matrix.
- (ii) Real orthogonal matrix.
- (iii) Real skew-symmetric matrix.
- (iv) Triangular matrix.
- (v) Hessenberg matrix.

1.2 (a) What is the outer product of two column vectors x and y ?

(b) How is the Hadamard product of two matrices A and B defined?

1.3 How is a submatrix of $A = (a_{ij}) \in \mathbf{R}^{m \times n}$ constructed? What characterizes a *principal* submatrix?

1.4 Verify the formulas (7.1.20) for the inverse of a 2×2 block triangular matrices.

1.5 What is the Woodbury formula and what can it be used for?

1.6 (a) What is the singular value decomposition and the pseudoinverse of a matrix A ? What is the relationship of the pseudoinverse to least squares solutions of a linear system?

(b) How is the singular value decomposition related to the eigensystem for certain symmetric matrices?

1.7 (a) What is meant by a flop in this book? What was the old meaning of a flop in matrix computation?

(b) How many flops are required to multiply two matrices in $\mathbf{R}^{n \times n}$ if the standard definition is used. Is this scheme optimal?

(c) How many real flops are needed for complex addition and how many for a complex multiplication of matrices in $\mathbf{C}^{n \times n}$?

1.8 (a) Given a vector norm define the matrix subordinate norm.

(b) Give explicit expressions for the matrix p norms for $p = 1, 2, \infty$.

1.8 Define the p norm of a vector x . Show that

$$\frac{1}{n}\|x\|_1 \leq \frac{1}{\sqrt{n}}\|x\|_2 \leq \|x\|_\infty,$$

which are special cases of (7.1.49).

1.9 What is meant by the “unit roundoff” u ? What is (approximatively) the unit roundoff in IEEE single and double precision?

Problems

- 1.1** Show that if $R \in \mathbf{R}^{n \times n}$ is strictly upper triangular, then $R^n = 0$.
- 1.2** (a) Let $A \in \mathbf{R}^{m \times p}$, $B \in \mathbf{R}^{p \times n}$, 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 rank of a sum and product of two matrices satisfy
- $$\text{rank}(AB) \leq \min\{\text{rank}(A), \text{rank}(B)\}.$$
- 1.3** If A and B are square upper triangular matrices show that AB is upper triangular, and that A^{-1} is upper triangular if it exists. Is the same true for lower triangular matrices?
- 1.4** To solve a linear system $Ax = b$, $A \in \mathbf{R}^n$, by Cramer's rule requires the evaluation of $n+1$ determinants of order n (see (7.1.34)). 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!
- 1.5** (a) Show that the product of two Hermitian matrices is symmetric if and only if A and B commute, that is, $AB = BA$.
1.6 Let $A \in \mathbf{R}^{n \times n}$ be a given matrix. Show that if $Ax = y$ has *at least one* solution for any $y \in \mathbf{R}^n$, then it has *exactly one* solution for any $y \in \mathbf{R}^n$. (This is a useful formulation for showing uniqueness of approximation formulas.)
1.7 Let $A \in \mathbf{R}^{m \times n}$ have rows a_i^T , i.e., $A^T = (a_1, \dots, a_m)$. Show that

$$A^T A = \sum_{i=1}^m a_i a_i^T.$$

What is the corresponding expression for $A^T A$ if A is instead partitioned into columns?

- 1.8** Let $S \in \mathbf{R}^{n \times n}$ be skew-symmetric, $S^T = -S$.
(a) Show that $I - S$ is nonsingular and the matrix

$$Q = (I - S)^{-1}(I + S),$$

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 $0 \leq \theta < \pi$.

- 1.9** Show that for $x \in \mathbf{R}^n$,

$$\lim_{p \rightarrow \infty} \|x\|_p = \max_{1 \leq i \leq n} |x_i|.$$

1.10 A Hadamard matrix is a square matrix whose entries are either +1 or -1 and whose rows and columns are mutually orthogonal. Hadamard matrices are used in error-correcting codes and in statistics.

- (a) Show that from the definition a Hadamard matrix $H \in \mathbf{R}^{n \times n}$ has the properties

$$H^T H = H H^T = nI_n, \quad \det H = \pm\sqrt{n}.$$

- (b) Show that if H is a Hadamard matrix of order m , then the partitioned matrix

$$\begin{pmatrix} H & H \\ H & -H \end{pmatrix}$$

is a Hadamard matrix of order $2n$. Starting with $H = 1$, use this to construct Hadamard matrices of order $2, 4, 8, \dots$

- 1.11** 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 comparison of the operator norms $\|A\|_1$, $\|A\|_2$, and $\|A\|_\infty$.

- 1.12** Show that any vector norm is uniformly continuous by proving the inequality

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

- 1.13** Show that for any matrix norm there exists a consistent vector norm.

Hint: Take $\|x\| = \|xy^T\|$ for any vector $y \in \mathbf{R}^n$, $y \neq 0$.

- 1.14** Derive the formula for $\|A\|_\infty$ given in Theorem 7.1.13.

- 1.15** Make a table corresponding to Table 7.1.1 for the vector norms $p = 1, 2, \infty$.

- 1.16** Prove that for any subordinate matrix norm

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

- 1.17** Show that $\|A\|_2 = \|PAQ\|_2$ if P and Q are orthogonal matrices.

- 1.18** Use the result

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

valid for any matrix operator norm $\|\cdot\|$, where $\rho(A^T A)$ denotes the spectral radius of $A^T A$, to deduce the upper bound in (7.1.55).

- 1.19** Prove the expression (7.1.54) for the matrix norm subordinate to the vector ∞ -norm.

- 1.20** (a) Let T be a nonsingular matrix, and let $\|\cdot\|$ be a given vector norm. Show that the function $N(x) = \|Tx\|$ is a vector norm.

- (b) What is the matrix norm subordinate to $N(x)$?

- (c) If $N(x) = \max_i |k_i x_i|$, what is the subordinate matrix norm?

- 1.21** Consider an upper block triangular matrix

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

and suppose that R_{11}^{-1} and R_{22}^{-1} exists. Show that R^{-1} exists.

1.22 Suppose that $A \in \mathbf{R}^{n \times n}$ is nonsingular and $f, g \in \mathbf{R}^n$. Show that

$$\text{rank}(A - \omega^{-1} A f g^T A) = n - 1,$$

if and only if $\omega = g^T A f \neq 0$.

Note: This is a special case of the stepwise rank reduction procedure by Wedderburn. Chu, Funderlic, and Golub [109] discuss Wedderburn rank reduction from the point of view of solving linear systems of equations. They show that several standard matrix factorizations are special instances of the Wedderburn formula.

1.23 Use the Woodbury formula to prove the identity

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

1.24 (a) Let A^{-1} be known and let B be a matrix coinciding with A except in one row. Show that if B is nonsingular then B^{-1} can be computed by about $2n^2$ multiplications using the Sherman–Morrison formula (7.1.26).

(b) Use the Sherman–Morrison formula to compute B^{-1} if

$$A = \begin{pmatrix} 1 & 0 & -2 & 0 \\ -5 & 1 & 11 & -1 \\ 287 & -67 & -630 & 65 \\ -416 & 97 & 913 & -94 \end{pmatrix}, \quad A^{-1} = \begin{pmatrix} 13 & 14 & 6 & 4 \\ 8 & -1 & 13 & 9 \\ 6 & 7 & 3 & 2 \\ 9 & 5 & 16 & 11 \end{pmatrix},$$

and B equals A except that the element 913 has been changed to 913.01.

1.25 Show that the eigenvalues of the matrix

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$$

are equal to $a \pm ib$.

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

$$A_1 = (A + BA^{-1}B)^{-1}, \quad B_1 = A^{-1}BA_1 = A_1BA^{-1}. \quad (7.1.98)$$

Hint: Rewrite C as a real matrix of twice the size and use block elimination.

1.27 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

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

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

7.2 Gaussian Elimination

The closer one looks, the more subtle and

remarkable Gaussian elimination appears.

—Lloyd N. Trefethen, Three mysteries of Gaussian elimination. SIGNUM Newslet. (1985).

7.2.1 Solving Triangular Systems

The emphasis in this chapter will mostly be on algorithms for *real* linear systems, since (with the exception of complex Hermitian systems) these occur most commonly in applications. However, all algorithms given can readily be generalized to the complex case.

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

$$U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{pmatrix}.$$

If also $u_{ij} = 0$ when $i = j$ then U is **strictly** upper triangular. Similarly, a matrix L is **lower triangular** if $l_{ij} = 0$, $i < j$, and strictly lower triangular if $l_{ij} = 0$, $i \leq j$. Clearly the transpose of an upper triangular matrix is lower triangular and vice versa.

Triangular matrices have several nice properties. It is easy to verify that sums, products and inverses of square upper (lower) triangular matrices are again triangular matrices of the same type. 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 the diagonal elements in U^{-1} are the inverse of the diagonal elements in U .

Triangular linear systems are easy to solve. In an upper triangular linear system $Ux = b$, the unknowns can be computed recursively from

$$x_n = b_n/u_{nn} \quad x_i = \left(b_i - \sum_{k=i+1}^n u_{ik}x_k \right)/u_{ii}, \quad i = n-1 : 1. \quad (7.2.1)$$

Since the unknowns are solved for in *backward* order, this is called **back substitution**.

Similarly, a lower triangular matrix has form

$$L = \begin{pmatrix} \ell_{11} & 0 & \dots & 0 \\ \ell_{21} & \ell_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \dots & \ell_{nn} \end{pmatrix}.$$

The solution of a lower triangular linear system $Ly = c$, can be computed by

forward substitution.

$$y_1 = c_1/u_{11} \quad y_i = \left(c_i - \sum_{k=1}^{i-1} \ell_{ik} y_k \right) / \ell_{ii}, \quad i = 2 : n. \quad (7.2.2)$$

Solving a triangular system requires n^2 flops.

The back substitution algorithm (7.2.1) is implemented in the following MATLAB function.

Algorithm 7.1. *Back Substitution.*

Given an upper triangular matrix $U \in \mathbf{R}^{n \times n}$ and a vector $b \in \mathbf{R}^n$, the following algorithm computes $x \in \mathbf{R}^n$ such that $Ux = b$:

```

for  $i = n : -1 : 1$ 
   $s := 0;$ 
  for  $k = i + 1 : n$ 
     $s := s + u_{ik} b_k;$ 
  end
   $b_i := (b_i - s) / u_{ii};$ 
end

```

Note that in order to minimize round-off errors b_i is added *last* to the sum; compare the error bound (2.4.3).

A lower triangular system $Ly = b$ is solved by forward substitution

$$l_{kk} y_k = b_k - \sum_{i=1}^{k-1} l_{ki} y_i, \quad k = 1 : n.$$

If we let \bar{y} denote the computed solution, then using (7.1.88) –(7.1.89) it is straightforward to derive a bound for the backward error in solving a triangular system of equations.

Theorem 7.2.1. *If the lower triangular system $Ly = b$, $L \in \mathbf{R}^{n \times n}$ is solved by substitution with the summation order outlined above, then the computed solution \bar{y} satisfies*

$$(L + \Delta L)\bar{y} = b, \quad |\Delta l_{ki}| \leq \begin{cases} \gamma_2 |l_{ki}|, & i = k \\ \gamma_{k+1-i} |l_{ki}|, & i = 1 : k-1 \end{cases}, \quad k = 1 : n. \quad (7.2.3)$$

Hence, $|\Delta L| \leq \gamma_n |L|$ and this inequality holds for any summation order.

An analogue 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.

Example 7.2.1.

Algorithm 7.2.1 is the inner product version for solving a triangular system by back substitution in which the elements in U are accessed in rowwise order. By changing the order of the two loops above a vector sum version of the algorithm is obtained, where the elements in U are accessed in columnwise order.

```

for  $k = n : -1 : 1$ 
     $b_k := b_k / u_{kk};$ 
    for  $i = k - 1 : -1 : 1$ 
         $b_i := b_i - u_{ik}b_k;$ 
    end
end

```

Here the elements in U are accessed columnwise instead of rowwise as in the previous algorithm. Such differences can influence the efficiency when implementing matrix algorithms. Which version to be preferred depends on several factors. In general the orientation of the algorithm should agree with the storage scheme used by in the implementation. In FORTRAN, which stores arrays columnwise, the second is better. But in C, which stores arrays rowwise, the first version should be preferred. for further comments on implementing algorithms; see Section 7.6.

7.2.2 Gaussian Elimination

Consider a linear system $Ax = b$, where the matrix $A \in \mathbf{R}^{m \times n}$, and vector $b \in \mathbf{R}^m$ are given and the vector $x \in \mathbf{R}^n$ is to be determined, i.e.,

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}. \quad (7.2.4)$$

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.
- Interchange two equations.

These correspond in an obvious way to row operations on the augmented matrix (A, b) . It is also possible 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 **Gaussian elimination**¹⁴ is to use such elementary operations in a systematic way to eliminate the unknowns in the system $Ax = b$, so that

¹⁴Named after Carl Friedrich Gauss (1777–1855), but known already in China as early as the first century BC.

at the end an equivalent upper triangular system is produced. This is then solved by back substitution. If $a_{11} \neq 0$, then in the first step we eliminate x_1 from the last $(n - 1)$ equations by subtracting the multiple

$$l_{i1} = a_{i1}/a_{11}, \quad i = 2 : n,$$

of the first equation from the i th equation. This produces a reduced system of $(n - 1)$ equations in the $(n - 1)$ unknowns x_2, \dots, x_n , where the new coefficients are given by

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

If $a_{22}^{(2)} \neq 0$, we can next in a similar way eliminate x_2 from the last $(n - 2)$ of these equations. Similarly, in step k of Gaussian elimination the current elements in A and b are transformed according to

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)}, & b_i^{(k+1)} &= b_i^{(k)} - l_{ik}b_1^{(k)}, \\ i &= k + 1 : n, & j &= k + 1 : m. \end{aligned} \quad (7.2.5)$$

where the multipliers are

$$l_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}, \quad i = k + 1 : n. \quad (7.2.6)$$

The elimination stops when we run out of rows or columns. Then the matrix A has been reduced to the form

$$A^{(k)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} \\ \ddots & \vdots & & \vdots & & \vdots \\ a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & & \vdots \\ a_{mk}^{(k)} & \cdots & a_{mn}^{(k)} \end{pmatrix}, \quad b^{(k)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_k^{(k)} \\ \vdots \\ b_m^{(k)} \end{pmatrix}, \quad (7.2.7)$$

where we have put $A^{(1)} = A$, $b^{(1)} = b$. The diagonal elements

$$a_{11}, \quad a_{22}^{(2)}, \quad a_{33}^{(3)}, \dots,$$

which appear during the elimination are called **pivotal elements**.

Let A_k denote the k th leading principal submatrix of A . Since the determinant of a matrix does not change under row operations the determinant of A_k equals the product of the diagonal elements then by (7.2.8)

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

For a square system, i.e. $m = n$, this implies that all pivotal elements $a_{ii}^{(i)}$, $i = 1 : n$, in Gaussian elimination are nonzero if and only if $m = n$ and $\det(A_k) \neq 0$, $k = 1 : n$.

In this case the elimination can continue until after $(n - 1)$ steps, when we are left with the single equation

$$a_{nn}^{(n)}x_n = b_n^{(n)} \quad (a_{nn}^{(n)} \neq 0).$$

We have now obtained an upper triangular system $A^{(n)}x = b^{(n)}$, which can be solved recursively by back substitution (7.2.1). The determinant of A is given by

$$\det(A) = a_{11}^{(1)}a_{22}^{(2)} \cdots a_{nn}^{(n)}. \quad (7.2.8)$$

Since this expression can easily overflow or underflow some scaling may be needed before this is used.

If in Gaussian elimination a zero pivotal element is encountered, i.e. $a_{kk}^{(k)} = 0$ for some $k \leq n$ then we cannot proceed. If A is square and nonsingular, then in particular its first k columns are linearly independent. This must also be true for the first k columns of the reduced matrix. Hence, some element $a_{ik}^{(k)}$, $i \geq k$ must be nonzero, say $a_{pk}^{(k)} \neq 0$. By interchanging rows k and p this element can be taken as pivot and it is possible to proceed with the elimination. Note that when rows are interchanged in A the same interchanges must be made in the elements of the vector b . Note that the determinant formula (7.2.8) must be modified to

$$\det(A) = (-1)^s a_{11}^{(1)}a_{22}^{(2)} \cdots a_{nn}^{(n)}, \quad (7.2.9)$$

where s denotes the total number of row and columns interchanges performed.

In the general case, suppose that $a_{ik}^{(k)} = 0$. If the entire submatrix $a_{ij}^{(k)}$, $i, j = k : n$, is zero, then $\text{rank}(A) = k$ and we stop. Otherwise there is a nonzero element, say $a_{pq}^{(k)} \neq 0$, which can be brought into pivoting 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 .) Proceeding in this way any matrix A can always be reduced in $r = \text{rank}(A)$ steps to upper **trapezoidal form**,

Theorem 7.2.2.

Let $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, and consider the linear system $Ax = b$ of m equations in n unknowns $x \in \mathbf{R}^n$. By carrying out $r = \text{rank}(A)$ steps of Gaussian elimination with row and column permutations on A and b the system can be transformed into an equivalent system $A^{(r)}\hat{x} = b^{(r)}$, where

$$A^{(r)} = \left(\begin{array}{ccc|ccc} a_{11}^{(1)} & \cdots & a_{1r}^{(1)} & a_{1,r+1}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & \ddots & \vdots & \vdots & \ddots & \vdots \\ \vdots & & a_{rr}^{(r)} & a_{r,r+1}^{(r)} & \cdots & a_{rn}^{(r)} \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{array} \right), \quad b^{(r)} = \left(\begin{array}{c} b_1^{(1)} \\ \vdots \\ \frac{b_r^{(r)}}{b_{r+1}^{(r+1)}} \\ \vdots \\ b_m^{(r+1)} \end{array} \right) \quad (7.2.10)$$

Here $a_{kk}^{(k)} \neq 0$, $k = 1 : r$ and the two rectangular blocks of zeros in $A^{(r)}$ have dimensions $(m - r) \times r$ and $(m - r) \times (n - r)$. The reduced form (7.2.10) gives $\text{rank}(A) = r$. Further, the system $Ax = b$ is consistent if and only if $b_k^{(r+1)} = 0$, $k = r + 1 : m$.

After the augmented matrix $(A \quad b)$ has been reduced to the upper trapezoidal form (7.2.10) it becomes apparent that there are several different possibilities. The linear system $Ax = b$, where $A \in \mathbf{R}^{m \times n}$, is **consistent** if and only if $b \in \mathcal{R}(A)$, or equivalently $\text{rank}(A, b) = \text{rank}(A)$. This is the case if and only if $b_i^{(r+1)} = 0$, $i = r + 1 : m$. Clearly, a consistent linear system always has *at least one solution* x .

1. If $r = n = m$ then the system is nonsingular and there is a unique solution x for any right-hand side b .
2. If $r < n$ and the system is consistent then it has infinitely many solutions. Arbitrary values can be assigned to the last $n - r$ components of the (possibly permuted) solution vector x . Then the first r components are uniquely defined. Such a system is said to be **underdetermined**, and has no solution. If $\text{rank}(A) = m$ then $\mathcal{R}(A)$ equals \mathbf{R}^m and the system is consistent for all $b \in \mathbf{R}^m$.
3. If $r \leq n < m$ then the system is only consistent for special right hand sides. In general, it has no solution and we have to be content with a solution such that the residual vector $r = b - Ax$ is small in some sense. Such a system is said to be **overdetermined**.

If exact arithmetic is used the reduced trapezoidal form (7.2.10) yields the rank of the matrix A and answers the question whether the given system is consistent or not. In floating-point calculations it may be difficult to decide if a pivot element, or an element in the transformed right hand side, should be considered as zero or not. For example a zero pivot in exact arithmetic will almost invariably be polluted by roundoff errors into a nonzero number. What tolerance to use in deciding when a pivot should be taken to be numerically zero will depend on the context.

Underdetermined and overdetermined systems arise quite frequently. If there are more parameters than needed to span the right hand side the system is underdetermined. In this case additional information is needed in order to decide which solution to pick. On the other hand, overdetermined systems arise when there is more data than needed to determine the solution. In this case the system is inconsistent and has no solution. Underdetermined and overdetermined systems are best treated using *orthogonal* transformations, for example, the SVD, rather than Gaussian elimination; see Chapter 8.

The following MATLAB function reduces a nonsingular linear system $Ax = b$ to the upper triangular form $Ux = c$ by Gaussian Elimination (7.2.6)–(7.2.5). It is assumed that all pivot elements $a_{kk}^{(k)}$, $k = 1 : n$, are nonzero.

Algorithm 7.2. Gaussian Elimination.

Given a matrix $A \in \mathbf{R}^{n \times n}$ and a vector $b \in \mathbf{R}^n$, the following algorithm computes by Gaussian elimination a reduced upper triangular system $Ux = c$. At the end the matrix U is stored in the upper triangular part of A and c has overwritten b .

```

for  $k = 1 : n - 1$ 
  for  $i = k + 1 : n$ 
     $l_{ik} := a_{ik}/a_{kk};$ 
    for  $j = k + 1 : n$ 
       $a_{ij} := a_{ij} - l_{ik}a_{kj};$ 
    end
     $b_i := b_i - l_{ik}b_k;$ 
  end
end
```

Note that when l_{ik} is computed the element $a_{ik}^{(k)}$ 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. Then the operations on the right-hand side b can be deferred to a later stage. This observation is important in that it shows that *when solving a sequence of linear systems*

$$Ax_i = b_i, \quad i = 1 : p,$$

with the same matrix A but different right hand sides, the operations on A only have to be carried out once.

From Algorithm 7.2 it follows that $(n-k)$ divisions and $(n-k)^2$ multiplications and additions are used in step k to transform the elements of A . A further $(n-k)$ multiplications and additions are used to transform the elements of b . Summing over k and neglecting low order terms we find that Gaussian elimination requires

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

flops and an additional $\sum_{k=1}^{n-1} 2(n-k) \approx n^2$ flops for each right hand side. Hence, except for very small values of n , *the reduction of A to triangular form dominates the work*. This conclusion is not valid for banded and or sparse systems; see Sections 7.4 and 7.7, respectively.

According to (7.2.5) in step k the elements $a_{ij}^{(k)} \quad i, j = k + 1 : n$ are modified with the rank one matrix

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

The efficiency can be improved if this observation is incorporated. For example, in MATLAB the two innermost loops should be written

```

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);

```

The three nested loops in Gaussian elimination can be reordered in $3 \cdot 2 \cdot 1 = 6$ ways. Each of those versions perform the same basic operation

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

but in 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 7.2 (see Problem 7.2.3).

7.2.3 LU Factorization

We now show another interpretation of Gaussian elimination. For notational convenience we assume that $m = n$ and that Gaussian elimination can be carried out without pivoting. Then Gaussian elimination can be interpreted as computing the factorization $A = LU$ of the matrix A into the product of a unit lower triangular matrix L and an upper triangular matrix U .

Depending on whether the element a_{ij} lies on or above or below the principal diagonal we have

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

Thus, in Gaussian elimination the elements a_{ij} , $1 \leq i, j \leq n$, are transformed according to

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}, \quad k = 1 : p, \quad p = \min(i-1, j). \quad (7.2.11)$$

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

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

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

$$a_{ij} = \sum_{k=1}^r l_{ik} u_{kj}, \quad u_{kj} = a_{kj}^{(k)}, \quad r = \min(i, j). \quad (7.2.12)$$

However, these equations are equivalent to the matrix equation $A = LU$, where $L = (l_{ik})$ and $U = (u_{kj})$ are lower and upper triangular matrices, respectively. Hence, Gaussian elimination computes a factorization of A into a product of a lower and an upper triangular matrix, the **LU factorization** of A . Note that since the unit diagonal elements in L need not be stored, it is possible to store the L and U factors in an array of the same dimensions as A .

It was shown in Section 7.2.2 that if A is nonsingular, then Gaussian elimination can always be carried through provided row interchanges are allowed. Also, such row interchanges are in general needed to ensure the numerical stability of Gaussian elimination. We now consider how the *LU* factorization has to be modified when such interchanges are incorporated.

Row interchanges and row permutations can be expressed as pre-multiplication with a transposition matrix

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

which is equal to the identity matrix except that columns i and j have been interchanged; cf. Section 7.1.3. If a matrix A is premultiplied by I_{ij} this results in the interchange of *rows* i and j . Similarly, post-multiplication results in the interchange of *columns* i and j . $I_{ij}^T = I_{ij}$.

Assume that in the k th step, $k = 1 : n - 1$, we select the pivot element from row p_k , and interchange the rows k and p_k . Notice that in these row interchanges also previously computed multipliers l_{ij} must take part. 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 Gaussian elimination on PA *without any interchanges*. This means that Gaussian elimination with row interchanges computes the *LU* factors of the matrix PA . We now summarize the results and prove the uniqueness of the *LU* factorization:

Theorem 7.2.3. The *LU* factorization

Let $A \in \mathbf{R}^{n \times n}$ be a given nonsingular matrix. Then there is a permutation matrix P such that Gaussian elimination on the matrix $\tilde{A} = PA$ can be carried out without pivoting giving the factorization

$$PA = LU, \quad (7.2.13)$$

where $L = (l_{ij})$ is a unit lower triangular matrix and $U = (u_{ij})$ an upper triangular matrix. The elements in L and U are given by

$$u_{ij} = \tilde{a}_{ij}^{(i)}, \quad 1 \leq i \leq j \leq n,$$

and

$$l_{ij} = \tilde{l}_{ij}, \quad l_{ii} = 1, \quad 1 \leq j < i \leq n,$$

where \tilde{l}_{ij} are the multipliers occurring in the reduction of $\tilde{A} = PA$. For a fixed permutation matrix P , this factorization is uniquely determined.

Proof. We prove the uniqueness. Suppose we have two factorizations

$$PA = L_1 U_1 = L_2 U_2.$$

Since PA is nonsingular so are the factors, and it follows that $L_2^{-1}L_1 = U_2U_1^{-1}$. 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 be the identity matrix. Hence, $L_2 = L_1$, and $U_2 = U_1$. \square

Writing $PAx = LUx = L(Ux) = Pb$ it follows that if the LU factorization of PA is known, then the solution x can be computed by solving the two triangular systems

$$Ly = Pb, \quad Ux = y, \quad (7.2.14)$$

which involves about $2n^2$ flops. We remark that sometimes it is advantageous to write the factorization (7.2.13) in the form

$$PA = LDU,$$

where both L and U are *unit triangular* and D is diagonal

Although the LU factorization is just a different interpretation of Gaussian elimination 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 $Ux = y$.

The LU factorization makes it clear that Gaussian elimination is symmetric with respect to rows and columns. If $A = LU$ then $A^T = U^T L^T$ is an LU factorization of A^T with L^T unit upper triangular. As an example of the use of the factorization, consider the solution of the transposed system $A^T x = c$. Since $P^T P = I$, and

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

we have that $A^T P^T Px = U^T (L^T Px) = c$. It follows that $\tilde{x} = Px$ can be computed by solving the two triangular systems

$$U^T d = c, \quad L^T \tilde{x} = d. \quad (7.2.15)$$

We then obtain $x = P^{-1} \tilde{x}$ by applying the interchanges $k \leftrightarrow p_k$, in reverse order $k = n - 1 : 1$ to \tilde{x} . Note that it is not at all trivial to derive this algorithm from the presentation of Gaussian elimination in the previous section!

In the general case when $A \in \mathbf{R}^{m \times n}$ of $\text{rank}(A) = r \leq \min\{m, n\}$, it can be shown that matrix $P_r A P_c \in \mathbf{R}^{m \times n}$ can be factored into a product of a unit lower **trapezoidal matrix** $L \in \mathbf{R}^{m \times r}$ and an upper trapezoidal matrix $U \in \mathbf{R}^{r \times n}$. 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

$$P_r A P_c = LU = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} (U_{11} \quad U_{12}), \quad (7.2.16)$$

where the matrices L_{11} and U_{11} are triangular and nonsingular. Note that the block L_{21} is empty if the matrix A has full row rank, i.e. $r = m$; the block U_{12} is empty if the matrix A has full column rank, i.e. $r = n$.

To solve the system

$$P_r A P_c (P_c^T x) = LU \tilde{x} = P_r b = \tilde{b}, \quad x = P_c \tilde{x},$$

using this factorization we set $y = Ux$ and consider

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

This uniquely determines y as the solution to $L_{11}y = \tilde{b}_1$. Hence, *the system is consistent if and only if $L_{21}y = \tilde{b}_2$* . Further, we have $U\tilde{x} = y$, or

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

For an arbitrary \tilde{x}_2 this system uniquely determines \tilde{x}_1 as the solution to the triangular system

$$U_{11}\tilde{x}_1 = y - U_{12}\tilde{x}_2.$$

Thus, if consistent the system has a unique solution only if A has full column rank.

The reduction of a matrix to triangular form by Gaussian elimination can be expressed entirely in matrix notations using **elementary elimination matrices**. This way of looking at Gaussian elimination, first systematically exploited by J. H. Wilkinson¹⁵, has the advantage that it suggests ways of deriving other matrix factorizations.

Elementary elimination matrices are lower triangular matrices of the form

$$L_j = I + l_j e_j^T = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & l_{j+1,j} & 1 & \\ & & \vdots & & \ddots \\ & & l_{n,j} & & 1 \end{pmatrix}, \quad (7.2.17)$$

¹⁵James 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.

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

$$L_j x = (I + l_j e_j^T) x = x + l_j x_j = \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ x_{j+1} + l_{j+1,j} x_j \\ \vdots \\ x_n + l_{n,j} x_j \end{pmatrix},$$

i.e., to the last $n - j$ components of x are *added* multiples of the component x_j . 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 we find that

$$L_j^{-1} = I - l_j e_j^T.$$

The primary computational significance of elementary elimination matrices is that they can be used to introduce zero components in a column vector x . Assuming that $e_k^T x = x_k \neq 0$, there is a unique elementary elimination matrix $L_k^{-1} = I - l_k e_k^T$ such that

$$L_k^{-1}(x_1, \dots, x_k, x_{k+1}, \dots, x_n)^T = (x_1, \dots, x_k, 0, \dots, 0)^T.$$

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

$$l_k = (0, \dots, 0, x_{k+1}/x_k, \dots, x_n/x_k)^T.$$

The product of two elementary elimination matrices $L_j L_k$ is a lower triangular matrix which differs from the unit 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:

$$L_j L_k = I + l_j e_j^T + l_k e_k^T, \quad j \leq k. \quad (7.2.18)$$

Note that no products of the elements l_{ij} occur! However, 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 Gaussian elimination with partial pivoting can be accomplished by premultiplication of A by a sequence of elementary elimination matrices combined with transposition matrices to express the interchange of rows. For simplicity, we first consider the case when $\text{rank}(A) = m = n$. In the first step assume that $a_{p_1,1} \neq 0$ is the pivot element. Interchange rows 1 and p_1 in A by premultiplication of A by a transposition matrix,

$$\tilde{A} = P_1 A, \quad P_1 = I_{1,p_1}.$$

If we next premultiply \tilde{A} by the elementary elimination matrix

$$L_1^{-1} = I - l_1 e_1^T, \quad l_{i1} = \tilde{a}_{i1}/\tilde{a}_{11}, \quad i = 2 : n,$$

this will zero out the elements under the main diagonal in the first column, i.e.

$$A^{(2)} e_1 = L_1^{-1} P_1 A e_1 = \tilde{a}_{11} e_1.$$

All remaining elimination steps are similar to this first one. The second step is achieved by forming $\tilde{A}^{(2)} = P_2 A^{(2)}$ and

$$A^{(3)} = L_2^{-1} P_2 A^{(2)} = L_2^{-1} P_2 L_1^{-1} P_1 A.$$

Here $P_2 = I_{2,p_2}$, where $a_{p_2,2}^{(2)}$ is the pivot element from the second column and $L_2^{-1} = I - l_2 e_2^T$ is an elementary elimination matrix with nontrivial elements equal to $l_{i2} = \tilde{a}_{i2}^{(2)}/\tilde{a}_{22}^{(2)}$, $i = 3 : n$. Continuing, we have after $n - 1$ steps reduced A to upper triangular form

$$U = L_{n-1}^{-1} P_{n-1} \cdots L_2^{-1} P_2 L_1^{-1} P_1 A. \quad (7.2.19)$$

Since $P_2^2 = I$, we have after the first two steps

$$A^{(3)} = L_2^{-1} \tilde{L}_1^{-1} P_2 P_1 A$$

where

$$\tilde{L}_1^{-1} = P_2 L_1^{-1} P_2 = I - (P_2 l_1)(e_1^T P_2) = I - \tilde{l}_1 e_1^T.$$

Hence, \tilde{L}_1^{-1} is an elementary elimination matrix of the same type as L_1^{-1} , except that two elements in l_1 have been interchanged. Premultiplying by $\tilde{L}_1 L_2$, gives

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

where the two elementary elimination matrices on the left hand side combine trivially. Proceeding in a similar way it can be shown that (7.2.19) implies

$$\tilde{L}_1 \tilde{L}_2 \cdots \tilde{L}_{n-1} U = P_{n-1} \cdots P_2 P_1 A,$$

where $\tilde{L}_{n-1} = L_{n-1}$ and

$$\tilde{L}_j = I + \tilde{l}_j e_j^T, \quad \tilde{l}_j = P_{n-1} \cdots P_{j+1} l_j, \quad j = 1 : n - 2.$$

Using the result in (7.2.18), the elimination matrices can trivially be multiplied together. It follows that

$$PA = LU, \quad P = P_{n-1} \cdots P_2 P_1,$$

where the elements in L are given by $l_{ij} = \tilde{l}_{ij}$, $l_{ii} = 1$, $1 \leq j < i \leq n$. This is the LU factorization of PA . Nothing new, except the notations, has been introduced. Needless to say, the transposition matrices and elimination matrices used here are, of course, are never explicitly stored in a computer implementation.

7.2.4 Pivoting and Stability

In Gaussian elimination row and column interchanges need to be performed in case a zero pivot was encountered. A basic rule of numerical computation says that if an algorithm breaks down when a zero element is encountered, then we can expect some form of instability and loss of precision also for nonzero but small elements. This is related to the fact that in floating-point computation the difference between a zero and nonzero number becomes fuzzy because of the effect of rounding errors.

Example 7.2.2. For $\epsilon \neq 1$ the system

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

is nonsingular and has the unique solution

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

Suppose $\epsilon = 10^{-6}$ is accepted as pivot in Gaussian elimination. 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. However, computing x_1 by back-substitution gives $10^{-6}x_1 = 1 - 1$, or $x_1 = 0$, which is completely wrong.

The simple example above illustrates that in general it is necessary to perform row (and/or column) interchanges *not only when a pivotal element is exactly zero, but also when it is small*. The two most common pivoting strategies are **partial pivoting** and **complete pivoting**. In partial pivoting the pivot is taken as the largest element in magnitude in the unreduced part of the k th column. In complete pivoting the pivot is taken as the largest element in magnitude in the whole unreduced part of the matrix.

Partial Pivoting. At the start of the k th stage interchange rows k and r , where r is the smallest integer for which

$$|a_{rk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|. \quad (7.2.20)$$

Complete Pivoting. At the start of the k th stage interchange rows k and r and columns k and s , where r and s are the smallest integers for which

$$|a_{rs}^{(k)}| = \max_{k \leq i, j \leq n} |a_{ij}^{(k)}|. \quad (7.2.21)$$

Complete pivoting requires $O(n^3)$ in total compared with only $O(n^2)$ for partial pivoting. Hence, complete pivoting involves a fairly high overhead since about as many arithmetic comparisons as floating-point operations has to be performed. Since practical experience shows that partial pivoting works well, this is the standard choice. Note, however, that when $\text{rank}(A) < n$ complete pivoting must be used.

Gaussian elimination can easily be modified to include partial pivoting. In the following MATLAB program the interchanges are recorded in the vector p , which is initialized to be $(1, 2, \dots, n)$.

Algorithm 7.3. Gaussian Elimination with Partial Pivoting.

GAUSSPP reduces a nonsingular linear system $Ax = b$ by Gaussian elimination with partial pivoting to the upper triangular form $Ux = c$.

```

function [U,c,p] = gausspp(A,b);
n = length(b); p = 1:n;
for k = 1:n-1
    % find element of maximum magnitude in k:th column
    [piv,q] = max(abs(A(k:n,k)));
    q = k-1+q;
    if q > k % switch rows k and q
        A([k q],:) = A([q k],:);
        b([k q]) = b([q k]);
        p([k q]) = p([q k]);
    end
    % elimination step
    for i = k+1:n
        A(i,k) = A(i,k)/A(k,k);
        for j = k+1:n
            A(i,j) = A(i,j) - A(i,k)*A(k,j);
        end
        b(i) = b(i) - A(i,k)*b(k);
    end
end
U = triu(A); c = b;

```

A major breakthrough in the understanding of Gaussian elimination came with the backward rounding error analysis of Wilkinson [609]. Using the standard model (7.1.76) for floating point arithmetic it can be shown that the computed triangular factors \bar{L} and \bar{U} of A obtained by Gaussian elimination *are the exact triangular factors of a perturbed matrix*

$$\bar{L}\bar{U} = A + E, \quad E = (e_{ij})$$

where, since e_{ij} is the sum of $\min(i-1, j)$ rounding errors

$$|e_{ij}| \leq 3u \min(i-1, j) \max_k |\bar{a}_{ij}^{(k)}|. \quad (7.2.22)$$

This result holds without any assumption about the size of the multipliers. *It 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* (see the remark on possible large multipliers for positive definite matrices; Section 7.4.3).

The growth of elements during the elimination is usually measured by the **growth ratio**.

Definition 7.2.4.

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

$$\rho_n = \max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}|. \quad (7.2.23)$$

It follows that $E = (e_{ij})$ can be bounded componentwise by

$$|E| \leq 3\rho_n u \max_{ij} |a_{ij}| F. \quad (7.2.24)$$

where F is the matrix with elements $f_{i,j} = \min\{i-1, j\}$. 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 (7.2.23), we ignore this difference here and in the following. Slightly refining the estimate

$$\|F\|_\infty \leq (1 + 2 + \dots + n) - 1 \leq \frac{1}{2}n(n+1) - 1$$

and using $\max_{ij} |a_{ij}| \leq \|A\|_\infty$, gives the normwise backward error bound:

Theorem 7.2.5.

Let \bar{L} and \bar{U} be the computed triangular factors of A , obtained by Gaussian elimination where floating-point arithmetic with unit roundoff u has been used, there is a matrix E such that

$$\bar{L}\bar{U} = A + E, \quad \|E\|_\infty \leq 1.5n^2\rho_n u \|A\|_\infty. \quad (7.2.25)$$

If pivoting is employed the computed multipliers satisfy the inequality

$$|l_{ik}| \leq 1, \quad i = k+1 : n,$$

Then it can be shown that an estimate similar to (7.2.25) holds with the constant 1 instead of 1.5. For both partial and complete pivoting it holds that

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

and the bound $\rho_n \leq 2^{n-1}$ follows by induction. For partial pivoting this bound is the best possible, but is attained for special matrices. In practice any substantial growth of elements is extremely uncommon with partial pivoting. Quoting Kahan [363, 1966]: “*Intolerable pivot-growth (with partial pivoting) is a phenomenon that happens only to numerical analysts who are looking for that phenomenon.*” Why large element growth rarely occurs in practice with partial pivoting is a subtle and still not fully understood phenomenon. Trefethen and Schreiber [577] show that for certain distributions of random matrices the average element growth with partial pivoting is close to $n^{2/3}$.

For complete pivoting a much better bound can be proved, and in practice the growth very seldom exceeds n ; see Section 7.5.2. A pivoting scheme that gives

a pivot of size between that of partial and complete pivoting is **rook pivoting**. In this scheme we pick a pivot element which is largest in magnitude in *both its column and its row*.

Rook Pivoting. At the start of the k th stage rows k and r and columns k and s are interchanged, where

$$|a_{rs}^{(k)}| = \max_{k \leq i \leq n} |a_{ij}^{(k)}| = \max_{k \leq j \leq n} |a_{ij}^{(k)}|. \quad (7.2.26)$$

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. The name derives from the fact that the pivot search resembles the moves of a rook in chess; see Figure 7.2.1.

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

Figure 7.2.1. Illustration of rook pivoting in a 5×5 matrix with positive integer entries as shown. The (2, 4) element 9 is chosen as pivot.

Rook pivoting involves at least twice as many comparisons as partial pivoting. In the worst case it can take $O(n^3)$ comparisons, i.e., the same order of magnitude as for complete pivoting. Numerical experience shows that the cost of rook pivoting usually equals a small multiple of the cost for partial pivoting. A pivoting related to rook pivoting is used in the solution of symmetric indefinite systems; see Section 7.3.4.

An error analysis of pairwise pivoting has been given by Sorensen [1985].

It is important to realize that the choice of pivots can be influenced by the scaling of equations and unknowns. For example, if the unknowns are physical quantities, then a different choice of units will correspond to a different scaling of the unknowns and the columns in A . Partial pivoting by row interchanges has the important property of being invariant under column scalings. On the other hand, partial pivoting by *column* interchanges is invariant under row scalings, but not under column scalings. In practice this turns out to be less satisfactory variant. Complete and rook pivoting are influenced both by row and column scalings. For more on how scaling of linear systems can influence the accuracy of computed solutions, see Section 7.5.5.

For certain important classes of matrices a bound independent of n can be given for the growth ratio in Gaussian elimination without pivoting or with partial pivoting. For these Gaussian elimination is backward stable.

- If A is real symmetric matrix $A = A^T$ and positive definite (i.e. $x^T Ax > 0$ for all $x \neq 0$) then $\rho_n(A) \leq 1$ with no pivoting (see Theorem 7.3.7).
- If A is row or column diagonally dominant then $\rho_n \leq 2$ with no pivoting.
- If A is Hessenberg then $\rho_n \leq n$ with partial pivoting.
- If A is tridiagonal then $\rho_n \leq 2$ with partial pivoting.

For the last two cases we refer to Section 7.4. We now consider the case when A is diagonally dominant.

Definition 7.2.6. *A matrix A is said to be **diagonally dominant by rows**, if*

$$\sum_{j \neq i} |a_{ij}| \leq |a_{ii}|, \quad i = 1 : n. \quad (7.2.27)$$

A is diagonally dominant by columns if A^T is diagonally dominant by rows.

Theorem 7.2.7.

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 [609, 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 Gaussian elimination without pivoting we have

$$a_{ij}^{(2)} = a_{ij} - l_{i1}a_{1j}, \quad l_{i1} = a_{i1}/a_{11}, \quad i, j \geq 2, \quad (7.2.28)$$

where

$$\sum_{i=2}^n |l_{i1}| \leq \sum_{i=2}^n |a_{i1}|/|a_{11}| \leq 1. \quad (7.2.29)$$

For $j = i$, using the definition and (7.2.29), 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.

Further, using (7.2.28) and (7.2.29), for $i \geq 2$,

$$\begin{aligned}\sum_{i=2}^n |a_{ij}^{(2)}| &\leq \sum_{i=2}^n (|a_{ij}| + |l_{i1}| |a_{1j}|) \leq \sum_{i=2}^n |a_{ij}| + |a_{1j}| \sum_{i=2}^n |l_{i1}| \\ &\leq \sum_{i=2}^n |a_{ij}| + |a_{1j}| = \sum_{i=1}^n |a_{ij}|.\end{aligned}$$

Hence, the sum of the moduli of the elements of any column of $A^{(k)}$ does not increase as k increases. Hence,

$$\max_{i,j,k} |a_{ij}^{(k)}| \leq \max_{i,k} \sum_{j=k}^n |a_{ij}^{(k)}| \leq \max_i \sum_{j=1}^n |a_{ij}| \leq 2 \max_i |a_{ii}| = 2 \max_{i,j} |a_{ij}|.$$

It follows that

$$\rho_n = \max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}| \leq 2.$$

The proof for matrices which are *row* diagonally dominant is similar. (Notice that Gaussian elimination with pivoting essentially treats rows and columns symmetrically!) \square

We conclude that for a row or column diagonally dominant matrix Gaussian elimination 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.

If (7.2.27) holds with strict inequality for all i , then A is said to be **strictly diagonally dominant** by rows. If A is strictly diagonally dominant, then it can be shown that all reduced matrices have the same property. In particular, all pivot elements must then be strictly positive and the nonsingularity of A follows. We mention a useful result for strictly diagonally dominant matrices.

Lemma 7.2.8.

Let A be strictly diagonally dominant by rows, and set

$$\alpha = \min_i \alpha_i, \quad \alpha_i := |a_{ii}| - \sum_{j \neq i} |a_{ij}| > 0, \quad i = 1 : n. \quad (7.2.30)$$

Then A is nonsingular, and $\|A^{-1}\|_\infty \leq \alpha^{-1}$.

Proof. By the definition of a subordinate norm (7.1.50) we have

$$\frac{1}{\|A^{-1}\|_\infty} = \inf_{y \neq 0} \frac{\|y\|_\infty}{\|A^{-1}y\|_\infty} = \inf_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \min_{\|x\|_\infty=1} \|Ax\|_\infty.$$

Assume that equality holds in (7.2.30) for $i = k$. Then

$$\begin{aligned}\frac{1}{\|A^{-1}\|_\infty} &= \min_{\|x\|_\infty=1} \max_i \left| \sum_j a_{ij} x_j \right| \geq \min_{\|x\|_\infty=1} \left| \sum_j a_{kj} x_j \right| \\ &\geq |a_{kk}| - \sum_{j,j \neq k} |a_{kj}| = \alpha.\end{aligned}$$

□

If A is strictly diagonally dominant by columns, then since $\|A\|_1 = \|A^T\|_\infty$ it holds that $\|A^{-1}\|_1 \leq \alpha^{-1}$. If A is strictly diagonally dominant in both rows and columns, then from $\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}$ it follows that $\|A^{-1}\|_2 \leq \alpha^{-1}$.

7.2.5 Computational 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 first 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 (7.2.12))

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

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 will 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

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj}, \quad j \geq k, \quad l_{ik} u_{kk} = a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk}, \quad i > k. \quad (7.2.31)$$

This algorithm is usually referred to as Doolittle's algorithm [172]. The more well-known Crout's algorithm [126] is similar except that instead the upper triangular matrix U is normalized to have a unit diagonal. The main work in Doolittle's algorithm is performed in the inner products of rows of L and columns in U .

```

for  $k = 1 : n$ 
  for  $j = k : n$ 
     $u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj};$ 
  end
  for  $i = k + 1 : n$ 
```

```


$$l_{ik} = \left( a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk} \right) / u_{kk};$$

end

$$l_{kk} = 1;$$

end

```

Since the LU factorization is unique this algorithm produces the same factors L and U as Gaussian elimination. In fact, successive partial sums in the equations (7.2.31) equal the elements $a_{ij}^{(k)}$, $j > k$, in Gaussian elimination. It follows that if each term in (7.2.31) is rounded separately, the compact algorithm is also *numerically equivalent* to Gaussian elimination. If the inner products can be accumulated in higher precision, then the compact algorithm is less affected by rounding errors.¹⁶

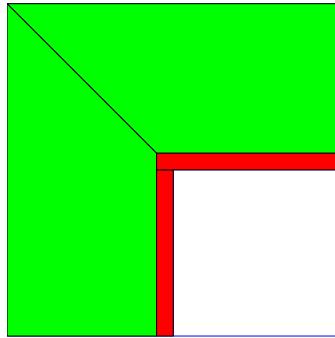


Figure 7.2.2. Computations in the k th step of Doolittle's method.

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.

Algorithm 7.4. Doolittle's Algorithm with Partial Pivoting.

```

% DOOL computes the LU factorization of a nonsingular
% matrix PA using partial pivoting.
function [L,U,p] = dool(A);
n = size(A,1); p = 1:n;
L = zeros(n,n); U = L;
for k = 1:n
    for i = k:n

```

¹⁶In the days of hand computations these algorithms had the advantage that they did away with the necessity in Gaussian elimination to write down $\approx n^3/3$ intermediate results—one for each multiplication.

```

L(i,k) = A(i,k) - L(i,1:k-1)*U(1:k-1,k);
end
[piv,q] = max(abs(L(k:n,k)));
q = k-1+q;
if q > k % switch rows k and q in L and A
    L([k q],1:k) = L([q k],1:k);
    A([k q],k+1:n) = A([q k],k+1:n);
    p([k q]) = p([q k]);
end
U(k,k) = L(k,k); L(k,k) = 1;
for j = k+1:n
    L(j,k) = L(j,k)/U(k,k);
    U(k,j) = A(k,j) - L(k,1:k-1)*U(1:k-1,j);
end
end

```

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 below,

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

Here the entries indicate in which step a certain element l_{ij} and u_{ij} is computed. The first example corresponds to the ordering in the algorithm given above. (Compare the comments after Algorithm 7.2.) Note that it is *not* easy to do complete pivoting with either of the last two variants.

Before the k th step, $k = 1 : n$, of the **bordering method** we have computed the LU-factorization $A_{11} = L_{11}U_{11}$ of the leading principal submatrix of order $k-1$ of A . To proceed we seek the LU-factorization of the bordered matrix

$$\begin{pmatrix} A_{11} & a_{1k} \\ a_{k1}^T & \alpha_{kk} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ l_{k1}^T & 1 \end{pmatrix} \begin{pmatrix} U_{11} & u_{1k} \\ 0 & u_{kk} \end{pmatrix}. \quad (7.2.32)$$

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

$$L_{11}u_{1k} = a_{1k}. \quad (7.2.33)$$

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

$$U_{11}^T l_{k1} = a_{k1}, \quad (7.2.34)$$

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

$$u_{kk} = \alpha_{kk} - l_{k1}^T u_{1k}. \quad (7.2.35)$$

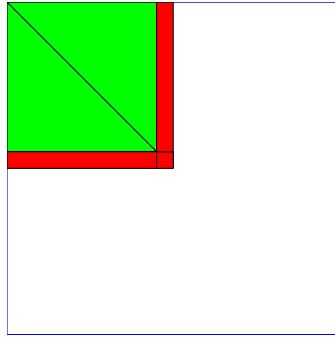


Figure 7.2.3. Computations in the k th step of the bordering method.

The main work in bordering method is done in solving the two triangular systems (7.2.33) and (7.2.34). A drawback is that this method cannot be combined with partial pivoting, since the Schur complement of A_{11} is not available.

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

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

denote the first k columns of the LU factorization 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 Figure 7.2.4 left)

$$L_{11}u_{1k} = a_{1k}, \quad l_{2k}u_{kk} + L_{21}u_{1k} = a_{2k}. \quad (7.2.36)$$

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

$$l_{2k}u_{kk} = a_{2k} - L_{21}u_{1k}. \quad (7.2.37)$$

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 (7.2.37) 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}^T and the unprocessed part of A .

In the column sweep method L and U are determined column by column. Alternatively L and U can be determined row by row. In step k of the **row sweep** method the k th row of A is processed. Let

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

denote the first k rows of the LU factorization of A , where that L_{11} , U_{11} and U_{12} have been computed. Equating elements in the k th row we find (see Figure 7.2.4

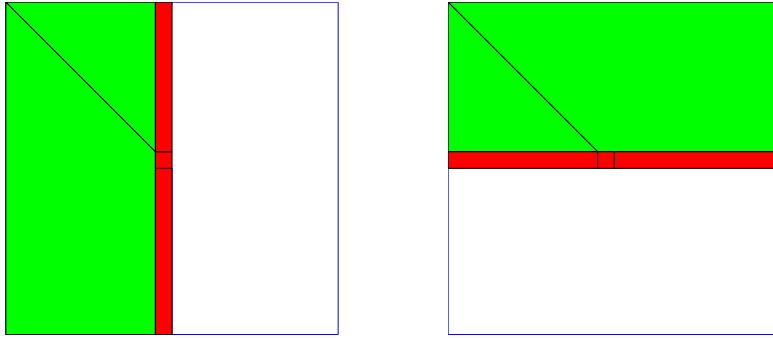


Figure 7.2.4. Computations in the k th step of the sweep methods. Left: The column sweep method. Right: The row sweep method.

right)

$$U_{11}^T l_{k1} = a_{k1}, \quad u_{2k}^T + l_{k1} U_{12} = a_{k2}^T. \quad (7.2.38)$$

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.

For a rectangular matrix $A \in \mathbf{R}^{m \times n}$, $m > n$, it is advantageous to process the matrix column by column. In the column sweep method we obtain after n steps we have $AP_c = LU$, where

$$L = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \in \mathbf{R}^{m \times n}, \quad (7.2.39)$$

is lower trapezoidal and $U \in \mathbf{R}^{n \times n}$ is square upper triangular (see Figure 7.2.4, left). Similarly, if $m < n$ and the matrix is processed row by row, we have after m steps an LU factorization with $L \in \mathbf{R}^{m \times m}$ square and lower triangular and

$$U = (U_{11} \quad U_{12}) \in \mathbf{R}^{m \times n}$$

upper trapezoidal.

7.2.6 Computing the Inverse Matrix

If the inverse matrix A^{-1} is known, then the solution of $Ax = b$ can be obtained through a matrix vector multiplication by $x = A^{-1}b$. This is theoretically satisfying, but in most practical computational problems it is unnecessary and inadvisable to compute A^{-1} . As succinctly expressed by Forsythe and Moler [215]: “*Almost anything you can do with A^{-1} can be done without it.*”

The work required to compute A^{-1} is about $2n^3$ flops, i.e., three times greater than for computing the LU factorization. (If A is a band matrix, then the difference can be much more spectacular; see Section 7.4.) 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 LU

factorization of A , however, that on some parallel computers matrix multiplication can be performed much faster than solving triangular systems.)

One advantage of computing the inverse matrix is that A^{-1} can be used to get a strictly reliable error estimate for a computed solution \bar{x} . A similar estimate is not directly available from the LU factorization. However, alternative ways to obtain error estimates are the use of a condition estimator (Section 7.5.1) or iterative refinement (Section 7.5.6).

Not only is the inverse matrix approach expensive, but if A is ill-conditioned, then the solution computed from $A^{-1}b$ usually is much less satisfactory than that computed from the LU factorization. Using LU factorization the relative precision of the residual vector of the computed solution will almost always be of the order of machine precision even when A is ill-conditioned.

Nevertheless, there are some applications where A^{-1} is required, e.g., in some methods for computing the matrix square root and the logarithm of a matrix; see Section 9.2.4. The inverse of a symmetric positive definite matrix is needed to obtain estimates of the covariances in regression analysis. However, often only certain elements of A^{-1} are needed and not the whole inverse matrix; see Section 8.2.1.

We first consider computing the inverse of a lower triangular matrix L . Setting $L^{-1} = Y = (y_1, \dots, y_n)$, we have $LY = I = (e_1, \dots, e_n)$. This shows that the columns of Y satisfy

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

These lower triangular systems can be solved by forward substitution. Since the vector e_j has $(j-1)$ leading zeros the first $(j-1)$ components in y_j are zero. Hence, L^{-1} is also a lower triangular matrix, and its elements can be computed recursively from

$$y_{jj} = 1/l_{jj}, \quad y_{ij} = -\left(\sum_{k=j}^{i-1} l_{ik} y_{kj}\right)/l_{ii}, \quad i = j+1 : n, \quad (7.2.40)$$

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

$$z_{jj} = 1/u_{jj}, \quad z_{ij} = -\left(\sum_{k=i+1}^j u_{ik} z_{kj}\right)/u_{ii}, \quad i = j-1 : -1 : 1. \quad (7.2.41)$$

If the columns are computed in the order $j = n : -1 : 1$, the Z can overwrite U in storage. The number of flops required to compute L^{-1} or U^{-1} is approximately equal to $n^3/3$. Variants of the above algorithm can be obtained by reordering the loop indices.

Now let $A^{-1} = X = (x_1, \dots, x_n)$ and assume that an LU factorization $A = LU$ has been computed. Then

$$Ax_j = L(Ux_j) = e_j, \quad j = 1 : n, \quad (7.2.42)$$

and the columns of A^{-1} are obtained by solving n linear systems, where the right hand sides equal the columns in the unit matrix. Setting (7.2.42) is equivalent to

$$Ux_j = y_j, \quad Ly_j = e_j, \quad j = 1 : n. \quad (7.2.43)$$

This method for inverting A requires $2n^3/3$ flops for inverting L and n^3 flops for solving the n upper triangular systems giving again a total of $2n^3$ flops.

A second method uses the relation

$$A^{-1} = (LU)^{-1} = U^{-1}L^{-1}. \quad (7.2.44)$$

Since the matrix multiplication $U^{-1}L^{-1}$ requires $2n^3/3$ flops (show this!) the total work to compute A^{-1} by the second method method (7.2.44) is also $2n^3$ flops. If we take advantage of that $y_{jj} = 1/l_{jj} = 1$, and carefully sequence the computations then L^{-1} , U^{-1} and finally A^{-1} can overwrite A so that no extra storage is needed.

There are many other variants of computing the inverse $X = A^{-1}$. From $XA = I$ we have

$$XLU = I \quad \text{or} \quad XL = U^{-1}.$$

In the MATLAB function `inv(A)`, U^{-1} is first computed by a column oriented algorithm. Then the system $XL = U^{-1}$ is solved for X . The stability properties of this and several other different matrix inversion algorithms are analyzed in [173, 1992]; see also Higham [328, Sec. 14.2].

In Gaussian elimination we use in the k th step the pivot row to eliminate elements *below* the main diagonal in column k . In **Gauss–Jordan elimination**¹⁷ the elements *above* the main diagonal are eliminated simultaneously. After $n - 1$ steps the matrix A has then been transformed into a *diagonal* matrix containing the nonzero pivot elements. Gauss–Jordan elimination was used in many early versions of linear programming and also for implementing stepwise regression in statistics.

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

$$M_j = \begin{pmatrix} 1 & & l_{1j} & & \\ & \ddots & \vdots & & \\ & & 1 & l_{j-1,j} & \\ & & & 1 & \\ & & & l_{j+1,j} & 1 \\ & & & \vdots & \ddots \\ & & l_{n,j} & & 1 \end{pmatrix}. \quad (7.2.45)$$

If partial pivoting is carried out we can write, cf. (7.2.19)

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

¹⁷Named after Wilhelm Jordan (1842–1899) was a German geodesist, who made surveys in Germany and Africa. He used this method to compute the covariance matrix in least squares problems.

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

$$A^{-1} = D^{-1} M_n M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1. \quad (7.2.46)$$

This expresses the inverse of A as a product of elimination and transposition matrices, and is called the **product form of the inverse**. The operation count for this elimination process is $\approx n^3$ flops, which is higher than for the LU factorization by Gaussian elimination. Gauss–Jordan elimination may still have advantages for some parallel implementations.

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

$$x = A^{-1}b = D^{-1} M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1 b. \quad (7.2.47)$$

This requires $2n^2$ flops. Note that no back substitution is needed!

The inverse A^{-1} can be obtained from the Gauss–Jordan factorization as follows. Using (7.2.47) where b is taken to be the columns of the unit matrix, compute

$$A^{-1} = D^{-1} M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1 (e_1, \dots, e_n).$$

If the computations are properly organized $2n^3$ flops are required as for the other methods described above. The method can be arranged so that the inverse emerges in the original array.

If row interchanges have been performed during the LU factorization, we have $PA = LU$, where $P = P_{n-1} \cdots P_2 P_1$ and P_k are transposition matrices. Then $A^{-1} = (LU)^{-1}P$. Hence, we obtain A^{-1} by performing the interchanges in *reverse order on the columns* of $(LU)^{-1}$.

The numerical properties of Gauss–Jordan elimination are not as good as for the methods based on LU factorization. It is also forward stable, i.e., will give about the same numerical accuracy in the computed solution \bar{x} as Gaussian elimination. However, as shown by Peters and Wilkinson [485], the residuals $b - A\bar{x}$ corresponding to the Gauss–Jordan solution \bar{x} can be a larger by a factor $\kappa(A)$ than those corresponding to the solution by LU factorization. Although the method is not backward stable in general it can be shown to be stable for diagonally dominant matrices (see Definition 7.2.6).

Review Questions

- 2.1** How many operations are needed (approximately) for
 - (a) The LU factorization of a square matrix?
 - (b) The solution of $Ax = b$, when the triangular factorization of A is known?
- 2.2** Show that if the k th diagonal entry of an upper triangular matrix is zero, then its first k columns are linearly dependent.
- 2.3** What is meant by partial and complete pivoting in Gaussian elimination? Mention two classes of matrices for which Gaussian elimination can be performed stably without any pivoting?

- 2.4** What is the *LU*-decomposition of an n by n matrix A , and how is it related to Gaussian elimination? Does it always exist? If not, give sufficient conditions for its existence.
- 2.5** How is the *LU*-decomposition used for solving a linear system? What are the advantages over using the inverse of A ? Give an approximate operation count for the solution of a dense linear system with p different right hand sides using the *LU*-decomposition.
- 2.6** Let B be a strictly lower or upper triangular matrix. Prove that the Neumann and Euler expansions for $(I - L)^{-1}$ are finite.

Problems

- 2.1** (a) Compute the *LU* factorization of A and $\det(A)$, where

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \end{pmatrix}.$$

- (b) Solve the linear system $Ax = b$, where $b = (2, 10, 44, 190)^T$.
- 2.2** (a) Show that $P = (e_n, \dots, e_2, e_1)$ is a permutation matrix and that $P = P^T = P^{-1}$, and that Px reverses the order of the elements in the vector x .
(b) Let the matrix A have an *LU* factorization. Show that there is a related factorization $PAP = UL$, where U is upper triangular and L lower triangular.
- 2.3** In Algorithm 7.2 for Gaussian elimination the elements in A are assessed in rowwise order in the innermost loop over j . If implemented in Fortran this algorithm may be inefficient since this language stores two-dimensional arrays by columns. Modify Algorithm 7.2 so that the innermost loop instead involves a fixed column index and a varying row index.
- 2.4** What does M_j^{-1} , where M_j is defined in (7.2.45), look like?
- 2.5** Compute the inverse matrix A^{-1} , where

$$A = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 4 & 1 & 2 \end{pmatrix},$$

- (a) By solving $AX = I$, using Gaussian elimination with partial pivoting.
(b) By *LU* factorization and using $A^{-1} = U^{-1}L^{-1}$.

7.3 Symmetric Matrices

7.3.1 Symmetric Positive Definite Matrices

Gaussian elimination can be adopted to several classes of matrices of special structure. As mentioned in Section 7.2.4, one case when Gaussian elimination can be

performed stably without any pivoting is when A is Hermitian or real symmetric and positive definite. Solving such systems is one of the most important problems in scientific computing.

Definition 7.3.1.

*A matrix $A \in \mathbf{C}^{n \times n}$ is called **Hermitian** if $A = A^H$, the conjugate transpose of A . If A is Hermitian, then the quadratic form $(x^H Ax)^H = x^H Ax$ is real and A is said to be **positive definite** if*

$$x^H Ax > 0 \quad \forall x \in \mathbf{C}^n, \quad x \neq 0, \quad (7.3.1)$$

*and **positive semidefinite** if $x^T Ax \geq 0$, for all $x \in \mathbf{R}^n$. Otherwise it is called **indefinite**.*

It is well known that all eigenvalues of an Hermitian matrix are real. An equivalent condition for an Hermitian matrix to be positive definite is that all its eigenvalues are positive $\lambda_k(A) > 0$, $k = 1 : n$. Since this condition can be difficult to verify the following sufficient condition is useful. A Hermitian matrix A , which has positive diagonal elements and is diagonally dominant, i.e.,

$$a_{ii} > \sum_{j \neq i} |a_{ij}|, \quad i = 1 : n,$$

is positive definite. (This follows since by Geršgorin's Theorem 10.2.1 all eigenvalues of A must be positive.)

Clearly a positive definite matrix is nonsingular, since if it were singular there should be a null vector $x \neq 0$ such that $Ax = 0$ and then $x^H Ax = 0$. Positive definite (semidefinite) matrices have the following important property:

Theorem 7.3.2. *Let $A \in \mathbf{C}^{n \times n}$ be positive definite (semidefinite) and let $X \in \mathbf{C}^{n \times p}$ have full column rank. Then $X^H AX$ is positive definite (semidefinite). In particular, any principal $p \times p$ submatrix*

$$\tilde{A} = \begin{pmatrix} a_{i_1 i_1} & \dots & a_{i_1 i_p} \\ \vdots & & \vdots \\ a_{i_p i_1} & \dots & a_{i_p i_p} \end{pmatrix} \in \mathbf{C}^{p \times p}, \quad 1 \leq p < n,$$

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

A Hermitian or symmetric matrix A of order n has only $\frac{1}{2}n(n+1)$ independent elements. If A is also positive definite then symmetry can be preserved in Gaussian

elimination and the number of operations and storage needed can be reduced by half. Indeed, Gauss derived his original algorithm for the symmetric positive definite systems coming from least squares problems (see Chapter 8). We consider below the special case when A is real and symmetric, but all results are easily generalized to the complex Hermitian case. (Complex symmetric matrices appear also in some practical problems. These are more difficult to handle.)

Lemma 7.3.3. *Let A be a real symmetric matrix. If Gaussian elimination can be carried through without pivoting, then the reduced matrices $A = A^{(1)}, A^{(2)}, \dots, A^{(n)}$ are all symmetric.*

Proof. Assume that $A^{(k)}$ is symmetric, for some k , where $1 \leq k < n$. Then (cf. Algorithm 7.2) after the k -th elimination step

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}a_{kj}^{(k)} = a_{ji}^{(k)} - \frac{a_{jk}^{(k)}}{a_{kk}^{(k)}}a_{ki}^{(k)} = a_{ji}^{(k+1)},$$

$k+1 \leq i, j \leq n$. This shows that $A^{(k+1)}$ is also a symmetric matrix, and the result follows by induction. \square

A more general result is the following. Let

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^H & A_{22} \end{pmatrix}$$

be a positive definite Hermitian, partitioned so that A_{11} is a square matrix. Then by Theorem 7.3.2 both A_{11} and A_{22} are Hermitian positive definite and therefore nonsingular. It follows that the Schur complement

$$S = A_{22} - A_{12}^H A_{11}^{-1} A_{12}$$

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.

Since all reduced matrices it follows that in Gaussian elimination without pivoting only the elements in $A^{(k)}$, $k = 2 : n$, on and below the main diagonal have to be computed. Since any diagonal element can be brought in pivotal position by a symmetric row and column interchange, the same conclusion holds if pivots are chosen arbitrarily along the diagonal.

Algorithm 7.5. *Outer Product LDL^T Factorization.*

Given the lower triangular part of a symmetric matrix A . The following algorithm computes (if it can be carried through) a *unit lower triangular* matrix L and a diagonal matrix D such that

$$A = LDL^T. \tag{7.3.2}$$

```

for  $k = 1 : n$ 
   $d_k := a_{kk}^{(k)}$ ;
  for  $i = k + 1 : n$ 
     $l_{ik} := a_{ik}^{(k)} / d_k$ ;
    for  $j = k + 1 : i$ 
       $a_{ij}^{(k+1)} := a_{ij}^{(k)} - d_k l_{ik} l_{jk}$ ;
    end
  end
end

```

In inner loop $d_k l_{jk}$ is substituted for $a_{kj}^{(k)}$. Only the lower triangular part of A is accessed.

This algorithm requires approximately $n^3/3$ flops, which is only 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 LDL^T factorization follows trivially from the uniqueness of the LU factorization.

Using the factorization $A = LDL^T$ the linear system $Ax = b$ decomposes into the two triangular systems

$$Ly = b, \quad L^T x = D^{-1}y. \quad (7.3.3)$$

The cost of solving these triangular systems is about $2n^2$ flops.

Example 7.3.1.

It may not always be possible to perform Gaussian elimination on a symmetric matrix, using pivots chosen from the diagonal. Consider, for example, the nonsingular symmetric matrix

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

If we take $\epsilon = 0$, then both diagonal elements are zero, and symmetric Gaussian elimination 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 in general destroy symmetry!

We will prove that Gaussian elimination without pivoting can be carried out with positive pivot elements if and only if A is real and symmetric positive definite. (The same result applies to complex Hermitian matrices, but since the modifications necessary for this case are straightforward, we discuss here only the real case.) For symmetric semidefinite matrices symmetric pivoting can be used. The *indefinite* case requires more substantial modifications, which will be discussed in Section 7.3.4.

Theorem 7.3.4.

The symmetric matrix $A \in \mathbf{R}^{n \times n}$ is positive definite if and only if there exists a unit lower triangular matrix L and a diagonal matrix D with positive elements such that

$$A = LDL^T, \quad D = \text{diag}(d_1, \dots, d_n),$$

Proof. Assume first that we are given a symmetric matrix A , for which Algorithm 7.3.1 yields a factorization $A = LDL^T$ with positive pivotal elements $d_k > 0$, $k = 1 : n$. Then for all $x \neq 0$ we have $y = L^T x \neq 0$ and

$$x^T Ax = x^T LDL^T x = y^T Dy > 0.$$

It follows that A is positive definite.

The proof of the other part of the theorem is by induction on the order n of A . The result is trivial if $n = 1$, since then $D = d_1 = A = a_{11} > 0$ and $L = 1$. Now write

$$A = \begin{pmatrix} a_{11} & a^T \\ a & \tilde{A} \end{pmatrix} = L_1 D_1 L_1^T, \quad L_1 = \begin{pmatrix} 1 & 0 \\ d_1^{-1}a & I \end{pmatrix}, \quad D_1 = \begin{pmatrix} d_1 & 0 \\ 0 & B \end{pmatrix},$$

where $d_1 = a_{11}$, $B = \tilde{A} - d_1^{-1}aa^T$. Since A is positive definite it follows that D_1 is positive definite, and therefore $d_1 > 0$, and B is positive definite. Since B is of order $(n-1)$, by the induction hypothesis there exists a unique unit lower triangular matrix \tilde{L} and diagonal matrix \tilde{D} with positive elements such that $B = \tilde{L}\tilde{D}\tilde{L}^T$. Then it holds that $A = LDL^T$, where

$$L = \begin{pmatrix} 1 & 0 \\ d_1^{-1}a & \tilde{L} \end{pmatrix}, \quad D = \begin{pmatrix} d_1 & 0 \\ 0 & \tilde{D} \end{pmatrix}.$$

□

Theorem 7.3.4 yields also the following useful characterization of a positive definite matrix.

Theorem 7.3.5 (Sylvester's Criterion).

Let $A \in \mathbf{R}^{n \times n}$ be symmetric and $A_k \in \mathbf{R}^{k \times k}$, $k = 1 : n$, 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 symmetric Gaussian elimination is carried out without pivoting then

$$\det(A_k) = d_1 d_2 \cdots d_k.$$

Hence, $\det(A_k) > 0$, $k = 1 : n$, if and only if all pivots are positive. However, by Theorem 7.3.2 this is the case if and only if A is positive definite. □

In order to prove a bound on the growth ratio for the symmetric positive definite we first show the following

Lemma 7.3.6. *For a symmetric positive definite matrix $A = (a_{ij}) \in \mathbf{R}^{n \times n}$ the maximum element of A lies on the diagonal.*

Proof. Theorem 7.3.2 and Sylvester's criterion imply that

$$0 < \det \begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix} = a_{ii}a_{jj} - a_{ij}^2, \quad 1 \leq i, j \leq n.$$

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 7.3.7.

Let A be symmetric and positive definite. Then in Gaussian elimination without pivoting the growth ratio satisfies $\rho_n \leq 1$.

Proof. In Algorithm 7.3.1 the diagonal elements are transformed in the k :th step of Gaussian elimination according to

$$a_{ii}^{(k+1)} = a_{ii}^{(k)} - (a_{ki}^{(k)})^2/a_{kk}^{(k)} = a_{ii}^{(k)} \left(1 - (a_{ki}^{(k)})^2/(a_{ii}^{(k)} a_{kk}^{(k)})\right).$$

If A is positive definite so are $A^{(k)}$ and $A^{(k+1)}$. Using Lemma 7.3.6 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

Example 7.3.2. The Hilbert matrix $H_n \in \mathbf{R}^{n \times n}$ with elements

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

can be shown to be positive definite for all n . If Gaussian elimination without pivoting is carried out (in exact arithmetic), then the pivots are positive. For example, for $n = 4$, symmetric Gaussian elimination yields the $H_4 = LDL^T$, where

$$D = \begin{pmatrix} 1 & & & \\ & 1/12 & & \\ & & 1/180 & \\ & & & 1/2800 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & & & \\ 1/2 & 1 & & \\ 1/3 & 1 & 1 & \\ 1/4 & 9/10 & 3/2 & 1 \end{pmatrix}.$$

7.3.2 Cholesky Factorization

Let A be a symmetric positive definite matrix A . Then the LDL^T factorization (7.3.2) exists and $D > 0$. Thus, we can write

$$A = LDL^T = (LD^{1/2})(LD^{1/2})^T, \quad D^{1/2} = \text{diag}(\sqrt{d}_1, \dots, \sqrt{d}_n). \quad (7.3.4)$$

If we redefine the upper triangular matrix $L := D^{1/2}L^T$ we obtain the **Cholesky factorization** of A ,

$$A = LL^T = R^T R, \quad (7.3.5)$$

and L (and R) is called the **Cholesky factor** of A .¹⁸ If we take the diagonal elements of L to be positive it follows from the uniqueness of the LDL^T factorization that this factorization is unique.

As for the compact schemes for LU factorization (see Section 7.2.6) it is possible to arrange the computations so that the elements in the Cholesky factor $R = (r_{ij})$ are determined directly. The matrix equation $A = LL^T$ with L lower triangular can be written

$$a_{ij} = \sum_{k=1}^j l_{ik}l_{jk} = \sum_{k=1}^{j-1} l_{ik}l_{jk} + l_{ij}l_{jj}, \quad 1 \leq j \leq i \leq n. \quad (7.3.6)$$

This is $n(n+1)/2$ 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 by the maximum diagonal element in A . Solving for r_{ij} from the corresponding equation in (7.3.6), we obtain

$$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk} \right) / l_{jj}, \quad j < i, \quad l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \right)^{1/2}.$$

If properly sequenced, these equations can be used in a recursive fashion to compute the elements in L .

As for the LU factorization there are several different ways to sequence the computation of the elements in L . A difference is that if the matrix A is symmetric and positive definite, then no pivoting is needed. In this case the elements in L can be computed one row or one column at a time; see Figure 7.3.1.

The row- and columnwise algorithm can be derived as follows. Consider the partitioned Cholesky decomposition

$$\begin{pmatrix} A_{11} & a_{21} & A_{31}^T \\ a_{21}^T & a_{22} & a_{32} \\ A_{31} & a_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 & 0 \\ l_{21}^T & l_{22} & 0 \\ L_{31} & l_{32} & L_{33} \end{pmatrix} \begin{pmatrix} L_{11}^T & l_{21} & L_{31}^T \\ 0 & l_{22} & l_{32}^T \\ 0 & 0 & L_{33}^T \end{pmatrix},$$

¹⁸André-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 and his work was posthumously published by a fellow officer, Benoit in 1924.

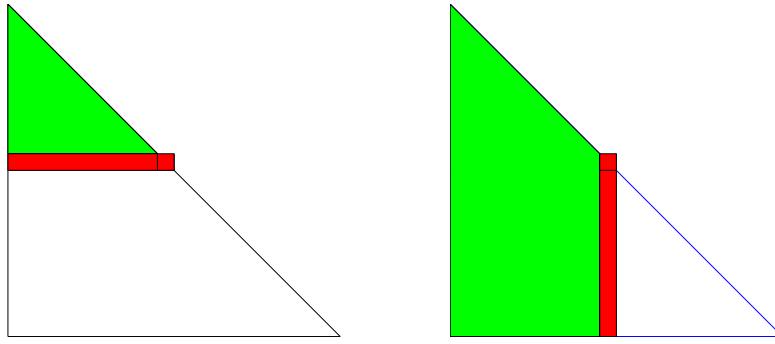


Figure 7.3.1. Computations in the k th step of the Cholesky methods. Left: The row sweep method. Right: The column sweep method.

where A_{11} is of order $k - 1$ and a_{22} a scalar. Equating the elements in the k th column on both sides gives

$$\begin{pmatrix} a_{21} \\ a_{22} \\ a_{32} \end{pmatrix} = \begin{pmatrix} L_{11}l_{21} \\ l_{21}^T l_{21} + l_{22}^2 \\ L_{31}l_{21} + l_{32}l_{22} \end{pmatrix}.$$

The last two equations give

$$l_{22} = (a_{22} - l_{21}^T l_{21})^{1/2}, \quad l_{32} = (a_{32} - L_{31}l_{21})/l_{22}. \quad (7.3.7)$$

This gives the elements in the k th column of L in terms of the first $k - 1$ columns, i.e. the columnwise version of Cholesky algorithm. The rowwise algorithm is obtained using the two first equations to get

$$l_{21} = L_{11}^{-1}a_{21}, \quad l_{22} = (a_{22} - l_{21}^T l_{21})^{1/2}. \quad (7.3.8)$$

This gives the elements in the k th row of L in terms of the first $k - 1$ rows, i.e. the rowwise version. These two versions of the Cholesky algorithm are not only mathematically equivalent but also *numerically equivalent*, i.e., they will compute the same Cholesky factor also when taking rounding errors into account. The algorithms require n square roots and approximately $n^3/3$ flops.

Assume that the lower half of A has been copied into L . Then the columnwise version of Cholesky can be described as follows;

```

for  $k = 1 : n$ 
  for  $j = 1 : k - 1$ 
     $l_{k:n,k} = l_{k:n,k} - l_{k,j}l_{k:n,j};$ 
  end
   $l_{k,k} = \sqrt{l_{k,k}};$ 
   $l_{k+1:n,k} = l_{k+1:n,k}/l_{k,k};$ 
end
```

Some applications lead to linear systems where $A \in \mathbf{R}^{n \times n}$ is a symmetric positive semidefinite matrix ($x^T Ax \geq 0$ for all $x \neq 0$) with $\text{rank}(A) = r < n$. One source is rank deficient least squares problems; see Section 8.4. Another example is when the finite element method is applied to a problem where rigid body motion occurs, which implies $r \leq n - 1$. In the semidefinite case the Cholesky factorization needs to be modified by performing symmetric pivoting.

Theorem 7.3.8.

Let $A \in \mathbf{R}^{n \times n}$ be a symmetric positive semidefinite matrix of rank $r < n$. Then there is a permutation matrix P so that the symmetric matrix $P^T AP$ has a Cholesky factorization

$$P^T AP = LL^T, \quad L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & 0 \end{pmatrix} \quad (7.3.9)$$

$L_{11} \in \mathbf{R}^{r \times r}$ has positive diagonal elements. Hence, L has a lower trapezoidal form

We prove the theorem by describing an outer product Cholesky algorithm to compute L , which essentially is the same as Algorithm 7.5. At the k th step, a symmetric rank one matrix is subtracted from A . Ignoring pivoting, we have at the start of the k th step,

$$A^{(k)} = (a_{ij}^{(k)}) = A - \sum_{j=1}^{k-1} l_j l_j^T = \begin{matrix} & \begin{matrix} k-1 & n-k+1 \end{matrix} \\ \begin{matrix} 0 & 0 \\ 0 & A_{22}^{(k)} \end{matrix} & \end{matrix}. \quad (7.3.10)$$

where $l_j = (0, \dots, 0, l_{jj}, \dots, l_{jn})$. Here $A_{22}^{(k)}$ is the Schur complement of the leading $k \times k$ principal minor of A . To advance one step we compute for $k = 1 : n$,

$$\begin{aligned} l_{kk} &= \sqrt{a_{kk}^{(k)}}, \\ l_{ik} &= a_{ik}^{(k)} / l_{kk}, \quad i = k + 1 : n, \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - l_{ik} l_{jk}^T, \quad j < i = k + 1 : n. \end{aligned}$$

Symmetry means that pivot elements can only be chosen from the diagonal. In the k th elimination step of Algorithm 7.5 a maximal diagonal element $a_{ss}^{(k)}$ in the reduced matrix $A^{(k)}$ is chosen as pivot, i.e.,

$$a_{ss}^{(k)} = \max_{k \leq i \leq n} a_{ii}^{(k)}. \quad (7.3.11)$$

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.

Due to rounding errors, negative diagonal elements can arise even for a positive semidefinite matrix A . When this happens the factorization should be terminated. A better stopping criterion is

$$\max_{k \leq i \leq n} a_{ii}^{(k)} \leq \epsilon a_{11}^{(1)}, \quad (7.3.12)$$

where $\epsilon = nu$. If (7.3.12) is satisfied then we set $\text{rank}(A) = k$ and stop.

The linear system $Ax = b$, or $P^T AP(P^T x) = P^T b$, then becomes

$$LL^T \tilde{x} = \tilde{b}, \quad \tilde{x} = P^T x, \quad \tilde{b} = P^T b.$$

Setting $z = L^T \tilde{x}$ the linear system reads

$$Lz = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} z = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix}, \quad L_{11} \in \mathbf{R}^{r \times r},$$

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 $x = P\tilde{x}$.

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 [328, Sec. 10.3.2]. Taking computational cost into consideration it is recommended that the stopping criterion

$$\max_{k \leq i \leq n} a_{ii}^{(k)} \leq \epsilon r_{11}^2 \quad (7.3.13)$$

is used, where $\epsilon = nu$ and u is the unit roundoff.

In the Cholesky factorization only elements in the upper 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 array to store R^T and not overwrite the original data. Another option is to store the elements of the upper triangular part of A columnwise in a vector. For example, the mapping of array-subscript of an upper triangular matrix of order 5 will be

$$\begin{pmatrix} 1 \\ 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{pmatrix}$$

This is known as **packed storage**. This data is then and 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 economizing storage.

Any matrix $A \in \mathbf{R}^{n \times n}$ can be written as the sum of a symmetric and a skew-symmetric part, $A = H + S$, where

$$A_H = \frac{1}{2}(A + A^T), \quad A_S = \frac{1}{2}(A - A^T). \quad (7.3.14)$$

A is symmetric if and only if $A_S = 0$. Sometimes A is called positive definite if its symmetric part A_H is positive definite. If the matrix A has a positive symmetric part then its leading principal submatrices are nonsingular and Gaussian elimination can be carried out to completion without pivoting. However, the resulting LU factorization may not be stable as shown by the example

$$\begin{pmatrix} \epsilon & 1 \\ -1 & \epsilon \end{pmatrix} = \begin{pmatrix} 1 \\ -1/\epsilon & 1 \end{pmatrix} \begin{pmatrix} \epsilon & 1 \\ & \epsilon + 1/\epsilon \end{pmatrix}, \quad (\epsilon > 0).$$

These results can be extended to complex matrices with positive definite Hermitian part $A_H = \frac{1}{2}(A + A^H)$, for which its holds that $x^H A x > 0$, for all nonzero $x \in \mathbf{C}^n$. Of particular interest are complex symmetric matrices, arising in computational electrodynamics, of the form

$$A = B + iC, \quad B, C \in \mathbf{R}^{n \times n}, \quad (7.3.15)$$

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 [237]).

7.3.3 Inertia of Symmetric Matrices

Let $A \in \mathbf{C}^{n \times n}$ be an Hermitian matrix. Hermitian matrices arise naturally in the study of quadratic forms $\psi(x) = x^H A x$. By the coordinate transformation $x = Ty$ this quadratic form is transformed into

$$\psi(Ty) = y^H \hat{A} y, \quad \hat{A} = T^H A T.$$

This mapping of A onto T^HAT is called a **congruence transformation** of A and we say that A and \hat{A} are **congruent**. (Notice that a congruence transformation with a nonsingular matrix means a transformation to a coordinate system which is usually not rectangular.)

The matrix \hat{A} is again a Hermitian matrix, but unless T is unitary the transformation does not preserve the eigenvalues of A . The **inertia** of A is defined as the number triple $\text{in}(A) = (\pi, \nu, \delta)$ of positive, negative, and zero eigenvalues of A . If A is positive definite matrix and $Ax = \lambda x$, we have

$$x^H Ax = \lambda x^H x > 0.$$

Hence, all eigenvalues must be positive and the inertia is $(n, 0, 0)$. Sylvester's famous law of inertia¹⁹ says that the inertia of A is preserved by a congruence transformation.

Theorem 7.3.9 (Sylvester's Law of Inertia).

If $A \in \mathbf{C}^{n \times n}$ is symmetric and $T \in \mathbf{C}^{n \times n}$ 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 eigenvalues. By definition we have $\text{in}(A) = \text{in}(D)$, $\text{in}(\hat{A}) = \text{in}(\hat{D})$, and hence, we want 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 $\pi \neq \hat{\pi}$, say $\pi > \hat{\pi}$, and that the eigenvalues are ordered so that $\lambda_j > 0$ for $j \leq \pi$ and $\hat{\lambda}_j > 0$ for $j \leq \hat{\pi}$. 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 $n - \pi + \hat{\pi} < n$ homogeneous linear relations

$$\xi_j = 0, \quad j > \pi, \quad \hat{\xi}_j = (S^{-1}x)_j = 0, \quad j \leq \hat{\pi}.$$

Then

$$\psi(x^*) = \sum_{j=1}^{\pi} \lambda_j |\xi_j^*|^2 > 0, \quad \psi(x^*) = \sum_{j=\hat{\pi}}^n \hat{\lambda}_j |\hat{\xi}_j^*|^2 \leq 0.$$

This is a contradiction and hence the assumption that $\pi \neq \hat{\pi}$ is false, so A and \hat{A} have the same number of positive eigenvalues. Using the same argument on $-A$ it follows that also $\nu = \hat{\nu}$, and since the number of eigenvalues is the same $\delta = \hat{\delta}$. \square

¹⁹Sylvester published the theorem in 1852 [561] but the result was later found in notes of Jacobi dated 1947 and published posthumously.

Let $A \in \mathbf{R}^{n \times n}$ be a real symmetric matrix and consider the quadratic equation

$$x^T Ax - 2bx = c, \quad A \neq 0. \quad (7.3.16)$$

The solution sets of this equation are sometimes called **conical sections**. If $b = 0$, then the surface has its center at the origin and equation (7.3.16) reads $x^T Ax = c$. The inertia of A completely determines the geometric type of the conical section.

Sylvester's theorem tells us that the geometric type of the surface can be determined without computing the eigenvalues. Since we can always multiply the equation by -1 we can assume that there are at least one positive eigenvalue. Then, for $n = 2$ there are three possibilities:

$$(2, 0, 0) \text{ ellipse; } (1, 0, 1) \text{ parabola; } (1, 1, 0) \text{ hyperbola.}$$

In n dimensions there will be $n(n+1)/2$ cases, assuming that at least one eigenvalue is positive.

7.3.4 Symmetric Indefinite Matrices

If A is a symmetric indefinite matrix then a factorization of the form $A = LDL^T$, where D is diagonal may not exist. Even if it does exist, it can be ill-conditioned because of unbounded element growth as the following simple example shows.

Example 7.3.3.

The symmetric indefinite matrix

$$A = \begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix}, \quad 0 < \epsilon \ll 1,$$

is well conditioned. The LDL^T factorization of A equals

$$A = \begin{pmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{pmatrix} \begin{pmatrix} \epsilon & 0 \\ 0 & \epsilon - \epsilon^{-1} \end{pmatrix} \begin{pmatrix} 1 & \epsilon^{-1} \\ 0 & 1 \end{pmatrix},$$

and there is unbounded element growth in the factors.

A stable way of factorizing an indefinite matrix is, of course, to compute an LU factorization using partial pivoting. But since partial pivoting destroys symmetry, this will use twice the storage space of a symmetric factorization. Further, an LU factorization does not give 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 blocks of order 1 or 2. Note that a symmetric block factorization has the form

$$A = \begin{pmatrix} B & C^T \\ C & E \end{pmatrix} = \begin{pmatrix} I_s & 0 \\ CB^{-1} & I_s \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & E - CB^{-1}C^T \end{pmatrix} \begin{pmatrix} I_s & B^{-1}C^T \\ 0 & I_s \end{pmatrix}. \quad (7.3.17)$$

where $s = 1$ or 2 . Since the Schur complement $E - CB^{-1}C^T$ is symmetric this can be repeated. By a symmetric permutation any nonsingular 2×2 principal submatrix can be brought to the upper left corner and used as pivot. Then

$$B^{-1} = \begin{pmatrix} a_{11} & a_{r1} \\ a_{r1} & a_{rr} \end{pmatrix}^{-1} = \frac{1}{\delta_{1r}} \begin{pmatrix} a_{rr} & -a_{r1} \\ -a_{r1} & a_{11} \end{pmatrix}, \quad \delta_{1r} = a_{11}a_{rr} - a_{r1}^2, \quad (7.3.18)$$

which determines the first two columns CB^{-1} of a unit lower triangular matrix. The elements in the Schur complement are

$$a_{ij}^{(3)} = a_{ij} - l_{i1}a_{1j} - l_{ir}a_{rj}, \quad 2 \leq j \leq i \leq n. \quad (7.3.19)$$

It can be shown that the reduced matrix is the same as if two steps of Gaussian elimination were taken, first pivoting on the element a_{12} and then on a_{21} .

A similar reduction is used if 2×2 pivots are taken at a later stage in the factorization. Ultimately a factorization $A = LDL^T$ is computed in which D is block diagonal with in general a mixture of 1×1 and 2×2 blocks, and L is unit lower triangular with $l_{k+1,k} = 0$ when $A^{(k)}$ is reduced by a 2×2 pivot. 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 which dominates the operation count is (7.3.19), and this is twice the work as for an 1×1 pivot. Therefore, the leading term in the operations count is always $n^3/6$, whichever type of pivots is used.

The main issue then is to find a pivotal strategy that will give control of element growth without requiring too much search. One possible strategy is comparable to that of complete pivoting. Consider the first stage of the factorization and 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 an 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 an 1×1 pivot. The magical number α has been chosen so as 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 straight forward but tedious (see Higham [328, Sec. 11.1.1]) is omitted here.

With this choice the element growth can be shown to be bounded by

$$\rho_n \leq (1 + 1/\alpha)^{n-1} < (2.57)^{n-1}. \quad (7.3.20)$$

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 occur at every step. The bound in (7.3.20) can be compared to the bound 2^{n-1} , which holds for Gaussian elimination with partial pivoting. 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 (7.3.20) can be achieved using the following partial pivoting strategy due to Bunch and Kaufman [84]. For simplicity of notations we restrict our attention to the first stage of the elimination. All later stages proceed similarly. First determine the off-diagonal element of largest magnitude in the first column,

$$\lambda = |a_{r1}| = \max_{i \neq 1} |a_{i1}|.$$

If $|a_{11}| \geq \alpha\lambda$, then take a_{11} as pivot. Else, determine the largest off-diagonal element in column r ,

$$\sigma = \max_{1 \leq i \leq n} |a_{ir}|, \quad i \neq r.$$

If $|a_{11}| \geq \alpha\lambda^2/\sigma$, then again take a_{11} as pivot, else if $|a_{rr}| \geq \alpha\sigma$, take a_{rr} as pivot. Otherwise take as pivot the 2×2 principal submatrix

$$\begin{pmatrix} a_{11} & a_{1r} \\ a_{1r} & a_{rr} \end{pmatrix}.$$

Note that 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.

Normwise backward stability can be shown to hold also for the Bunch–Kaufman pivoting strategy. However, it is no longer true that the elements of L are bounded independently of A . The following example (Higham [328, Sec. 11.1.2]) shows that L is unbounded:

$$A = \begin{pmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 0 & 1 & \\ \epsilon^{-1} & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & \epsilon & \\ \epsilon & 0 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \epsilon^{-1} \\ 1 & 1 & 0 \\ & & 1 \end{pmatrix}. \quad (7.3.21)$$

Note that 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 .

The method can be implemented for A stored in conventional form as an $n \times n$ array or stored as an 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 .

$$\left(\begin{array}{c|cc|ccc} d_{11} & l_{21} & l_{31} & l_{41} & l_{51} & l_{61} \\ \hline a_{21} & d_{22} & d_{32} & l_{42} & l_{52} & l_{62} \\ a_{31} & a_{32} & d_{33} & l_{12} & l_{12} & l_{63} \\ \hline a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array} \right)$$

For solving a linear system $Ax = b$ the LDL^T 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 [14]. 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 the 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 an 1×1 pivot is taken, according to the test

$$\max(|a_{rr}|, |a_{ss}|) \geq \alpha |a_{rs}|.$$

Aasen [1] has given an algorithm that for a symmetric matrix $A \in \mathbf{R}^{n \times n}$ computes the factorization

$$PAP^T = LTL^T, \quad (7.3.22)$$

where L is unit lower triangular and T symmetric tridiagonal.

None of the algorithms described here preserves the band structure of the matrix A . In this case Gaussian elimination with partial pivoting can be used but as remarked before this will destroy symmetry and does not reveal the inertia. For the special case of a *tridiagonal* symmetric indefinite matrices an algorithm for computing an LDL^T factorization will be given in Section 7.4.4.

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. Further, since

$$(x^T Ax)^T = x^T A^T x = -x^T Ax,$$

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

$$|a_{p,q}| = \max\{\max_{1 < i \leq n} |a_{i,1}|, \max_{1 < i \leq n} |a_{i,2}|\},$$

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 is smaller than for Gaussian elimination with partial pivoting for a general matrix.

Review Questions

- 3.1** (a) Give two necessary and sufficient conditions for a real symmetric matrix A to be positive definite.
 (b) Show that if A is symmetric positive definite so is its inverse A^{-1} .
 - 3.2** What simplifications occur in Gaussian elimination applied to a symmetric positive definite matrix?
 - 3.3** What is the relation of Cholesky factorization to Gaussian elimination? Give an example of a symmetric matrix A , for which the Cholesky decomposition does not exist.
 - 3.4** Show that if A is skew-symmetric, then iA is Hermitian.
 - 3.5** Show that the Cholesky factorization is unique for positive definite matrices provided R is normalized to have positive diagonal entries.
 - 3.6** (a) Formulate and prove Sylvester's law of inertia.
 (b) Show that for $n = 3$ there are six different geometric types of conical sections $x^T Ax - 2b^T x = c$, provided that $A \neq 0$ and is normalized to have at least one positive eigenvalue.
-

Problems

- 3.1** If the matrix A is symmetric positive definite, how should you compute $x^T Ax$ for a given vector x ?
- 3.2** Let the matrix A be symmetric positive definite. Show that $|a_{ij}| \leq (a_{ii} + a_{jj})/2$.
- 3.3** Show by computing the Cholesky factorization $A = LL^T$ that the matrix

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$$

is positive definite.

3.4 Let $A = (a_{ij}) \in \mathbf{R}^{n \times n}$ be a symmetric positive definite matrix. Prove the special case of Hadamard's inequality

$$|\det A| \leq \prod_{i=1}^n a_{ii}. \quad (7.3.23)$$

where equality holds only if A is diagonal.

Hint: Use the Cholesky decomposition $A = R^T R$ and show that $\det A = (\det R)^2$.

3.5 Show that the matrix

$$A = \begin{pmatrix} a & -1 & \cdots & -1 \\ -1 & a & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & \cdots & a \end{pmatrix} \in \mathbf{R}^{n \times n}$$

is positive definite if $a > \sqrt{n}$.

Hint: Reverse the rows and columns of A and then compute the Cholesky factorization.

3.6 The Hilbert matrix $H_n \in \mathbf{R}^{n \times n}$ with elements

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

is symmetric positive definite for all n . Denote by \bar{H}_4 the corresponding matrix with elements rounded to five decimal places, and compute its Cholesky factor \bar{L} . Then compute the difference $(\bar{L}\bar{L}^T - \bar{A})$ and compare it with $(A - \bar{A})$.

3.6 Let $A + iB$ be Hermitian and positive definite, where $A, B \in \mathbf{R}^{n \times n}$. Show that the real matrix

$$C = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}$$

is symmetric and positive definite. How can a linear system $(A + iB)(x + iy) = b + ic$ be solved using a Cholesky factorization of C ?

3.7 Implement the Cholesky factorization using packed storage for A and R .

7.4 Banded Linear Systems

7.4.1 Band Matrices

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 is said to have **upper bandwidth** r and to have **lower bandwidth** s if r and s are the smallest integers such that

$$a_{ij} = 0, \quad j > i + r, \quad a_{ij} = 0, \quad i > j + s, \quad (7.4.1)$$

respectively. This means that the number of nonzero diagonals above and below the main diagonal are r and s respectively. The maximum number of nonzero elements

in any row is then $w = r + s + 1$, which is the **bandwidth** of A . 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 $r = 1$, $s = 2$ and $w = 4$. For a symmetric matrix the lower and upper bandwidth are equal $W = r = s$. In this case we can also define the (half) bandwidth

$$\beta = \max\{|i - j| \mid a_{ij} \neq 0\}. \quad (7.4.2)$$

Linear systems $Ax = b$ where the matrix A has a small bandwidth arise in problems where each variable x_i is coupled by an equation only to a few other variables x_j such that $|j - i|$ is small. Note that the bandwidth of a matrix depends on the ordering of its rows and columns. An important, but hard, problem is to find an optimal ordering of columns that minimize the bandwidth. However, there are good heuristic algorithms that can be used in practice and give almost optimal results; see Section 7.5.1.

A band matrix $A \in \mathbf{R}^{n \times n}$ may be stored by diagonals in an array of dimension $n \times (r + s + 1)$ or $(r + s + 1) \times n$. For example, the matrix above can be stored as

$$\begin{array}{ccccccccc}
 * & * & a_{11} & a_{12} & & & & & \\
 * & a_{21} & a_{22} & a_{23} & & * & a_{12} & a_{23} & a_{34} & a_{45} & a_{56} \\
 a_{31} & a_{32} & a_{33} & a_{34} & , & \text{or} & a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\
 a_{42} & a_{43} & a_{44} & a_{45} & & & a_{21} & a_{32} & a_{43} & a_{54} & a_{65} & * \\
 a_{53} & a_{54} & a_{55} & a_{56} & & & a_{31} & a_{42} & a_{53} & a_{64} & * & * \\
 a_{64} & a_{65} & a_{66} & * & & & & & & & &
 \end{array}$$

Notice that except for a few elements indicated by asterisks in the initial and final rows, only nonzero elements of A are stored. For example, passing along a row in the second storage scheme above moves along a diagonal of the matrix, and the columns are aligned.

Several classes of band matrices that occur frequently have special names. A matrix for which $r = s = 1$ is called **tridiagonal**; e.g.,

$$A = \begin{pmatrix} a_1 & c_2 & & & \\ b_2 & a_2 & c_3 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-1} & a_{n-1} & c_n \\ & & & b_n & a_n \end{pmatrix}.$$

Note that the $3n - 2$ nonzero elements in A can be conveniently be stored in three vectors a , b and c of length $\leq n$.

If $r = 0$, $s = 1$ ($r = 1$, $s = 0$) the matrix is called lower (upper) **bidiagonal**. A matrix with $s = 1$ ($r = 1$) is called an upper (lower) **Hessenberg** matrix²⁰.

²⁰Named after the German mathematician and engineer Karl Hessenberg (1904–1959). These matrices first appeared in [320].

It is convenient to introduce some additional notations for manipulating band matrices.²¹

Definition 7.4.1.

If $a = (a_1, a_2, \dots, a_{n-r})^T$ is a column vector with $n - r$ components, then

$$A = \text{diag}(a, k), \quad |k| < n,$$

is a square matrix of order n 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.

Let A be a square matrix of order n . Then

$$\text{diag}(A, k) \in \mathbf{R}^{(n-k)}, \quad |k| < n,$$

is the column vector formed from the elements of the k th diagonal of A .

For example, $\text{diag}(A, 0) = \text{diag}(a_{11}, a_{22}, \dots, a_{nn})$, is the main diagonal of A , and if $0 \leq k < n$,

$$\begin{aligned} \text{diag}(A, k) &= \text{diag}(a_{1,k+1}, a_{2,k+2}, \dots, a_{n-k,n}), \\ \text{diag}(A, -k) &= \text{diag}(a_{k+1,1}, a_{k+2,2}, \dots, a_{n,n-k}), \end{aligned}$$

is the k th superdiagonal and subdiagonal of A , respectively.

7.4.2 Multiplication of Band Matrices

Clearly the product of two diagonal matrices D_1 and D_2 is another diagonal matrix where the elements are equal to the elementwise product of the diagonals. What can be said of the product of two band matrices? An elementary, but very useful result tells us which diagonals in the product are nonzero.

Lemma 7.4.2.

Let $A, B \in \mathbf{R}^{n \times n}$ have lower (upper) bandwidth r and s respectively. Then the product AB has lower (upper) bandwidth $r + s$.

Assume that A and B are band matrices of order n , which 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. This gives the following rule:

Lemma 7.4.3.

Let $A = \text{diag}(a, r)$ and $B = \text{diag}(b, s)$ and set $C = AB$. If $|r + s| \geq n$ then $C = 0$; otherwise $C = \text{diag}(c, r + s)$, where the elements of the vector $c \in \mathbf{R}^{(n-|r+s|)}$

²¹These notations are taken from MATLAB.

are obtained by pointwise multiplication of shifted vectors a and b :

$$c = \begin{cases} (a_1 b_{r+1}, \dots, a_{n-r-s} b_{n-s})^T, & \text{if } r, s \geq 0, \\ (a_{|s|+1} b_1, \dots, a_{n-|r|} b_{n-|r+s|})^T, & \text{if } r, s \leq 0, \\ (0, \dots, 0, a_1 b_1, \dots, a_{n-s} b_{n-s})^T, & \text{if } r < 0, \quad s > 0, \quad r + s \geq 0, \\ (0, \dots, 0, a_1 b_1, \dots, a_{n-|r|} b_{n-|r|})^T, & \text{if } r < 0, \quad s > 0, \quad r + s < 0, \\ (a_1 b_{|r+s|+1}, \dots, a_{n-r} b_{n-|s|}, 0, \dots, 0)^T, & \text{if } r > 0, \quad s < 0, \quad r + s \geq 0, \\ (a_{r+1} b_1, \dots, a_{n-r} b_{n-|s|}, 0, \dots, 0)^T, & \text{if } r > 0, \quad s < 0, \quad r + s < 0. \end{cases} \quad (7.4.3)$$

Note that when $rs < 0$, zeros are added at the beginning or end to get a vector c of length $n - |r + s|$.

Example 7.4.1.

The number of cases in this lemma looks a bit forbidding, so to clarify the result we consider a specific case. Let A and B be tridiagonal matrices of size $n \times n$

$$A = \begin{pmatrix} a_1 & c_1 & & & \\ b_1 & a_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2} & a_{n-1} & c_{n-1} \\ & & & b_{n-1} & a_n \end{pmatrix}, \quad B = \begin{pmatrix} d_1 & f_1 & & & \\ e_1 & d_2 & f_2 & & \\ & \ddots & \ddots & \ddots & \\ & & e_{n-2} & d_{n-1} & f_{n-1} \\ & & & e_{n-1} & d_n \end{pmatrix}.$$

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 are exactly the same as in the conventional schemes, but only $3^2 = 9$ pointwise vector multiplications are required.

7.4.3 LU Factorization of Band Matrices

Band matrices are well-suited for LU factorization. If no pivoting is required then *the band structure is preserved*. Recall that pivoting is not needed for stability, e.g., when A is diagonally dominant or symmetric and positive definite.

Theorem 7.4.4.

Let A be a band matrix with upper bandwidth r and lower bandwidth s . If A has an LU-decomposition $A = LU$, then U has upper bandwidth r and L lower bandwidth s .

Proof. The factors L and U are unique and can be computed, for example, by Doolittle's method (7.2.18). Assume that the first $k - 1$ rows of U and columns of L have bandwidth r and s , that is, for $p = 1 : k - 1$

$$l_{ip} = 0, \quad i > p + s, \quad u_{pj} = 0, \quad j > p + r. \quad (7.4.4)$$

The proof is by induction in k . The assumption is trivially true for $k = 1$. Since $a_{kj} = 0$, $j > k + r$ we have from (7.2.12) and (7.4.4)

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj} = 0 - 0 = 0, \quad j > k + r.$$

Similarly, it follows that $l_{ik} = 0$, $i > k + s$, which completes the induction step. \square

It is important to note that zeros within the band usually of A are in general not preserved and L and U will be full band matrices. Algorithm 7.2, Gaussian elimination without pivoting, should be modified as follows to operate only on elements within the band. The algorithms given below are written assuming that the matrix is conventionally stored. A useful exercise for the reader is to rewrite them for the case when A , L , and U are stored by diagonals.

Algorithm 7.6. Band Gaussian Elimination.

Let $A \in \mathbf{R}^{n \times n}$ be a given matrix with upper bandwidth r and lower bandwidth s . The following algorithm computes the LU factorization of A , provided it exists. The element a_{ij} is overwritten by l_{ij} if $i > j$ and by u_{ij} otherwise.

```

for  $k = 1 : n - 1$ 
    for  $i = k + 1 : \min(k + s, n)$ 
         $l_{ik} := a_{ik}^{(k)} / a_{kk}^{(k)}$ ;
        for  $j = k + 1 : \min(k + r, n)$ 
             $a_{ij}^{(k+1)} := a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$ ;
        end
    end
end

```

An operation count shows that this algorithm requires t flops, where

$$t = \begin{cases} 2nr(s+1) - rs^2 - \frac{1}{3}r^3, & \text{if } r \leq s; \\ 2ns(s+1) - \frac{4}{3}s^3, & \text{if } r = s; \\ 2ns(r+1) - sr^2 - \frac{1}{3}s^3, & \text{if } r > s. \end{cases}$$

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 7.4.3. The solution of the two band triangular systems $Ly = b$ and $Ux = y$ are obtained from

$$\begin{aligned} y_i &= b_i - \sum_{\substack{j=s \\ \max(1,i-s)}}^{i-1} l_{ij} y_j, \quad i = 1 : n \\ x_i &:= \left(y_i - \sum_{j=i+1}^{\min(i+r,n)} u_{ij} x_j \right) / u_{ii}, \quad i = n : (-1) : 1. \end{aligned}$$

These algorithms require $(2n - s)s$ and $(2n - r)(r + 1)$ flops, respectively. They are easily modified so that y and x overwrites b in storage.

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:

$$\begin{array}{ccccccccc} a_{31} & a_{32} & a_{33} & a_{34} & & u_{11} & u_{12} & u_{13} & u_{14} \\ a_{21} & a_{22} & a_{23} & & & l_{21} & a_{22}^{(2)} & a_{23}^{(2)} & \mathbf{a}_{24}^{(2)} \\ a_{11} & a_{12} & & & \implies & l_{31} & a_{32}^{(2)} & \mathbf{a}_{33}^{(2)} & \mathbf{a}_{34}^{(2)} \\ a_{42} & a_{43} & a_{44} & a_{45} & & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{53} & a_{54} & a_{55} & a_{56} & & a_{53} & a_{54} & a_{55} & a_{56} \\ \dots & & & & & \dots & & & \dots \end{array}.$$

where fill elements are shown in boldface. Hence, *the upper bandwidth of U may 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.

Example 7.4.2.

Upper (lower) Hessenberg matrices are unsymmetric band matrices with $s = 1$ ($r = 1$). Such matrices are of particular interest when solving unsymmetric eigenvalue problems. An upper Hessenberg matrix of order five has the structure

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ 0 & h_{32} & h_{33} & h_{34} & h_{35} \\ 0 & 0 & h_{43} & h_{44} & h_{45} \\ 0 & 0 & 0 & h_{54} & h_{55} \end{pmatrix}. \quad (7.4.5)$$

Performing Gaussian elimination the first step will only affect the first two rows of the matrix. The reduced matrix is again Hessenberg and all the remaining steps are similar to the first. If partial pivoting is used then in the first step either h_{11} or

h_{21} will be chosen as pivot. Since these rows have the same structure the Hessenberg form will be preserved during the elimination. Clearly, only $t = n(n + 1)$ flops are needed. Note that with partial pivoting the elimination will not give a factorization $PA = LU$ with L lower bidiagonal. Whenever we pivot, the interchanges should be applied also to L , which will spread out the elements. Therefore, L will be lower triangular with only one nonzero off-diagonal element in each column. However, it is more convenient to leave the elements in L in place.

If $A \in \mathbf{R}^{n \times n}$ is Hessenberg then $\rho_n \leq n$ with partial pivoting. This follows since at the start of the k stage row $k + 1$ of the reduced matrix has not been changed and the pivot row has elements of modulus at most k times the largest element of H .

In the special case when Let A be a symmetric positive definite band matrix with upper and lower bandwidth $r = s$. Then a simple corollary of Theorem 7.4.4 is that the factor L in the Cholesky factorization $A = LL^T$ has lower bandwidth r . From Algorithm 7.5 we easily derive the following band version:

Algorithm 7.7. *Band Cholesky Algorithm.*

```

for  $j = 1 : n$ 
     $p = \max(1, j - r);$ 
    for  $i = p : j - 1$ 
         $r_{ij} = \left( a_{ij} - \sum_{k=p}^{i-1} r_{ki} r_{kj} \right) / r_{ii};$ 
    end
     $r_{jj} = \left( a_{jj} - \sum_{k=p}^{j-1} r_{kj}^2 \right)^{1/2};$ 
end

```

If $r \ll n$ this algorithm requires about $nr(r + 3)$ flops and n square roots. As input we just need the upper triangular part of A , which can be stored in an $n \times (r + 1)$ array.

Example 7.4.3.

Band matrices arise frequently in, for example, the numerical solution of boundary value problems for ordinary and partial differential equations. Consider 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 we impose a uniform square mesh of size $h = 1/(n + 1)$ on Ω . Let $u_{i,j}$ denote approximations to $u(x_i, y_j)$ at the mesh points $x_i = ih$, $y_j = jh$. Approximating the second

derivatives by symmetric difference quotients at interior mesh points gives an 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 unknown values $u_{i,j}$. If the mesh points are enumerated line by line (the so-called “natural” or lexicographical ordering) and a vector u is formed from the unknown function values, the difference equation can then be written in matrix form as

$$Au = b, \quad u = (u_1, u_2, \dots, u_{n^2}),$$

where u_i is the vector of unknowns in the i th line and the right hand side is formed by the known values of $u(x, y)$ at the boundary. Note that the matrix A is symmetric by construction.

It can be verified that for the model problem the matrix $A \in \mathbf{R}^{n^2 \times n^2}$ has lower and upper bandwidth n and block-tridiagonal form

$$A = \begin{pmatrix} 2I + T & -I & & \\ -I & 2I + T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & 2I + T \end{pmatrix}, \quad T = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ \ddots & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}. \quad (7.4.6)$$

where $T \in \mathbf{R}^{n \times n}$ is symmetric tridiagonal,

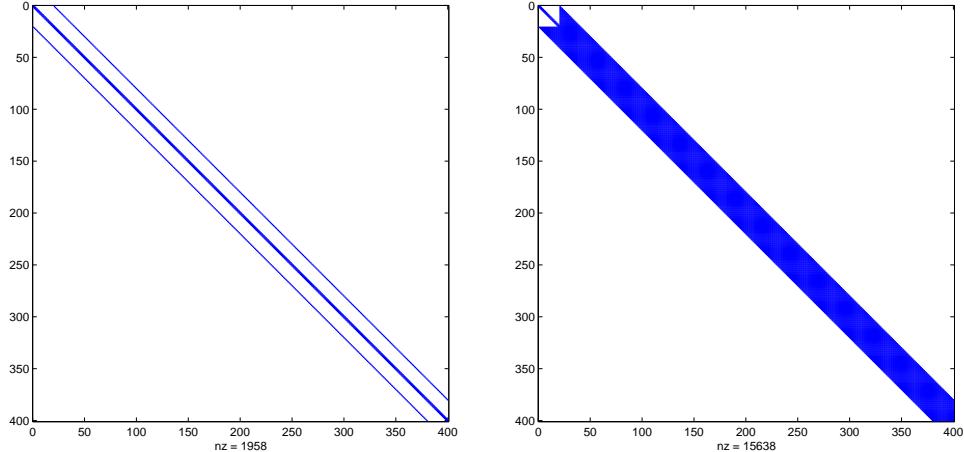


Figure 7.4.1. Structure of A (left) and $L + L^T$ (right) for the Laplace equation, $n = 20$ with rowwise ordering of the unknowns.

The matrix A in (7.4.6) is symmetric positive definite and its structure is shown in Figure 7.4.1 (left). In the Cholesky factorization of A the zero elements inside the outer diagonals of A will fill in. Hence, even if symmetry and band

structure is taking into account, the Cholesky factorization will require about 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 , as shown in Figure 7.4.1 (right).

7.4.4 Tridiagonal Linear Systems

A matrix of the form

$$A = \begin{pmatrix} a_1 & c_2 & & & \\ b_2 & a_2 & c_3 & & \\ \ddots & \ddots & \ddots & \ddots & \\ & b_{n-1} & a_{n-1} & c_n & \\ & & b_n & a_n & \end{pmatrix}. \quad (7.4.7)$$

is called **tridiagonal**. Note that the $3n - 2$ nonzero elements in A are conveniently stored in three vectors a, c , and d . A is irreducible (see Definition 7.1.3) if b_i and c_i are nonzero for $i = 2 : n$. Let A be reducible, say $c_k = 0$. Then A can be written as a lower block triangular form

$$A = \begin{pmatrix} A_1 & 0 \\ L_1 & A_2 \end{pmatrix},$$

where A_1 and A_2 are tridiagonal. If A_1 or A_2 is reducible then this blocking can be applied recursively until a block form with irreducible tridiagonal blocks is obtained.

If Gaussian elimination with partial pivoting 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 (cf. the Hessenberg case in Example 7.4.2). If A is diagonally dominant, then no pivoting is required and the factorization $A = LU$ exists. By Theorem 7.4.4 it has the form

$$A = LU = \begin{pmatrix} 1 & & & \\ \gamma_2 & 1 & & \\ & \gamma_3 & \ddots & \\ & & \ddots & 1 \\ & & & \gamma_n & 1 \end{pmatrix} \begin{pmatrix} d_1 & c_2 & & & \\ & d_2 & c_3 & & \\ & & \ddots & \ddots & \\ & & & d_{n-1} & c_n \\ & & & & d_n \end{pmatrix}. \quad (7.4.8)$$

By equating elements in A and LU it is verified that the upper diagonal in U equals that in A . The other elements in L and U are obtained by the recursion $d_1 = a_1$,

$$\gamma_k = b_k/d_{k-1}, \quad d_k = a_k - \gamma_k c_k, \quad k = 2 : n. \quad (7.4.9)$$

The elements γ_k and d_k can overwrite b_k and a_k , respectively. The solution to the system $Ax = f$ can then be computed by solving $Ly = f$ by $Ux = y$ by back- and forward substitution:

$$y_1 = f_1, \quad y_i = f_i - \gamma_i y_{i-1}, \quad i = 2 : n, \quad (7.4.10)$$

$$x_n = y_n/d_n, \quad x_i = (y_i - c_{i+1}x_{i+1})/d_i, \quad i = n-1 : -1 : 1. \quad (7.4.11)$$

The total number of flops is about $3n$ for the factorization and $2.5n$ for the solution. Note that the divisions in the substitution can be avoided if (7.4.9) is modified to compute d_k^{-1} . This may be more efficient since a division on many computers takes more time than a multiplication.

If A is tridiagonal then it is easily proved by induction that $\rho_n \leq 2$ with partial pivoting. This result is a special case of a more general result.

Theorem 7.4.5. [Bothe [75]]

If $A \in \mathbf{C}^{n \times n}$ has upper and lower bandwidth p then the growth factor in Gaussian elimination with partial pivoting satisfies

$$\rho_n \leq 2^{2p-1} - (p-1)2^{p-2}.$$

In particular, for a tridiagonal matrix ($p = 1$) we have $\rho_n \leq 2$.

When A is symmetric positive definite and tridiagonal (7.4.7)

$$A = \begin{pmatrix} a_1 & b_2 & & & \\ b_2 & a_2 & b_3 & & \\ \ddots & \ddots & \ddots & \ddots & \\ & b_{n-1} & a_{n-1} & b_n & \\ & & b_n & a_n & \end{pmatrix}, \quad (7.4.12)$$

we can write the factorization

$$A = LDL^T, \quad D = \text{diag}(d_1, \dots, d_n), \quad (7.4.13)$$

where L is as in (7.4.8). The factorization algorithm then reduces to $d_1 = a_1$,

$$\gamma_k = b_k/d_{k-1}, \quad d_k = a_k - \gamma_k b_k, \quad k = 2 : n. \quad (7.4.14)$$

Sometimes it is more convenient to write

$$A = U^T D^{-1} U, \quad D = \text{diag}(a_1, \dots, a_n).$$

Then U is given by (7.4.8) (with $c_k = b_k$), and the elements in U and D are computed from $d_1 = a_1$,

$$d_k = a_k - b_k^2/d_{k-1}, \quad k = 2 : n. \quad (7.4.15)$$

The recursion (7.4.9) for the LU factorization of a tridiagonal matrix is highly serial. An algorithm for solving tridiagonal systems, which has considerable inherent parallelism, is **cyclic reduction** also called **odd-even reduction**. This is the most preferred method for solving large tridiagonal systems on parallel computers.

The basic step in cyclic reduction is to eliminate all the odd unknowns to obtain a reduced tridiagonal system involving only even numbered unknowns. This process is repeated recursively until a system involving only a small order of unknowns remains. This is then solved separately and the other unknowns can then be

computed in a back substitution process. We illustrate this process on a tridiagonal system $Ax = f$ of order $n = 2^3 - 1 = 7$. If P is a permutation matrix such that $P(1, 2, \dots, 7) = (1, 3, 5, 7, 2, 4, 6)^T$ the transformed system $PAP^T(Px) = P^Tf$, will have the form

$$\left(\begin{array}{ccc|ccc} a_1 & & & c_2 & & \\ & a_3 & & b_3 & c_4 & \\ & & a_5 & b_5 & c_6 & \\ & & & b_7 & & \\ \hline b_2 & c_3 & & a_2 & & \\ b_4 & c_5 & & & a_4 & \\ b_6 & c_7 & & & & a_6 \end{array} \right) \left(\begin{array}{c} x_1 \\ x_3 \\ x_5 \\ x_7 \\ \hline x_2 \\ x_4 \\ x_6 \end{array} \right) = \left(\begin{array}{c} f_1 \\ f_3 \\ f_5 \\ \hline f_7 \\ f_2 \\ f_4 \\ f_6 \end{array} \right).$$

It is easily verified that after eliminating the odd variables from the even equations the resulting system is again tridiagonal. Rearranging these as before the system becomes

$$\left(\begin{array}{cc|c} a'_2 & a'_6 & c'_4 \\ & b'_6 & \\ \hline b'_4 & c'_6 & a'_4 \end{array} \right) = \left(\begin{array}{c} x_2 \\ x_6 \\ x_4 \end{array} \right) = \left(\begin{array}{c} f'_2 \\ f'_6 \\ f'_4 \end{array} \right).$$

After elimination we are left with one equation in one variable

$$a''_4 x_4 = f''_4.$$

Solving 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 we get the odd unknowns x_1, x_3, x_5, x_7 . Clearly this scheme can be generalized. For a system of dimension $n = 2^p - 1$, p steps are required in the reduction. Note, however, that 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 any dimension n .

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 there is some fill in during the elimination and about 2.7 times more operations are needed.

Example 7.4.4.

Consider the linear system $Ax = b$, where A is a symmetric positive definite tridiagonal matrix. Then A has positive diagonal elements and the symmetrically scaled matrix 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 has the 2×2 block form

$$\begin{pmatrix} I & F \\ F^T & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}, \quad (7.4.16)$$

with F lower bidiagonal. After block elimination the Schur complement system becomes

$$(I - F^T F)x = d - F^T c.$$

Here $I - F^T F$ is again a positive definite tridiagonal matrix. Thus, the process can be repeated recursively.

Boundary value problems, where the solution is subject to *periodic boundary conditions*, often lead to matrices of the form

$$B = \left(\begin{array}{cccc|c} a_1 & c_2 & & & b_1 \\ b_2 & a_2 & c_3 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-1} & a_{n-1} & c_n \\ c_1 & & b_n & & a_n \end{array} \right), \quad (7.4.17)$$

which are tridiagonal except for the two corner elements b_1 and c_1 . We now consider the real symmetric case, $b_i = c_i$, $i = 1 : n$. Partitioning B in 2×2 block form as above, we seek a factorization

$$B = \begin{pmatrix} A & u \\ v^T & a_n \end{pmatrix} = \begin{pmatrix} L & 0 \\ y^T & 1 \end{pmatrix} \begin{pmatrix} U & z \\ 0 & d_n \end{pmatrix}$$

where $u = b_1 e_1 + c_n e_{n-1}$, $v = c_1 e_1 + b_n e_{n-1}$. Multiplying out we obtain the equations

$$A = LU, \quad u = Lz, \quad v^T = y^T U, \quad a_n = y^T z + d_n$$

Assuming that no pivoting is required the factorization $A = LU$, where L and U are bidiagonal, is obtained using (7.4.9). The vectors y and z are obtained from the lower triangular systems

$$Lz = b_1 e_1 + c_n e_{n-1}, \quad U^T y = c_1 e_1 + c_n e_{n-1},$$

and $d_n = a_n - y^T z$. Note that y and z will be full vectors.

Cyclic reduction can be applied to systems $Bx = f$, where B has the tridiagonal form in (7.4.17). If n is even the reduced system obtained after eliminating the odd variables in the even equations will again have the form (7.4.17). For example, when $n = 2^3 = 8$ the reordered system is

$$\left(\begin{array}{ccc|ccc|c} a_1 & & & c_2 & & & b_1 \\ & a_3 & & b_3 & c_4 & & \\ & & a_5 & b_5 & c_6 & & \\ & & & b_7 & c_8 & & \\ \hline b_2 & c_3 & & a_2 & & & \\ b_4 & c_5 & & & a_4 & & \\ b_6 & c_7 & & & & a_6 & \\ c_1 & & b_8 & & & & a_8 \end{array} \right) \begin{pmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \\ \hline x_2 \\ x_4 \\ x_6 \\ x_8 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_3 \\ f_5 \\ f_7 \\ \hline f_2 \\ f_4 \\ f_6 \\ f_8 \end{pmatrix}.$$

If $n = 2^p$ the process can be applied recursively. After p steps one equation in a single unknown is obtained. Cyclic reduction here does not require extra storage and has also a slightly lower operation count than ordinary Gaussian elimination.

We finally consider the case when A is a **symmetric indefinite tridiagonal** matrix. It would be possible to use LU factorization with partial pivoting, but this destroys symmetry and gives no information about the inertia of A . Instead a block factorization $A = LDL^T$ can be computed using no interchanges as follows. Set $\sigma = \max_{1 \leq i \leq n} |a_{ij}|$ and $\alpha = (\sqrt{5} - 1)/2 \approx 0.62$. In the first stage we take a_{11} as pivot if $\sigma|a_{11}| \geq a_{21}^2$. Otherwise we take the 2×2 pivot

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

This factorization can be shown to be normwise backward stable and is a good way to solve such symmetric indefinite tridiagonal linear systems.

7.4.5 Inverses of Band Matrices

It is important to note that *the inverse A^{-1} of a band matrix in general has no zero elements*. This is true more generally. It has been shown that the inverse of an irreducible sparse matrix is structurally full. That means that for an irreducible sparsity pattern, it is always possible to find numerical values such that all entries in its inverse will be nonzero; see [175]. Hence, one should never attempt to *explicitly* compute the inverse of a band matrix. Even storing the elements in A^{-1} may be infeasible when the band matrix has large dimensions.

For the inverse of a tridiagonal matrix it is possible to find a simple representation. We first show that the lower triangular part of the inverse of an upper Hessenberg matrix has a very simple structure.

Theorem 7.4.6.

Let $H \in \mathbf{R}^{n \times n}$ 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

$$(H^{-1})_{ij} = p_i q_j, \quad i \geq j. \quad (7.4.18)$$

Proof. See Ikebe [347] \square

A tridiagonal matrix A is both lower and upper Hessenberg. Hence, if A is irreducible it follows that there are vectors x, y, p and q such that

$$S = (A^{-1})_{ij} = \begin{cases} u_i v_j, & i \leq j, \\ p_i q_j, & i < j. \end{cases} \quad (7.4.19)$$

or

$$S = \begin{pmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 & \cdots & u_1 v_n \\ p_2 q_1 & u_2 v_2 & u_2 v_3 & \cdots & u_2 v_n \\ p_3 q_1 & p_3 q_2 & u_3 v_3 & \cdots & u_3 v_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_n q_1 & p_n q_2 & p_n q_3 & \cdots & u_n v_n \end{pmatrix}.$$

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**; see Section 7.8.4.

The following algorithm has been suggested by N. J. Higham to compute the vectors x, y, p and q :

1. Compute the LU factorization of A .
2. Use the LU factorization to solve for the vectors y and z , where $A^T y = e_1$ and $A z = e_n$. Similarly, solve for p and r , where $A p = e_1$ and $A^T r = e_n$.
3. Set $q = p_n^{-1} r$ and $x = y_n^{-1} z$.

This algorithm is not foolproof and can fail because of overflow.

Example 7.4.5. Let A be a symmetric, positive definite tridiagonal matrix with elements $a_1 = 1$,

$$a_i = 2, \quad b_i = c_i = -1, \quad i = 2 : 5.$$

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 $x = p$, $y = q$, can be determined up to a scaling factor from the first and last columns of A^{-1} .

7.4.6 Envelope Methods

In some applications, one encounters matrices of band structure, where the bandwidth differs from row to row. For this class of matrices, called **variable-band matrices**, we define

$$f_i(A) = \min\{j \mid a_{ij} \neq 0\}, \quad l_j(A) = \min\{i \mid a_{ij} \neq 0\}. \quad (7.4.20)$$

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 $f_i(A) = l_i(A)$, $i = 1 : n$.

Definition 7.4.7.

The envelope (or profile) of A is the index set

$$\text{Env}(A) = \{(i, j) \mid f_i \leq j \leq i; \text{ or } l_j \leq i < j; \}. \quad (7.4.21)$$

The envelope of a symmetric matrix is defined by the envelope of its lower triangular part including the main diagonal.

For a variable band matrix it is convenient to use a storage scheme, in which the elements a_{ij} , $(i, j) \in \text{Env}(A)$ are stored and operated on. This storage scheme is convenient because only zeros inside the envelope will suffer fill during Gaussian elimination. An illustration of the envelope of a matrix is given below, where \times denotes a nonzero element and 0 a zero elements inside the envelope.

$$\begin{pmatrix} \times & \times \\ \times & \times & \times \\ & \times & 0 & \times & \times \\ \times & 0 & 0 & \times & \times & 0 \\ & & & \times & \times & 0 \\ & & & \times & 0 & 0 & \times & \times \\ & & & & \times & \times & \times \end{pmatrix}.$$

Theorem 7.4.8.

Assume that the triangular factors L and U of A exist. Then it holds that

$$\text{Env}(L + U) = \text{Env}(A),$$

i.e., the nonzero elements in L and U are contained in the envelope of A .

Proof. Assume that the theorem is true for matrices of order $n - 1$. Let $A \in \mathbf{R}^{n \times n}$ 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

$$L_{11}u_{1n} = a_{1n}, \quad U_{11}^Tl_{n1} = a_{n1}.$$

(cf. the bordering method Section 7.2.5). The solutions of these lower triangular systems are obtained by forward substitution. Therefore, if a_{1n} 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

Review Questions

- 4.1 Give an example of matrix multiplication by diagonals.

- 4.2** (a) If a is a column vector what is meant by $\text{diag}(a, k)$?
(b) If A is a square matrix what is meant by $\text{diag}(A, k)$?
- 4.3** (a) What do you know about the LU decomposition of a band matrix? Why should you avoid computing the inverse of a band matrix?
- 4.4** (a) Let $A \in \mathbf{R}^{n \times n}$ be a band matrix with upper bandwidth p and lower bandwidth q . Show how A can be efficiently stored when computing the LU factorization.
(b) Assuming that the LU factorization can be carried out without pivoting, what are the structures of the resulting L and U factors of A ?
(c) What can you say about the structure of the inverses of L and U ?
- 4.5** Let $A \in \mathbf{R}^{n \times n}$ be a band matrix with upper bandwidth p and lower bandwidth q . Assuming that the LU factorization of A can be carried out without pivoting, roughly how many operations are needed? You need only give the dominating term when $p, q \ll n$.
- 4.6** Give a bound for the growth ratio ρ_n in Gaussian elimination with partial pivoting, when the matrix A is: (a) Hessenberg; (b) tridiagonal.

Problems

- 4.1** (a) Let $A, B \in \mathbf{R}^{n \times n}$ have lower (upper) bandwidth r and s respectively. Show that the product AB has lower (upper) bandwidth $r + s$.
(b) An upper Hessenberg matrix H is a matrix with lower bandwidth $r = 1$. Using the result in (a) deduce that the product of H and an upper triangular matrix is again an upper Hessenberg matrix.

- 4.2** Show that an irreducible nonsymmetric tridiagonal matrix A can be written $A = DT$, where T is symmetric tridiagonal and $D = \text{diag}(d_k)$ is diagonal with elements

$$d_1 = 1, \quad d_k = \prod_{j=2}^k c_j/b_j, \quad k = 2 : n. \quad (7.4.22)$$

- 4.3** (a) Let $A \in \mathbf{R}^{n \times n}$ be a symmetric, tridiagonal matrix such that $\det(A_k) \neq 0$, $k = 1 : n$. Then the decomposition $A = LDL^T$ exists and can be computed by the formulas given in (7.4.14). Use this to derive a recursion formula for computing $\det(A_k)$, $k = 1 : n$.

- (b) Determine the largest n for which the symmetric, tridiagonal matrix

$$A = \begin{pmatrix} 2 & 1.01 & & & \\ 1.01 & 2 & 1.01 & & \\ & 1.01 & \ddots & \ddots & \\ & & \ddots & \ddots & 1.01 \\ & & & 1.01 & 2 \end{pmatrix} \in \mathbf{R}^{n \times n}$$

is positive definite.

- 4.4** (a) Show that for $a \geq 2$ it holds that $B = \mu LL^T$, where

$$B = \begin{pmatrix} \mu & -1 & & \\ -1 & a & -1 & \\ & -1 & \ddots & \ddots & \\ & & \ddots & a & -1 \\ & & & -1 & a \end{pmatrix}, \quad L = \begin{pmatrix} 1 & & & & \\ -\sigma & 1 & & & \\ & -\sigma & \ddots & & \\ & & \ddots & 1 & \\ & & & -\sigma & 1 \end{pmatrix},$$

and

$$\mu = a/2 \pm (a^2/4 - 1)^{1/2}, \quad \sigma = 1/\mu.$$

Note that L has constant diagonals.

- (b) Suppose we want to solve the system $Ax = b$, where the matrix A differs from B in the element (1,1),

$$A = B + \delta e_1 e_1^T, \quad \delta = a - \mu, \quad e_1^T = (1, 0, \dots, 0).$$

Show, using the Sherman–Morrison formula (7.1.26), that the solution $x = A^{-1}b$ can be computed from

$$x = y - \gamma L^{-T} f, \quad \gamma = \delta(e_1^T y)/(\mu + \delta f^T f)$$

where y and f satisfies $\mu LL^T y = b$, $Lf = e_1$.

- 4.5** Consider the symmetric tridiagonal matrix

$$A_n = \begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & 4 & 1 \\ & & & 1 & 4 \end{pmatrix}.$$

For $n = 20, 40$ use the Cholesky factorization of A_n and Higham's algorithm to determine vectors x and y so that $(A_n^{-1})_{ij} = x_i y_j$ for $i, j = 1 : n$. Verify that there is a range of approximately θ^n in the size of the components of these vectors, where $\theta = 2 + \sqrt{3}$.

- 4.5** (a) Write a 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 7.4.3.

(b) Write a function for computing the product $C = AB$ of two band matrices using the $w_1 w_2$ calls to the function in (a), where w_1 and w_2 are the bandwidth of A and B , respectively.

- 4.6** Derive expressions for computing δ_k , $k = 1 : n - 1$ and d_n in the factorization of the periodic tridiagonal matrix A in (7.4.17).

- 4.7** Let B be a symmetric matrix of the form (7.4.17). Show that

$$B = T + \sigma uu^T, \quad u = (1, 0, \dots, 0, -1)^T.$$

where T is a certain symmetric, tridiagonal matrix. What is σ and T . Derive an algorithm for computing L by modifying the algorithm (7.4.14).

4.8 Consider the symmetric tridiagonal matrix

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

- (a) Show that $A^2 = I$ and thus A is nonsingular and $A^{-1} = A$.
- (b) Show that A cannot be represented as

$$\begin{pmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \end{pmatrix}.$$

by two vectors $u, v \in \mathbf{R}^3$.

7.5 Perturbation and Error Analysis

7.5.1 Perturbation Analysis for Linear Systems

Consider a linear system $Ax = b$ where A is nonsingular and $b \neq 0$. The sensitivity of the solution x and the inverse A^{-1} to perturbations in A and b is of practical importance, since the matrix A and vector b are rarely known exactly. They may be subject to observational errors, or given by formulas which involve roundoff errors in their evaluation. (Even if they were known exactly, they may not be represented exactly as floating-point numbers in the computer.)

We start with deriving some results that are needed in the analysis.

Lemma 7.5.1.

Let $E \in \mathbf{R}^{n \times n}$ 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

$$\|(I - E)^{-1}\| \leq 1/(1 - \|E\|). \quad (7.5.1)$$

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 consider the identity $(I - E)(I - E)^{-1} = I$ or

$$(I - E)^{-1} = I + E(I - E)^{-1}.$$

Taking norms, we conclude

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

and (7.5.1) follows. \square

Corollary 7.5.2.

Assume that $\|B - A\| \|B^{-1}\| = \eta < 1$. Then it holds that

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

Proof. We have $\|A^{-1}\| = \|A^{-1}BB^{-1}\| \leq \|A^{-1}B\| \|B^{-1}\|$. The first inequality then follows by taking $E = B^{-1}(B - A) = I - B^{-1}A$ in Lemma 7.5.1. From the identity

$$A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1} \tag{7.5.2}$$

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

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

$$\delta x = (I + A^{-1}\delta A)^{-1}A^{-1}(-\delta Ax + \delta b), \tag{7.5.3}$$

which is the basic identity for the perturbation analysis.

In the simple case that $\delta A = 0$, we have $\delta x = A^{-1}\delta b$, which implies that $|\delta x| = |A^{-1}| |\delta b|$. Taking norms gives

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

Usually it is more appropriate to consider *relative* perturbations,

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A, x) \frac{\|\delta b\|}{\|b\|}, \quad \kappa(A, x) := \frac{\|Ax\|}{\|x\|} \|A^{-1}\|. \tag{7.5.4}$$

Here $\kappa(A, x)$ is the condition number with respect to perturbations in b . It is important to note that this implies that the size of the residual vector $r = b - A\bar{x}$ gives no direct indication of the *error* in an approximate solution \bar{x} . For this we need information about A^{-1} or the condition number $\kappa(A, x)$.

The inequality (7.5.4) 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

$$\kappa(A, x) \leq \|A\| \|A^{-1}\|, \tag{7.5.5}$$

which shows that a relative perturbation in the right hand side can at most be amplified by the factor $\|A\| \|A^{-1}\|$. but here *equality will hold only for rather special right hand sides* b . δb . Equation (7.5.5) motivates the following definition:

Definition 7.5.3. For a square nonsingular matrix A the **condition number** is

$$\kappa = \kappa(A) = \|A\| \|A^{-1}\|. \quad (7.5.6)$$

where $\|\cdot\|$ denotes any matrix norm.

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 2-norm we have using the SVD that $\|A\|_2 = \sigma_1$ and $\|A^{-1}\| = 1/\sigma_n$, where σ_1 and σ_n are the largest and smallest singular values of A . Hence,

$$\kappa_2(A) = \sigma_1/\sigma_n. \quad (7.5.7)$$

Note that $\kappa(\alpha A) = \kappa(A)$, i.e., the condition number is invariant under multiplication of A by a scalar. From the definition it follows easily also that

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

Further, for all p -norms it follows from the identity $AA^{-1} = I$ that

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

that is, the condition number is always greater or equal to one.

We now show that $\kappa(A)$ is also the condition number with respect to perturbations in A .

Theorem 7.5.4.

Consider the linear system $Ax = b$, where the matrix $A \in \mathbf{R}^{n \times n}$ is nonsingular. Let $(A + \delta A)(x + \delta x) = b + \delta b$ 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

$$\|\delta x\| \leq \frac{\|A^{-1}\|}{1 - \eta} (\|\delta A\| \|x\| + \|\delta b\|). \quad (7.5.8)$$

Proof. Taking norms in equation (7.5.3) gives

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

By assumption $\|A^{-1}\delta A\| \leq \|A^{-1}\| \|\delta A\| = \eta < 1$. Using Lemma 7.5.1 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 it holds that $\eta \ll 1$ and therefore $1/(1 - \eta) \approx 1$. Therefore, if upper bounds

$$\|\delta A\| \leq \epsilon_A \|A\|, \quad \|\delta b\| \leq \epsilon_b \|b\|, \quad (7.5.9)$$

for $\|\delta A\|$ and $\|\delta b\|$ are known, then for the normwise relative perturbation it holds that

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

Substituting $b = I$, $\delta b = 0$ and $x = A^{-1}$ in (7.5.8) and proceeding similarly from $(A + \delta A)(X + \delta X) = I$, we obtain the perturbation bound for $X = A^{-1}$

$$\frac{\|\delta X\|}{\|X\|} \lesssim \kappa(A) \frac{\|\delta A\|}{\|A\|}. \quad (7.5.10)$$

This shows that $\kappa(A)$ is indeed the condition number of A with respect to inversion.

Matrices with small condition numbers are said to be **well-conditioned**. For any real, orthogonal matrix Q it holds that

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

so Q is perfectly conditioned in the 2-norm. Furthermore, for any orthogonal P and Q , we have $\kappa(PAQ) = \kappa(A)$, for the 2-norm and the Frobenius norm. i.e., $\kappa(A)$ is invariant under orthogonal transformations. This important fact is one reason why orthogonal transformations play a central role in matrix computations.

When a linear system is ill-conditioned, i.e. $\kappa(A) \gg 1$, roundoff errors will in general cause a computed solution to have a large error. How large may 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} , then we can guarantee a (normwise) relative error in the solution $\leq 10^{-3}$ if $\kappa \leq 0.5 \cdot 10^4$. But to guarantee a (normwise) relative error in the solution $\leq 10^{-6}$ we need to have $\kappa \leq 5$.

Table 7.5.1. Condition numbers of Hilbert matrices of order ≤ 12 .

n	$\kappa_2(H_n)$	n	$\kappa_2(H_n)$
1	1	7	$4.753 \cdot 10^8$
2	19.281	8	$1.526 \cdot 10^{10}$
3	$5.241 \cdot 10^2$	9	$4.932 \cdot 10^{11}$
4	$1.551 \cdot 10^4$	10	$1.602 \cdot 10^{13}$
5	$4.766 \cdot 10^5$	11	$5.220 \cdot 10^{14}$
6	$1.495 \cdot 10^7$	12	$1.678 \cdot 10^{16}$

Example 7.5.1.

The Hilbert matrix H_n of order n is a matrix 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. In Table 7.6.1 approximate condition numbers of Hilbert matrices of order ≤ 12 , computed in IEEE double precision, are given. For $n > 12$ the Hilbert matrices are too ill-conditioned even for IEEE double precision! From a result by G. Szegö (see Gautschi [234, p. 34]) it follows that

$$\kappa_2(H_n) \approx \frac{(\sqrt{2} + 1)^{4(n+1)}}{2^{15/4}\sqrt{\pi n}} \sim e^{3.5n},$$

i.e., the condition numbers grow exponentially with n . Although the severe ill-conditioning exhibited by the Hilbert matrices is rare, moderately ill-conditioned linear systems do occur regularly in many practical applications!

The relative distance of a matrix A to the set of singular matrices in some norm is defined as

$$\text{dist}(A) := \min \left\{ \frac{\|\delta A\|}{\|A\|} \mid (A + \delta A) \text{ singular} \right\}. \quad (7.5.11)$$

The following theorem shows that the reciprocal of the condition number $\kappa(A)$ can be interpreted as a measure of the nearness to singularity of a matrix A .

Theorem 7.5.5 (Kahan [363]).

Let $A \in \mathbf{C}^{n \times n}$ 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

$$\text{dist}(A) = \kappa^{-1}(A). \quad (7.5.12)$$

Proof. If $(A + \delta A)$ is singular, then there is a vector $x \neq 0$ such that $(A + \delta A)x = 0$. Then, setting $y = Ax$, 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 $\|A^{-1}x\| = \|A^{-1}\|$. Set $y = A^{-1}x/\|A^{-1}\|$ so that $\|y\| = 1$ and $Ay = x/\|A^{-1}\|$. Let z be a dual vector to y so that (see Definition 7.1.12) $\|z\|_D\|y\| = z^H y = 1$, where $\|\cdot\|_D$ is the dual norm. Then $\|z\|_D = 1$, and if we take

$$\delta A = -xz^H/\|A^{-1}\|,$$

it follows that

$$(A + \delta A)y = Ay - xz^H y/\|A^{-1}\| = (x - x)/\|A^{-1}\| = 0.$$

Hence, $(A + \delta A)$ is singular. Further,

$$\|\delta A\| \|A^{-1}\| = \|xz^H\| = \max_{\|v\|=1} \|(xz^H)v\| = \|x\| \max_{\|v\|=1} |z^H v| = \|z\|_D = 1,$$

and thus $\|\delta A\| = 1/\|A^{-1}\|$, which proves the theorem. \square

The result in Theorem 7.5.5 can be used to get a *lower bound* for the condition number $\kappa(A)$. For the 2-norm the result follows directly from the SVD $A = U\Sigma V^H$. The closest singular matrix then equals $A + \delta A$, where

$$\delta A = -\sigma_n u_n v_n^H, \quad \|\delta A\|_2 = \sigma_n = 1/\|A^{-1}\|_2. \quad (7.5.13)$$

Sharper perturbation bounds for linear systems can be obtained by a componentwise perturbation analysis. Assume that

$$|\delta a_{ij}| \leq \omega e_{ij}, \quad |\delta b_i| \leq \omega f_i, \quad i, j = 1 : n,$$

for some $\omega \geq 0$, where $e_{ij} \geq 0$ and $f_i \geq 0$ are known. These bounds can be written as

$$|\delta A| \leq \omega E, \quad |\delta b| \leq \omega f, \quad (7.5.14)$$

where the absolute value of a matrix A and vector b is defined by

$$|A|_{ij} = (|a_{ij}|), \quad |b|_i = (|b_i|).$$

The partial ordering “ \leq ” for matrices A, B and vectors x, y , is to be interpreted componentwise.²² It is easy to show 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.

It is a well known fact that the computed solution of a triangular system $Tx = b$ often is far more accurate than predicted by the normwise condition number. This observation does not hold in general, for counter examples exist. However, it is true of many special kinds of triangular matrices arising from pivoted factorizations. Then it can be explained as due to an **artificial ill-conditioning** of the triangular matrix T . By this is meant that a row-scaling exist such that T is well-conditioned.

Example 7.5.2.

When solving a weighted least squares problem the following upper triangular matrix is obtained

$$U = \begin{pmatrix} 1 & 1 & 1 \\ & 10^{-3} & 10^{-3} \\ & & 10^{-6} \end{pmatrix}.$$

²²Note that $A \leq B$ in other contexts means that $B - A$ is positive semidefinite.

Then $\kappa(U) \approx 3 \cdot 10^6$. After a row scaling so that the diagonal elements are equal to unity, the matrix becomes well-conditioned $\kappa(DU) \approx 3$.

For deriving the componentwise bounds we need the following result.

Lemma 7.5.6.

Let $F \in \mathbf{R}^{n \times n}$ be a matrix for which $\|F\| < 1$. Then the matrix $(I - |F|)$ is nonsingular and

$$|(I - F)^{-1}| \leq (I - |F|)^{-1}. \quad (7.5.15)$$

Proof. The nonsingularity follows from Lemma 7.5.1. Using the identity $(I - F)^{-1} = I + F(I - F)^{-1}$ we obtain

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

from which the inequality (7.5.15) follows. \square

Theorem 7.5.7.

Consider the perturbed linear system $(A + \delta A)(x + \delta x) = b + \delta b$, where A is nonsingular. Assume that δA and δb satisfy the componentwise bounds in (7.5.14) and that

$$\omega \|A^{-1}|E\| < 1.$$

Then $(A + \delta A)$ is nonsingular and

$$\|\delta x\| \leq \frac{\omega}{1 - \omega \kappa_E(A)} \|A^{-1}|(E|x| + f)\|, \quad (7.5.16)$$

where $\kappa_E(A) = \|A^{-1}|E\|$.

Proof. Taking absolute values in (7.5.3) gives

$$|\delta x| \leq |(I + A^{-1}\delta A)^{-1}| |A^{-1}|(|\delta A||x| + |\delta b|). \quad (7.5.17)$$

Using Lemma 7.5.6 it follows from the assumption that the matrix $(I - |A^{-1}\delta A|)$ is nonsingular and from (7.5.17) we get

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

Using the componentwise bounds in (7.5.14) we get

$$|\delta x| \leq \omega (I - \omega |A^{-1}|E)^{-1} |A^{-1}|(E|x| + f), \quad (7.5.18)$$

provided that $\omega \kappa_E(A) < 1$. Taking norms in (7.5.18) and using Lemma 7.5.1 with $F = A^{-1}\delta A$ proves (7.5.16). \square

Taking $E = |A|$ and $f = |b|$ in (7.5.14) corresponds to bounds for the **componentwise relative errors** in A and b ,

$$|\delta A| \leq \omega |A|, \quad |\delta b| \leq \omega |b|. \quad (7.5.19)$$

For this special case Theorem 7.5.7 gives

$$\|\delta x\| \leq \frac{\omega}{1 - \omega \kappa_{|A|}(A)} \|A^{-1}(|A| |x| + |b|)\|, \quad (7.5.20)$$

where

$$\kappa_{|A|}(A) = \|A^{-1}\| |A|\|, \quad (7.5.21)$$

(or $\text{cond}(A)$) is the **Bauer–Skeel condition number** of the matrix A . Note that since $|b| \leq |A| |x|$, it follows that

$$\|\delta x\| \leq 2\omega \|A^{-1}\| |A| |x|\| + O(\omega^2) \leq 2\omega \kappa_{|A|}(A) \|x\| + O(\omega^2).$$

If $\hat{A} = DA$, $\hat{b} = Db$ where $D > 0$ is a diagonal scaling matrix, then $|\hat{A}^{-1}| = |A^{-1}| |D^{-1}|$. Since the perturbations scale similarly, $\delta\hat{A} = D\delta A$, $\delta\hat{b} = D\delta b$, it follows that

$$|\hat{A}^{-1}| |\delta\hat{A}| = |A^{-1}| |\delta A|, \quad |\hat{A}^{-1}| |\delta\hat{b}| = |A^{-1}| |\delta b|.$$

Thus, the bound in (7.5.20) and also $\kappa_{|A|}(A)$ are *invariant under row scalings*.

For the l_1 -norm and l_∞ -norm it holds that

$$\kappa_{|A|}(A) = \|A^{-1}\| |A|\| \leq \|A^{-1}\| \| |A|\| = \|A^{-1}\| |A|\| = \kappa(A),$$

i.e., the solution of $Ax = b$ is no more badly conditioned with respect to the componentwise relative perturbations than with respect to normed perturbations. On the other hand, it is possible for $\kappa_{|A|}(A)$ to be much smaller than $\kappa(A)$.

The analysis in Section 7.5.1 may not be adequate, when the perturbations in the elements of A or b are of different magnitude, as illustrated by the following example.

Example 7.5.3.

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 ∞ -norm we have $\|\delta b\|_\infty = 1$, $\|A^{-1}\|_\infty = 1$ (neglecting terms of order 10^{-8}). Theorem 7.5.4 then gives the gross overestimate $\|\delta x\|_\infty \leq 1$.

Multiplying the first equation by 10^{-4} , we obtain an equivalent system $\hat{A}x = \hat{b}$, where

$$\hat{A} = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix}, \quad \hat{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The perturbation in the vector b is now $|\delta\hat{b}| \leq 10^{-4}(1, 1)^T$, and from $\|\delta\hat{b}\|_\infty = 10^{-4}$, $\|(\hat{A})^{-1}\|_\infty = 1$, 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 we obtain a

well-conditioned system. How to scale linear systems for Gaussian elimination is a surprisingly intricate problem, which is further discussed in Section 7.5.5.

Consider the linear systems in Example 7.5.3. Neglecting terms of order 10^{-8} we have

$$|\hat{A}^{-1}| |\hat{A}| = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix} \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix} = \begin{pmatrix} 1 & 2 \cdot 10^{-4} \\ 2 \cdot 10^{-4} & 1 \end{pmatrix},$$

By the scaling invariance $\text{cond}(A) = \text{cond}(\hat{A}) = 1 + 2 \cdot 10^{-4}$ in the ∞ -norm. Thus, the componentwise condition number correctly reveals that the system is well-conditioned for componentwise small perturbations.

7.5.2 Backward Error Bounds

One common reason for poor accuracy in the computed solution is that the problem is ill-conditioned. But poor accuracy can also be caused by a poorly constructed algorithm. We say in general that an algorithm is **unstable** if it can introduce large errors in the computed solutions to a well-conditioned problem.

We consider in the following a finite algorithm with input data (a_1, \dots, a_r) , which by a sequence of arithmetic operations are transformed into the output data (w_1, \dots, w_s) . There are two basic forms of roundoff error analysis for such an algorithm, which are both useful:

- (i) In **forward** error analysis, one attempts to find bounds for the errors in the solution $|\bar{w}_i - w_i|$, $i = 1 : s$, where \bar{w}_i denotes the computed value of w_i . The main tool used in forward error analysis is the propagation of errors, as studied in Volume I, Section 2.4.2.
- (ii) In **backward** error analysis, one attempts to determine a modified set of data $a_i + \Delta a_i$ such that the computed solution \bar{w}_i is the *exact solution*, and give bounds for $|\Delta a_i|$. There may be an infinite number of such sets; in this case we seek to minimize the size of $|\Delta a_i|$. However, it can also happen, even for very simple algorithms, that no such set exists.

In backward error analysis no reference is made to the exact solution for the original data. In practice, when the data are known only to a certain accuracy, the “exact” solution may not be well defined. Then any solution whose backward error is smaller than the domain of uncertainty of the data may be considered to be a satisfactory result.

A frequently occurring backward error problem is the following. Suppose we are given an approximate solution y to a linear system $Ax = b$. We want to find out if y is the *exact solution to a nearby perturbed system* $(A + \Delta A)y = b + \Delta b$. To this end we define the normwise backward error of y as

$$\eta(y) = \min\{\epsilon \mid (A + \Delta A)y = b + \Delta b, \|\Delta A\| \leq \epsilon\|A\|, \|\Delta b\| \leq \epsilon\|b\|\}. \quad (7.5.22)$$

The following theorem tells us that the normwise backward error of y is small if the residual vector $b - Ay$ is small.

It is possible to derive a simple **a posteriori** bound for the backward error of a computed solution \bar{x} . These bounds are usually much sharper than a priori bounds and hold regardless of the algorithm used to compute \bar{x} .

Given \bar{x} , there are an infinite number of perturbations δA and δb for which $(A + \delta A)\bar{x} = b + \delta b$ holds. Clearly δA and δb must satisfy

$$\delta A\bar{x} - \delta b = b - A\bar{x} = r,$$

where $r = b - A\bar{x}$ is the residual vector corresponding to the computed solution. An obvious choice is to take $\delta A = 0$, and $\delta b = -r$. By instead taking $\delta b = 0$, we get for the 2-norm the following result: (Similar bounds for the l_1 -norm and l_∞ -norm are given in Problem 7.5.5.)

Theorem 7.5.8 (Rigal and Gaches [502]).

The normwise backward error of y is given by

$$\eta(y) = \frac{\|r\|}{\|A\| \|y\| + \|b\|}, \quad (7.5.23)$$

where $r = b - Ay$, and $\|\cdot\|$ is any consistent norm.

Let \bar{x} be a purported solution to $Ax = b$, and set $r = b - A\bar{x}$. Then if

$$\delta A = r\bar{x}^T / \|\bar{x}\|_2^2, \quad (7.5.24)$$

\bar{x} satisfies $(A + \delta A)\bar{x} = b$ and this has the smallest l_2 -norm $\|\delta A\|_2 = \|r\|_2 / \|\bar{x}\|_2$ of any such δA .

Proof. Clearly \bar{x} satisfies $(A + \delta A)\bar{x} = b$ if and only if $\delta A\bar{x} = r$. For any such δA it holds that $\|\delta A\|_2 \|\bar{x}\|_2 \geq \|r\|_2$ or $\|\delta A\|_2 \geq \|r\|_2 / \|\bar{x}\|_2$. For the particular δA given by (7.5.24) we get $\delta A\bar{x} = r\bar{x}^T\bar{x} / \|\bar{x}\|^2 = r$. From

$$\|r\bar{x}^T\|_2 = \sup_{\|y\|_2=1} \|r\bar{x}^T y\|_2 = \|r\|_2 \sup_{\|y\|_2=1} |\bar{x}^T y| = \|r\|_2 \|\bar{x}\|_2,$$

it follows that $\|\delta A\|_2 = \|r\|_2 / \|\bar{x}\|_2$ and hence the δA in (7.5.24) is of minimum l_2 -norm. \square

Define the **componentwise backward error** $\omega(y)$ of y by

$$\omega(y) = \min\{\epsilon \mid (A + \Delta A)y = b + \Delta b, |\Delta A| \leq \epsilon\|A\|, |\Delta b| \leq \epsilon|b|\}. \quad (7.5.25)$$

As the following theorem shows, there is a simple expression also for $\omega(y)$.

Theorem 7.5.9 (Oettli and Prager [453]).

Let $r = b - A\bar{x}$, E and f be nonnegative and set

$$\omega = \max_i \frac{|r_i|}{(E|\bar{x}| + f)_i}, \quad (7.5.26)$$

where $0/0$ is interpreted as 0 . If $\omega \neq \infty$, there is a perturbation δA and δb with

$$|\delta A| \leq \omega E, \quad |\delta b| \leq \omega f, \quad (7.5.27)$$

such that

$$(A + \delta A)\bar{x} = b + \delta b. \quad (7.5.28)$$

Moreover, ω is the smallest number for which such a perturbation exists.

Proof. From (7.5.26) we have

$$|r_i| \leq \omega(E|\bar{x}| + f)_i,$$

which implies that $r = D(E|\bar{x}| + f)$, where $|D| \leq \omega I$. It is then readily verified that

$$\delta A = DE \operatorname{diag}(\operatorname{sign}(\bar{x}_1), \dots, \operatorname{sign}(\bar{x}_n)), \quad \delta b = -Df$$

are the required backward perturbations.

Further, given perturbations δA and δb satisfying equations (7.5.27)–(7.5.28) for some ω we have

$$|r| = |b - A\bar{x}| = |\delta A\bar{x} - \delta b| \leq \omega(E|\bar{x}| + f).$$

Hence, $\omega \geq |r_i|/(E|\bar{x}| + f)_i$, which shows that ω as defined by (7.5.26) is optimal. \square

In particular, we can take $E = |A|$, and $f = |b|$ in Theorem 7.5.4, to get an expression for the componentwise relative backward error ω of a computed solution. This can then be used in (7.5.19) or (7.5.20) to compute a bound for $\|\delta x\|$.

Example 7.5.4.

Consider the linear system $Ax = b$, where

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix}, \quad b = \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix}.$$

Suppose that we are given the approximate solution $\bar{x} = (0.9911, -0.4870)^T$. The residual vector corresponding to \bar{x} is very small,

$$r = b - A\bar{x} = (-10^{-8}, 10^{-8})^T.$$

However, 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. (Compute the determinant of A and then the inverse A^{-1} !)

It should be emphasized that the system in this example is contrived. In practice one would be highly unfortunate to encounter such an ill-conditioned 2×2 matrix.²³

²³As 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”!

By means of backward error analysis it has been shown, even for many quite complicated matrix algorithms, that the computed results which the algorithm produces under the influence of roundoff error are the *exact* output data of a problem of the same type in which the relative change in data only are of the order of the unit roundoff u .

It is often possible to show that a small **backward error** in the following sense:

Definition 7.5.10.

An algorithm for solving a linear system $Ax = b$ is said to be (*normwise*) **backward stable** if, for any data $A \in \mathbf{R}^{n \times n}$ and $b \in \mathbf{R}^n$, there exist perturbation matrices and vectors δA and δb , such that the solution \bar{x} computed by the algorithm is the exact solution to a neighbouring system

$$(A + \delta A)\bar{x} = (b + \delta b), \quad (7.5.29)$$

where

$$\|\delta A\| \leq c_1(n)u\|A\|, \quad \|\delta b\| \leq c_2(n)u\|b\|.$$

A computed solution \bar{x} is called a (*normwise*) **stable solution** if it satisfies (7.5.29).

Since the data A and b usually are subject to errors and not exact, it is reasonable to be satisfied with the computed solution \bar{x} if the backward errors δA and δb are small in comparison to the uncertainties in A and b . As seen from (7.5.5), this does not mean that \bar{x} is close to the exact solution x .

7.5.3 Estimating Condition Numbers

The perturbation analysis has shown that the normwise relative perturbation in the solution x of a linear system can be bounded by

$$\|A^{-1}\| (\|\delta A\| + \|\delta b\|/\|x\|), \quad (7.5.30)$$

or, in case of componentwise analysis, by

$$\| |A^{-1}(|E|x| + f)| \| . \quad (7.5.31)$$

To compute these upper bounds exactly is costly since $2n^3$ flops are required to compute A^{-1} , even if the LU factorization of A is known (see Section 7.2.5). In practice, it will suffice with an *estimate* of $\|A^{-1}\|$ (or $\|A^{-1}\|$), which need not be very precise.

The first algorithm for condition estimation to be widely used was suggested by Cline, Moler, Stewart, and Wilkinson [114]. It is based on computing

$$y = (A^T A)^{-1} u = A^{-1} (A^{-T} u) \quad (7.5.32)$$

by solving the two systems $A^T w = u$ and $Ay = w$. A *lower bound* for $\|A^{-1}\|$ is then given by

$$\|A^{-1}\| \geq \|y\|/\|w\|. \quad (7.5.33)$$

If an LU factorization of A is known this only requires $O(n^2)$ flops. The computation of $y = A^{-T}w$ involves solving the two triangular systems

$$U^T v = u, \quad L^T w = v.$$

Similarly, the vector y and w are obtained by solving the triangular systems

$$Lz = w, \quad Uy = z,$$

For (7.5.33) to be a reliable estimate the vector u must be carefully chosen so that it reflects any possible ill-conditioning of A . Note that if A is ill-conditioned this is likely to be reflected in U , whereas L , being unit upper triangular, tends to be well-conditioned. To enhance the growth of v we take $u_i = \pm 1$, $i = 1 : n$, where the sign is chosen to maximize $|v_i|$. The final estimate is taken to be

$$1/\kappa(A) \leq \|w\|/(\|A\|\|y\|), \quad (7.5.34)$$

since then a singular matrix is signaled by zero rather than by ∞ and overflow is avoided. We stress that (7.5.34) always *underestimates* $\kappa(A)$. Usually the l_1 -norm is chosen 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. [164, 1979, pp. 11-13]. In practice it has been found that the LINPACK condition estimator seldom is off by a factor more than 10. However, counter examples can be constructed showing that it can fail. This is perhaps to be expected for any estimator using only $O(n^2)$ operations.

Equation (7.5.32) can be interpreted as performing one step of the inverse power method (see Section 10.3.3) on $A^T A$ using the special starting vector u . This is a simple method for computing the largest singular value $\sigma_1(A^{-1}) = \|A^{-1}\|_2$. An alternative to starting with the vector u is to use a *random* starting vector and perhaps carrying out several steps of inverse iteration with $A^T A$.

An alternative 1-norm condition estimator has been devised by Hager [300] and improved by Higham [325]. This estimates

$$\|B\|_1 = \max_j \sum_{i=1}^n |b_{ij}|,$$

assuming that Bx and $B^T x$ can be computed for an arbitrary vector x . It can also be used to estimate the infinity norm since $\|B\|_\infty = \|B^T\|_1$. It is based on the observation that

$$\|B\|_1 = \max_{x \in S} \|Bx\|_1, \quad S = \{x \in \mathbf{R}^n \mid \|x\|_1 \leq 1\}.$$

is the maximum of a convex function $f(x) = \|Bx\|_1$ over the convex set S . This implies that the maximum is obtained at an extreme point of S , i.e. one of the $2n$ points

$$\{\pm e_j \mid j = 1 : n\},$$

where e_j is the j th column of the unit matrix. If $y_i = (Bx)_i \neq 0$, $i = 1 : n$, then $f(x)$ is differentiable and by the chain rule the gradient is

$$\partial f(x) = \xi^T B, \quad \xi_i = \begin{cases} +1 & \text{if } y_i > 0, \\ -1 & \text{if } y_i < 0. \end{cases}$$

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

$$f(y) \geq f(x) + \partial f(x)(y - x) \quad \forall x, y \in \mathbf{R}^n.$$

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 S . We set $\partial f(x) = z^T$, where $z = B^T \xi$, and find an index j for which $|z_j| = \max_i |z_i|$. It can be shown that $|z_j| \leq z^T x$ then x is a local maximum. If this inequality is satisfied, then we stop. By the convexity of $f(x)$ and the fact that $f(e_j) = f(-e_j)$ we conclude that $f(e_j) > f(x)$. Replacing x by e_j we repeat the process. Since the estimates are strictly increasing each vertex of S is visited at most once. The iteration must therefore terminate in a finite number of steps. It has been observed that usually it terminates after just four iterations with the exact value of $\|B\|_1$.

We now show that the final point generated by the algorithm is a local maximum. Assume first that $(Bx)_i \neq 0$ for all i . Then $f(x) = \|Bx\|_1$ is linear in a neighborhood of x . It follows that x is a local maximum of $f(x)$ over S if and only if

$$\partial f(x)(y - x) \leq 0 \quad \forall y \in S.$$

If y is a vertex of 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 S . Since S is the convex hull of its vertices it follows that $\partial f(x)(y - x) \leq 0$, for all $y \in S$. Hence, x is a local maximum. In case some component of Bx is zero the above argument must be slightly modified; see Hager [300].

Algorithm 7.8. Hager's 1-norm estimator.

```

 $x = n^{-1}e$ 
repeat
     $y = Bx$ 
     $\xi = \text{sign}(y)$ 
     $z = B^T \xi$ 
    if  $\|z\|_\infty \leq z^T x$ 
         $\gamma = \|y\|_1$ ; quit
    end
     $x = e_j$ , where  $|z_j| = \|z\|_\infty$ 
end
```

To use this algorithm to estimate $\|A^{-1}\|_1 = \||A^{-1}| \|_1$, we take $B = A^{-1}$. In each iteration we are then required to solve the systems $Ay = x$ and $A^Tz = \xi$.

It is less obvious that Hager's estimator can also be used to estimate the componentwise relative error (7.5.31). The problem is then to estimate an expression of the form $\||A^{-1}|g\|_\infty$, for a given vector $g > 0$. Using a clever trick devised by Arioli, Demmel and Duff [12], this can be reduced to estimating $\|B\|_1$ where

$$B = (A^{-1}G)^T, \quad G = \text{diag}(g_1, \dots, g_n) > 0.$$

We have $g = Ge$ where $e = (1, 1, \dots, 1)^T$ and hence

$$\||A^{-1}|g\|_\infty = \||A^{-1}|Ge\|_\infty = \||A^{-1}G|e\|_\infty = \||A^{-1}G|\|_\infty = \|(A^{-1}G)^T\|_1,$$

where in the last step we have used that the ∞ norm is absolute (see Section 7.1.4). Since Bx and B^Ty can be found by solving linear systems involving A^T and A the work involved is similar to that of the LINPACK estimator. This together with ω determined by (7.5.26) gives an approximate bound for the error in a computed solution \bar{x} . Hager's condition estimator is used MATLAB.

We note that the unit lower triangular matrices L obtained from Gaussian elimination with pivoting are not arbitrary but their off-diagonal elements satisfy $|l_{ij}| \leq 1$. When Gaussian elimination without pivoting is applied to a row diagonally dominant matrix it gives a row diagonally dominant upper triangular factor $U \in \mathbf{R}^{n \times n}$ satisfying

$$|u_{ii}| \geq \sum_{j=i+1}^n |u_{ij}|, \quad i = 1 : n - 1. \quad (7.5.35)$$

and it holds that $\text{cond}(U) \leq 2n - 1$; (see [328, Lemma 8.8].

Definition 7.5.11. *For any triangular matrix T the comparison matrix is*

$$M(T) = (m_{ij}), \quad m_{ij} = \begin{cases} |t_{ii}|, & i = j; \\ -|t_{ij}|, & i \neq j; \end{cases}$$

7.5.4 Rounding Error Analysis of Gaussian Elimination

In the practical solution of a linear system of equations, rounding errors are introduced in each arithmetic operation and cause errors in the computed solution. In the early days of the computer era around 1946 many mathematicians were pessimistic about the numerical stability of Gaussian elimination. 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 revealed that this pessimism was unfounded. In practice Gaussian elimination with partial pivoting is a remarkably stable method and has become the universal algorithm for solving dense systems of equations.

The bound given in Theorem 7.2.5 is satisfactory only if the growth factor ρ_n is not too large, but this quantity is only known *after* the elimination has been completed. In order to obtain an *a priori bound* on ρ_n we use the inequality

$$|a_{ij}^{(k+1)}| = |a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)}| \leq |a_{ij}^{(k)}| + |a_{kj}^{(k)}| \leq 2 \max_k |\bar{a}_{ij}^{(k)}|,$$

valid If partial pivoting is employed. By induction this gives the upper bound $\rho_n \leq 2^{n-1}$, which is attained for matrices $A_n \in \mathbf{R}^{n \times n}$ of the form

$$A_4 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}. \quad (7.5.36)$$

Already for $n = 54$ we can have $\rho_n = 2^{53} \approx 0.9 \cdot 10^{16}$ and can lose all accuracy using IEEE double precision ($u = 1.11 \cdot 10^{-16}$). Hence, the *worst-case behavior of partial pivoting is very unsatisfactory*.

For complete pivoting, Wilkinson [609] has proved that

$$\rho_n \leq (n \cdot 2^1 3^{1/2} 4^{1/3} \dots n^{1/(n-1)})^{1/2} < 1.8 \sqrt{n} n^{\frac{1}{4} \log n},$$

and that this bound is not attainable. This bound is *much* smaller than that for partial pivoting, e.g., $\rho_{50} < 530$. It was long conjectured that $\rho_n \leq n$ for real matrices and complete pivoting. This was finally disproved in 1991 when a matrix of order 13 was found for which $\rho_n = 13.0205$. A year later a matrix of order 25 was found for which $\rho_n = 32.986$.

Although complete pivoting has a much smaller worst case growth factor than partial pivoting it is more costly. Moreover, complete (as well as rook) pivoting has the drawback that it cannot be combined with the more efficient blocked methods of Gaussian elimination (see Section 7.5.1). Fortunately from decades of experience and extensive experiments it can be concluded that substantial growth in elements using partial pivoting occurs only for a tiny proportion of matrices arising naturally. We quote Wilkinson [611, pp. 213–214].

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.

So far only a few exceptions to the experience related by Wilkinson have been reported. One concerns linear systems arising from a class of two-point boundary value problems, when solved by the shooting method. Another is the class of linear systems arising from a quadrature method for solving a certain Volterra integral equation. These examples show that Gaussian elimination with partial pivoting cannot be unconditionally trusted. When in doubt some safeguard like monitoring the element growth should be incorporated. Another way of checking and improving the reliability of Gaussian elimination with partial pivoting is iterative refinement, which is discussed in Section 7.5.6.

Why large element growth rarely occurs with partial pivoting is still not fully understood. Trefethen and Schreiber [577] have shown that for certain distributions of random matrices the average element growth was close to $n^{2/3}$ for partial pivoting.

We now give a componentwise roundoff analysis for the LU factorization of A . Note that all the variants given in Section 7.2 for computing the LU factorization of a matrix will essentially lead to the same error bounds, since each does the same operations with the same arguments. Also, note that since Gaussian elimination with pivoting is equivalent to Gaussian elimination without pivoting on a permuted matrix, we need not consider pivoting.

Theorem 7.5.12.

If the LU factorization of A runs to completion then the computed factors \bar{L} and \bar{U} satisfy

$$A + E = \bar{L}\bar{U}, \quad |E| \leq \gamma_n |\bar{L}| |\bar{U}|, \quad (7.5.37)$$

where $\gamma_n = nu/(1 - nu)$.

Proof. In the algorithms in Section 7.2.6) we set $l_{ii} = 1$ and compute the other elements in L and U from the equations

$$\begin{aligned} u_{ij} &= a_{ij} - \sum_{p=1}^{i-1} l_{ip} u_{pj}, \quad j \geq i; \\ l_{ij} &= \left(a_{ij} - \sum_{p=1}^{j-1} l_{ip} u_{pj} \right) / u_{jj}, \quad i > j, \end{aligned}$$

Using (7.1.90) 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 $\bar{l}_{ii} = l_{ii} = 1$. These inequalities may be written in matrix form \square

To prove the estimate the error in a computed solution \bar{x} of a linear system given in Theorem 7.5.12, we must also take into account the rounding errors performed in the solution of the two triangular systems $\bar{L}y = b$, $\bar{U}x = y$; see Theorem 7.2.1.

Theorem 7.5.13.

Let \bar{x} be the computed solution of the system $Ax = b$, using LU factorization and substitution. Then \bar{x} satisfies exactly

$$(A + \Delta A)\bar{x} = b, \quad (7.5.38)$$

where δA is a matrix, depending on both A and b , such that

$$|\Delta A| \leq \gamma_n (3 + \gamma_n) |\bar{L}| |\bar{U}|. \quad (7.5.39)$$

Proof. From Theorem 7.2.1 it follows that the computed \bar{y} and \bar{x} satisfy

$$(\bar{L} + \delta\bar{L})\bar{y} = b, \quad (\bar{U} + \delta\bar{U})\bar{x} = \bar{y},$$

where

$$|\delta\bar{L}| \leq \gamma_n |\bar{L}|, \quad |\delta\bar{U}| \leq \gamma_n |\bar{U}|. \quad (7.5.40)$$

Note that $\delta\bar{L}$ and $\delta\bar{U}$ depend upon b . Combining these results, it follows that the computed solution \bar{x} satisfies

$$(\bar{L} + \delta\bar{L})(\bar{U} + \delta\bar{U})\bar{x} = b,$$

and using equations (7.5.37)–(7.5.40) proves the backward error

$$|\Delta A| \leq \gamma_n (3 + \gamma_n) |\bar{L}| |\bar{U}|. \quad (7.5.41)$$

for the computed solution \bar{x} given in Theorem 7.5.12. \square

Note that although the perturbation δA depends upon b the *bound* on $|\delta A|$ is independent on 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 terms of order $O((nu)^2)$ in (7.5.41) it follows that

$$\|\delta A\|_\infty \leq 1.5n(n+1)\gamma_n\rho_n \|A\|_\infty. \quad (7.5.42)$$

By taking b to be the columns e_1, e_2, \dots, e_n of the unit matrix in succession we obtain the n computed columns of the inverse X of A . For the k th column we have

$$(A + \Delta A_r)\bar{x}_r = e_r,$$

where we have written ΔA_r to emphasize that the perturbation *is different for each column*. Hence, we cannot say that Gaussian elimination computes the exact inverse corresponding to some matrix $A + \Delta A$. We obtain the estimate

$$\|A\bar{X} - I\|_\infty \leq 1.5n(n+1)\gamma_n\rho_n \|A\|_\infty \|\bar{X}\|_\infty. \quad (7.5.43)$$

From $A\bar{X} - I = E$ it follows that $\bar{X} - A^{-1} = A^{-1}E$ and $\|\bar{X} - A^{-1}\|_\infty \leq \|A^{-1}\|_\infty \|E\|_\infty$, which together with (7.5.43) can be used to get a bound for the error in the computed inverse. We should stress again that we recommend that computing explicit inverses is avoided.

The residual for the computed solution satisfies $\bar{r} = b - A\bar{x} = \delta A\bar{x}$, and using (7.5.42) it follows that

$$\|\bar{r}\|_\infty \leq 1.5n(n+1)\gamma_n\rho_n \|A\|_\infty \|\bar{x}\|_\infty.$$

This shows the remarkable fact that *Gaussian elimination will give a small relative residual even for ill-conditioned systems*. Unless the growth factor is large the quantity

$$\|b - A\bar{x}\|_\infty / (\|A\|_\infty \|\bar{x}\|_\infty)$$

will in practice be of the order nu . It is important to realize that this property of Gaussian elimination is not shared by most other methods for solving linear systems. For example, if we first compute the inverse A^{-1} and then $x = A^{-1}b$ the residual \bar{r} may be much larger even if the accuracy in \bar{x} is about the same.

The error bound in Theorem 7.5.12 is instructive in that it shows that a particularly favourable case is when $|\bar{L}| |\bar{U}| = |\bar{L}\bar{U}|$. This is true when \bar{L} and \bar{U} are nonnegative. Then

$$|\bar{L}| |\bar{U}| = |\bar{L}\bar{U}| = |A + \Delta A| \leq |A| + |\bar{L}| |\bar{U}|,$$

and neglecting terms of order $O((nu)^2)$ we find that the computed \bar{x} satisfies

$$(A + \Delta A)\bar{x} = b, \quad |\Delta A| \leq 3\gamma_n |A|.$$

A class of matrices for which Gaussian elimination without pivoting gives positive factors L and U is the following.

Definition 7.5.14.

*A matrix $A \in \mathbf{R}^{n \times n}$ is called **totally positive** if the determinant of every square submatrix of A is positive.*

It is known (see de Boor and Pinkus [73]) that if A is totally positive, then it has an LU factorization with $L > 0$ and $U > 0$. Since the property of a matrix being totally positive is destroyed under row permutations, *pivoting should not be used when solving such systems*. Totally positive systems occur, e.g., in spline interpolation.

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 ∞ -norm in $O(n^2)$ operations without forming the matrix explicitly, since

$$\| |\bar{L}| |\bar{U}| \|_\infty = \| |\bar{L}| |\bar{U}| e \|_\infty = \| |\bar{L}| (|\bar{U}| e) \|_\infty.$$

This useful observation is due to Chu and George [108].

An error analysis for the Cholesky factorization of a symmetric positive definite matrix $A \in \mathbf{R}^{n \times n}$ is similar to that for LU factorization.

Theorem 7.5.15.

Suppose that the Cholesky factorization of a symmetric positive definite matrix $A \in \mathbf{R}^{n \times n}$ runs to completion and produces a computed factor \bar{R} and a computed solution \bar{x} to the linear system. Then it holds that

$$A + E_1 = \bar{L}\bar{U}, \quad |E_1| \leq \gamma_{n+1} |\bar{R}^T| |\bar{R}|, \tag{7.5.44}$$

and

$$(A + E_2)\bar{x} = b, \quad |E_2| \leq \gamma_{3n+1} |\bar{R}^T| |\bar{R}|. \tag{7.5.45}$$

Theorem 7.5.16 (J. H. Wilkinson [613]).

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 $2n^{3/2}u\kappa(A) < 0.1$. The computed \bar{L} satisfies

$$\bar{L}\bar{L}^T = A + E, \quad \|E\|_2 < 2.5n^{3/2}u\|A\|_2, \quad (7.5.46)$$

and hence is the exact Cholesky factor of a matrix close to A .

This is essentially the best normwise bounds that can be obtained, although Meinguet [432] has shown that for large n the constants 2 and 2.5 in Theorem 7.5.16 can be improved to 1 and 2/3, respectively.

In practice we can usually expect much smaller backward error in the computed solutions than the bounds derived in this section. It is appropriate to recall here a remark by J. H. Wilkinson (1974):

“All too often, too much attention is paid to the precise error bound that has been established. The main purpose of such an analysis is either to establish the essential numerical stability of an algorithm or to show why it is unstable and in doing so expose what sort of change is necessary to make it stable. The precise error bound is not of great importance.”

7.5.5 Scaling of Linear Systems

In a linear system of equations $Ax = b$ the i th equation may be multiplied by an arbitrary positive scale factor d_i , $i = 1 : n$, without changing the exact solution. In contrast, such a scaling will usually change the computed numerical solution. In this section we show that *a proper row scaling is important for Gaussian elimination with partial pivoting to give accurate computed solutions*, and give some rules for scaling.

We first show that *if the pivot sequence is fixed* then Gaussian elimination is unaffected by such scalings, or more precisely:

Theorem 7.5.17.

Denote by \bar{x} and \bar{x}' the computed solutions obtained by Gaussian elimination in floating-point arithmetic to the two linear systems of equations

$$Ax = b, \quad (D_r A D_c)x' = D_r b,$$

where D_r and D_c are diagonal scaling matrices. Assume that the elements of D_r and D_c 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 $\bar{x} = D_c \bar{x}'$, i.e., the components in the solution differ only in the exponents.

Proof. The proof follows by examination of the scaling invariance of the basic step in Algorithm 7.2

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - (a_{ik}^{(k)} a_{kj}^{(k)}) / a_{kk}^{(k)}.$$

□

This result shows that scaling will affect the accuracy of a computed solution only if it leads to a change in the selection of pivots. When partial pivoting is used the row scaling may affect the choice of pivots; indeed we can always find a row scaling which leads to *any predetermined pivot sequence*. However, since only elements in the pivotal column are compared, the choice of pivots is independent of the column scaling. Since a bad choice of pivots can give rise to large errors in the computed solution, it follows that for Gaussian elimination with partial pivoting to give accurate solutions *a proper row scaling is important*.

Example 7.5.5. The system $Ax = b$ in Example 7.5.3 has the solution $x = (0.9999, 0.9999)^T$, correctly rounded to four decimals. Partial pivoting will here select the element a_{11} as pivot. Using three-figure floating-point arithmetic, the computed solution becomes

$$\bar{x} = (0, 1.00)^T \quad (\text{Bad!}).$$

If Gaussian elimination 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

$$\bar{x} = (1.00, 1.00)^T \quad (\text{Good!}).$$

From the above discussion we conclude that the need for a proper scaling is of great importance for Gaussian elimination to yield good accuracy. As discussed in Section 7.5.1, an estimate of $\kappa(A)$ is often used to access the accuracy of the computed solution. If the perturbation bound (7.5.5) is applied to the scaled system $(D_r A D_c)x' = D_r b$, then we obtain

$$\frac{\|D_c^{-1}\delta x\|}{\|D_c^{-1}x\|} \leq \kappa(D_r A D_c) \frac{\|D_r \delta b\|}{\|D_r b\|}. \quad (7.5.47)$$

Hence, if $\kappa(D_r A D_c)$ can be made smaller than $\kappa(A)$, then it seems that we might expect a correspondingly more accurate solution. Note however that in (7.5.47) the perturbation in x is measured in the norm $\|D_c^{-1}x\|$, and we may only have found a norm in which the error *looks better!* We conclude that the column scaling D_c 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 $D_c = \text{diag}(c)$.

We now discuss the choice of row scaling. A scheme which is sometimes advocated is to choose $D_r = \text{diag}(d_i)$ so that each row in $D_r A$ has the same l_1 -norm, i.e.,

$$d_i = 1/\|a_i^T\|_1, \quad i = 1 : n. \quad (7.5.48)$$

(Sometimes the l_∞ -norm, of the rows are instead made equal.) This scaling, called **row equilibration**, can be seen to avoid the bad pivot selection in Example 7.5.3. However, suppose that through an unfortunate choice of physical units the solution x has components of widely varying magnitude. Then, as shown by the following example, row equilibration can lead to a *worse* computed solution than if no scaling is used!

Example 7.5.6. Consider the following system

$$A = \begin{pmatrix} 3 \cdot 10^{-6} & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 3 + 3 \cdot 10^{-6} \\ 6 \\ 2 \end{pmatrix} \quad |\epsilon| \ll 1$$

which has the exact solution $x = (1, 1, 1)^T$. The matrix A is *well-conditioned*, $\kappa(A) \approx 3.52$, but the choice of a_{11} as pivot leads to a disastrous loss of accuracy. Assume that through an unfortunate choice of units, the system has been changed into

$$\hat{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2 \cdot 10^6 & 2 & 2 \\ 10^6 & 2 & -1 \end{pmatrix},$$

with exact solution $\hat{x} = (10^{-6}, 1, 1)^T$. If now the rows are equilibrated, the system becomes

$$\tilde{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 2 \cdot 10^{-6} & 2 \cdot 10^{-6} \\ 1 & 2 \cdot 10^{-6} & -10^{-6} \end{pmatrix}, \quad \tilde{b} = \begin{pmatrix} 3 + 3 \cdot 10^{-6} \\ 6 \cdot 10^{-6} \\ 2 \cdot 10^{-6} \end{pmatrix}.$$

Gaussian elimination 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 $\hat{A}\bar{x} = \hat{b}$ 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!

A theoretical solution to the row scaling problem in Gaussian elimination with partial pivoting has been given by R. D. Skeel [533]. He shows a pivoting rule in Gaussian elimination should depend not only on the coefficient matrix but also on the solution. His scaling rule is instead based on minimizing a bound on the backward error that contains the quantity

$$\frac{\max_i(|D_r A||\bar{x}|)_i}{\min_i(|D_r A||\bar{x}|)_i}.$$

Scaling Rule: (R. D. Skeel) Assume that $\min_i(|A||x|)_i > 0$. Then scale the rows of A and b by $D_r = \text{diag}(d_i)$, where

$$d_i = 1/(|A||x|)_i, \quad i = 1 : n. \quad (7.5.49)$$

A measure of **ill-scaling** of the system $Ax = b$ is

$$\sigma(A, x) = \max_i(|A||x|)_i / \min_i(|A||x|)_i. \quad (7.5.50)$$

This scaling rule gives infinite scale factors for rows which 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.

Unfortunately scaling according to this rule is not in general practical, since 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 (7.5.49), which corresponds to row equilibration. Note that this assumption is violated in Example 7.5.6.

7.5.6 Iterative Refinement of Solutions

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 $r = b - A\bar{x}$ be the corresponding residual vector. Then one can attempt to improve the solution by solving the system $A\delta = r$ 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
   $r^{(s)} := b - Ax^{(s)}; \quad (\text{in precision } u_2 = \beta^{-t_2})$ 
  solve  $A\delta^{(s)} = r^{(s)}; \quad (\text{in precision } u_1 = \beta^{-t_1})$ 
   $x^{(s+1)} := x^{(s)} + \delta^{(s)};$ 
   $s := s + 1;$ 
end

```

When \bar{x} has been computed by Gaussian elimination this approach is attractive since 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 *extended precision* $t_2 > t_1$ for computing the residuals $r^{(s)}$; these are then rounded to single precision u_1 before solving for $\delta^{(s)}$. Since $x^{(s)}$, A and b are stored in single precision, only the accumulation of the inner

product terms are in precision u_2 , and no multiplications in extended 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 extended precision without much extra cost. If $f\ell_e$ denotes computation with extended precision and u_e the corresponding unit roundoff then the forward error bound for an inner product becomes

$$|f\ell(f\ell_e((x^T y)) - x^T y)| < u|x^T y| + \frac{n u_e}{1 - n u_e/2} (1 + u) |x^T||y|, \quad (7.5.51)$$

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. However, since computations in extended precision are machine dependent it has been difficult to make such programs portable.²⁴ The development of extended and mixed Precision BLAS (Basic Linear Algebra Subroutines) (see [410]) has made this feasible. A portable and parallelizable implementation of the mixed precision algorithm is described in [147].

In the ideal case that the rounding errors committed in computing the corrections can be neglected we have

$$x^{(s+1)} - x = (I - (\bar{L}\bar{U})^{-1}A)^s(\bar{x} - x).$$

where \bar{L} and \bar{U} denote the computed LU factors of A . Hence, the 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 Gaussian elimination using precision u_1 , then by Theorem 7.2.5 we have

$$\bar{L}\bar{U} = A + E, \quad \|E\|_\infty \leq 1.5n^2 \rho_n u_1 \|A\|_\infty,$$

and ρ_n is the growth factor. It follows that an upper bound for the initial rate of convergence is given by

$$\rho = \|(\bar{L}\bar{U})^{-1}E\|_\infty \leq n^2 \rho_n u_1 \kappa(A).$$

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 at which $\|\delta_c\|/\|x_c\| < \beta^{-t_1}$, when we may say that the solution is correct to working precision.

²⁴It was suggested that the IEEE 754 standard should require inner products to be precisely specified, but that did not happen.

2. In general the attainable accuracy is limited to $\min(k + t_2 - t_1, t_1)$ digits, which gives the case above when $t_2 \geq 2t_1$. Note that although the computed solution improves progressively with each iteration this is *not always reflected* in a corresponding decrease in the norm of the residual, which may stay about the same.

Iterative refinement can be used to compute a more accurate solution, in case A is ill-conditioned. However, 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 which is correct to only a few places.

In many descriptions of iterative refinement it is stressed that it is essential that the residuals are computed with a higher precision than the rest of the computation, for the process to yield a more accurate solution. This is true if the initial solution has been computed by a backward stable method, such as Gaussian elimination with partial pivoting, and provided that the system is well scaled. However, iterative refinement using single precision residuals, *can considerably improve the quality of the solution, for example, when the system is ill-scaled*, i.e., when $\sigma(A, x)$ defined by (7.5.50) is large, or if the pivot strategy has been chosen for the preservation of sparsity, see Section 7.5.

Example 7.5.7.

As an illustration consider again the badly scaled system in Example 7.5.5

$$\tilde{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 2 \cdot 10^{-6} & 2 \cdot 10^{-6} \\ 1 & 2 \cdot 10^{-6} & -10^{-6} \end{pmatrix}, \quad \tilde{b} = \begin{pmatrix} 3 + 3 \cdot 10^{-6} \\ 6 \cdot 10^{-6} \\ 2 \cdot 10^{-6} \end{pmatrix},$$

with exact solution $\tilde{x} = (10^{-6}, 1, 1)^T$. Using floating-point arithmetic with unit roundoff $u = 0.47 \cdot 10^{-9}$ the solution computed by Gaussian elimination with partial pivoting has only about four correct digits. From the residual $r = \tilde{b} - \tilde{A}\bar{x}$ we compute the Oettli–Prager backward error $\omega = 0.28810 \cdot 10^{-4}$. The condition estimate computed by (7.5.34) is $3.00 \cdot 10^6$, and wrongly indicates that the loss of accuracy should be blamed on ill-conditioning.

Using one step of iterative refinement using a single precision residual gives

$$\tilde{x} = \bar{x} + d = (0.999999997 \cdot 10^{-6} \quad 1.000000000 \quad 1.000000000)^T.$$

This should be recomputed using IEEE single and double precision! This is almost as good as for Gaussian elimination with column pivoting 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 the 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.

The following theorem states that if Gaussian elimination with partial pivoting is combined with iterative refinement in single precision then the resulting method will give a small relative backward error provided that the system is not too ill-conditioned or ill-scaled.

Theorem 7.5.18. (R. D. Skeel.)

As long as the product of $\text{cond}(A^{-1}) = \|A\|A^{-1}\|_\infty$ and $\sigma(A, x)$ is sufficiently less than $1/u$, where u is the machine unit, it holds that

$$(A + \delta A)x^{(s)} = b + \delta b, \quad |\delta a_{ij}| < 4n\epsilon_1|a_{ij}|, \quad |\delta b_i| < 4n\epsilon_1|b_i|, \quad (7.5.52)$$

for s large enough. Moreover, the result is often true already for $s = 2$, i.e., after only one improvement.

Proof. For exact conditions under which this theorem holds, see Skeel [534]. \square

As illustrated above, Gaussian elimination with partial or complete pivoting may not provide all the accuracy that the data deserves. How often this happens in practice is not known. 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 ω using (7.5.26) 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 2.
2. Perform one step of iterative refinement using the single precision residual r computed in step 1 to obtain the improved solution \tilde{x} . Compute the backward error $\tilde{\omega}$ of \tilde{x} . Repeat until the test on $\tilde{\omega}$ is passed.

7.5.7 Interval Matrix Computations

In order to treat multidimensional problems we introduce interval vectors and matrices. 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 $[a_{ij}] = [\underline{a}_{ij}, \bar{a}_{ij}]$, $i = 1 : m$, $j = 1 : n$,

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, which contains the set $\{Ax \mid A \in [A], x \in [x]\}$, but normally does not coincide with it. By the inclusion property it holds that

$$\{Ax \mid A \in [A], x \in [x]\} \subseteq [A][x] = \left(\sum_{j=1}^n [a_{ij}][x_j] \right). \quad (7.5.53)$$

In general, there will be an overestimation in enclosing the image with an interval vector caused by the fact that the image of an interval vector under a transformation in general is not an interval vector. This phenomenon, intrinsic to interval computations, is called the **wrapping effect**.

Example 7.5.8.

We have

$$A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad [x] = \begin{pmatrix} [0, 1] \\ [0, 1] \end{pmatrix}, \quad \Rightarrow \quad A[x] = \begin{pmatrix} [0, 2] \\ [-1, 1] \end{pmatrix}.$$

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 $x = (3/2 \ 1/2)^T$.)

The magnitude of an interval vector or matrix is interpreted componentwise and defined by

$$|[x]| = (|[x_1]|, |[x_2]|, \dots, |[x_n]|)^T,$$

where the magnitude of the components are defined by

$$|[a, b]| = \max\{|x| \mid x \in [a, b]\}, \quad (7.5.54)$$

The ∞ -norm of an interval vector or matrix is defined as the ∞ -norm of their magnitude,

$$\|[x]\|_\infty = \|[x]\|_\infty, \quad \|A\|_\infty = \|[A]\|_\infty. \quad (7.5.55)$$

Using interval arithmetic it is possible to compute strict enclosures of the product of two matrices. Note that this is needed also in the case of the product of two point matrices since rounding errors will in general occur. In this case we want to compute an interval matrix $[C]$ such that $fl(A \cdot B) \subset [C] = [C_{\inf}, C_{\sup}]$. The following simple code does that using two matrix multiplications:

$$\begin{aligned} \text{setround}(-1); \quad C_{\inf} &= A \cdot B; \\ \text{setround}(1); \quad C_{\sup} &= A \cdot B; \end{aligned}$$

Here and in the following we assume that the command $\text{setround}(i)$, $i = -1, 0, 1$ sets the rounding mode to $-\infty$, to nearest, and to $+\infty$, respectively.

We next consider the product of a point matrix A and an interval matrix $[B] = [B_{\inf}, B_{\sup}]$. In implementing 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. The following code, suggested by A. Neumeier, performs this task efficiently using four matrix multiplications:

$$\begin{aligned} A_- &= \min(A, 0); \quad A_+ = \max(A, 0); \\ \text{setround}(-1); \\ C_{\inf} &= A_+ \cdot B_{\inf} + A_- \cdot B_{\sup}; \\ \text{setround}(1); \\ C_{\sup} &= A_- \cdot B_{\inf} + A_+ \cdot B_{\sup}; \end{aligned}$$

(Note that the commands $A_- = \min(A, 0)$ and $A_+ = \max(A, 0)$ acts componentwise.) Rump [510] 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

$$\mathcal{X} = \{x \mid Ax = b, A \in [A], b \in [b]\}. \quad (7.5.56)$$

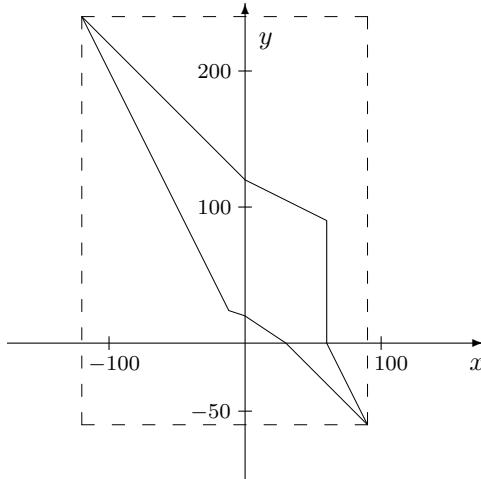


Figure 7.5.1. The solution set (solid line) and its enclosing hull (dashed line) for the linear system in Example 7.6.5.

Computing this solution set can be shown to be an intractable problem (NP-complete). Even for a 2×2 linear system this set may not be easy to represent.

Example 7.5.9. (E. Hansen [309, Chapter 4])

Consider a linear system with

$$[A] = \begin{pmatrix} [2, 3] & [0, 1] \\ [1, 2] & [2, 3] \end{pmatrix}, \quad [b] = \begin{pmatrix} [0, 120] \\ [60, 240] \end{pmatrix}. \quad (7.5.57)$$

The solution set \mathcal{X} in (7.5.56) is the star shaped region in Figure 7.7.1.

An enclosure of the solution set of an interval linear system can be computed by a generalization of Gaussian elimination adopted 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 [312]. For general systems this approach unfortunately tends to give interval sizes which grow exponentially during the elimination. For example, if $[x]$ and $[y]$ are intervals then in the 2×2 reduction

$$\begin{pmatrix} 1 & [x] \\ 1 & [y] \end{pmatrix} \sim \begin{pmatrix} 1 & [x] \\ 0 & [y] - [x] \end{pmatrix}.$$

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 [308]. In this an approximate inverse C is used to precondition the system. Assuming that an initial interval vector $[x^{(0)}]$ is known, such that $[x^{(0)}] \supseteq \mathcal{X}$ where \mathcal{X} is the solution set (7.5.56). An improved enclosure can then be obtained as follows:

By the inclusion property of interval arithmetic, for all $\tilde{A} \in [A]$ and $\tilde{b} \in [b]$ it holds that

$$[x^{(1)}] = \tilde{A}^{-1}\tilde{b} = C\tilde{b} + (I - C\tilde{A})\tilde{A}^{-1}\tilde{b} \in C[b] + (I - C[A])[x^{(0)}]$$

This suggests the iteration known as **Krawczyk's method**

$$[x^{(i+1)}] = \left(C[b] + (I - C[A])[x^{(i)}] \right) \cap [x^{(i)}], \quad i = 0, 1, 2, \dots, \quad (7.5.58)$$

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 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

$$[x^{(0)}] = C\text{mid}[b] + [-\beta, \beta]e, \quad e = (1, 1, \dots, 1),$$

with β 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 [510, 509] has developed a MATLABtoolbox INTLAB (INTerval LABoratory). This is very 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 $x_m = C\text{mid}[b]$; see Rump [510].

Example 7.5.10.

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 (7.5.58) 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

$$[E] = I - C[A] = \begin{pmatrix} [-0.01, 0.001] & [-0.001, 0.001] \\ [-0.001, 0.001] & [-0.01, 0.001] \end{pmatrix},$$

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

$$[X^{(i+1)}] = \left(I + E[X^{(i)}] \right) \cap [X^{(i)}], \quad i = 0, 1, 2, \dots.$$

converges rapidly in this case.

Review Questions

- 5.1** How is the condition number $\kappa(A)$ of a matrix A defined? How does $\kappa(A)$ relate to perturbations in the solution x to a linear system $Ax = b$, when A and b are perturbed? Outline roughly a cheap way to estimate $\kappa(A)$.
- 5.2** The result of a roundoff error analysis of Gaussian elimination can be stated in the form of a backward error analysis. Formulate this result. (You don't need to know the precise expression of the constants involved.)
- 5.3** (a) Describe the main steps in iterative refinement with extended precision for computing more accurate solutions of linear system.
 (b) Sometimes it is worthwhile to do a step of iterative refinement in using fixed precision. When is that?
-

Problems

- 5.1** (a) Compute the inverse A^{-1} of the matrix A in Problem 6.4.1 ???? and determine the solution x to $Ax = b$ when $b = (4, 3, 3, 1)^T$.
 (b) Assume that the vector b is perturbed by a vector δb such that $\|\delta b\|_\infty \leq 0.01$. Give an upper bound for $\|\delta x\|_\infty$, where δx is the corresponding perturbation in the solution.
 (c) Compute the condition number $\kappa_\infty(A)$, and compare it with the bound for the quotient between $\|\delta x\|_\infty/\|x\|_\infty$ and $\|\delta b\|_\infty/\|b\|_\infty$ which can be derived from (b).
- 5.2** Show that the matrix A in Example 7.5.4 has the inverse

$$A^{-1} = 10^8 \begin{pmatrix} 0.1441 & -0.8648 \\ -0.2161 & 1.2969 \end{pmatrix},$$

and that $\kappa_\infty = \|A\|_\infty \|A^{-1}\|_\infty = 2.1617 \cdot 1.5130 \cdot 10^8 \approx 3.3 \cdot 10^8$, which shows that the system is “perversely” ill-conditioned.

- 5.3** (Higham [328, 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 $\text{cond}(U^T) = 1+2/\epsilon$. This shows that a triangular system can be much worse conditioned than its transpose.

- 5.4** Let the matrix $A \in \mathbf{R}^{n \times n}$ be nonnegative, and solve $A^T x = e$, where $e = (1, 1, \dots, 1)^T$. Show that then $\|A^{-1}\|_1 = \|x\|_\infty$.
- 5.5** 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 error

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 $s_i = 1$, if $x_i \geq 0$; $s_i = -1$, if $x_i < 0$.

- 5.6** Use the result in Theorem 7.5.5 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 $\kappa_\infty(A) = 1.5(1 + |\epsilon|^{-1})$.)

- 5.7** Compute the LU factors of the matrix in (7.5.36).

7.6 Block Algorithms

7.6.1 BLAS and Linear Algebra Software

The first collection of high quality software was a series of algorithms written in Algol 60 that appeared in a handbook edited by Wilkinson and Reinsch [614]. This contains 11 subroutines for linear systems, least squares, and linear programming and 18 routines for the algebraic eigenvalue problem.

The collection LINPACK of Fortran subroutines for linear systems that followed contained several important innovations; see Dongarra et al. [164]. The routines were kept machine independent partly by performing as much of the computations as possible by calls to so-called Basic Linear Algebra Subprograms (BLAS) [400]. These identified frequently occurring vector operations in linear algebra 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 and β are scalars, x and y are vectors, Both single and double precision real and complex operations were provided. By carefully optimizing these BLAS for each specific computer, performance could be enhanced without sacrificing portability. LINPACK was followed by EISPACK, a collection of routines for the algebraic eigenvalue problem; see Smith et al. [536], Garbow et al. [229, 1977].

The original BLAS, now known as level 1 BLAS, were found to be unsatisfactory when vector computers were introduced in the 1980s. This brought about the development of level 2 BLAS for matrix-vector operations in 1988 [166]. The level 2

BLAS are operations involving one matrix and one or several vectors, for example,

$$\begin{aligned}y &= \alpha Ax + \beta y, \\y &= \alpha A^T x + \beta y, \\B &= \alpha xy^T + A, \\x &= Tx, \\x &= T^{-1}x,\end{aligned}$$

where x and y are vectors, A a matrix and T an upper or lower triangular matrix.

It was soon realized that level 2 BLAS did not provide codes with close to full efficiency, the reason being delay in getting data to the arithmetic processors. As a result time to access and write data will dominate over time spent on floating-point operations. The computer memory can be modeled as a linear array consisting of words addressed by integers, each containing a double precision number. The address space of the memory is typically 2^{32} representing 524 million double precision numbers. The memory is divided into blocks of contiguous words called pages. To handle such a large memory size modern computer architecture make use of a hierarchical memory structure, where large slow memories are backed up by fast smaller ones. Usually there are as many as four different levels. The main memory is supported by a backing store that usually is a disk. When a word is referenced the hardware determines where the page containing it is located. If the page is not in the main memory, then what is known as a **page fault** occurs and the page is swapped with one of the pages in main memory. This is done in such a way that the process is invisible to the programmer, one calls it virtual memory. Since reads and writes to disk is slow it is important that the code causes as few page faults as possible occurs.

To complicate things further, modern computers have a faster and smaller **cache memory** consisting of blocks smaller than a page, which are swapped in and out of main memory. Words 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 system is to avoid as much as possible data transfers between memories, registers and functional units, since these can be more costly than arithmetic operations on the data. Therefore, the operations have to be carefully sequenced. A key to achieving efficiency is locality of reference. Contiguous memory locations are likely to be located in the same page, whereas referencing locations far removed from another are likely to generate a page faults.

Level 2 BLAS involve $O(n^2)$ data, where n is the dimension of the matrix involved, and the same number of arithmetic operations. However, when RISC-type microprocessors with hierarchical memories were introduced, they failed to obtain adequate performance. Then level 3 BLAS were introduced in [165]. These were derived in a fairly obvious manner from some level 2 BLAS, by replacing the vectors x and y by matrices B and C ,

$$\begin{aligned}C &= \alpha AB + \beta C, \\C &= \alpha A^T B + \beta C,\end{aligned}$$

$$\begin{aligned} C &= \alpha AB^T + \beta C, \\ B &= \alpha TB, \\ B &:= \alpha T^{-1}B. \end{aligned}$$

Level 3 BLAS use $O(n^2)$ data but perform $O(n^3)$ arithmetic operations. Therefore, they give a surface-to-volume effect for the ratio of data movement to operations. This avoids excessive data movements between different parts of memory hierarchy. Level 3 BLAS are used in LAPACK, which achieves close to optimal performance on a large variety of computer architectures.

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 [358].

The LAPACK collection of subroutines [10] 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 much as possible of the computations by calls to so-called Level 2 and 3 BLAS. These enables the LAPACK routines to combine high performance with portable code and is also an aid to clarity, portability and modularity.

LAPACK is continually improved and updated and is available for free from <http://www.netlib.org/lapack95/>. 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

LAPACK95 is a Fortran 95 interface to the Fortran 77 LAPACK library. It is relevant for anyone who writes in the Fortran 95 language and needs reliable software for basic numerical linear algebra. It improves upon the original user-interface to the LAPACK package, taking advantage of the considerable simplifications that Fortran 95 allows. LAPACK95 Users' Guide 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. The LAPACK subroutines form the backbone of MATLAB which has simplified matrix computations tremendously.

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 [66].

7.6.2 Block and Partitioned Algorithms

For linear algebra algorithms to achieve high performance on modern computer architectures they need to be rich in matrix-matrix multiplications. This has the effect of reducing data movement, since a matrix-matrix multiplication involves $O(n^2)$ data and does $O(n^3)$ flops. The different versions of LU factorization we have considered so far correspond to different ways of ordering the loops and use level 1 and level 2 BLAS. In order to achieve 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 **partitioned algorithm** is a scalar algorithm in which the operations have been grouped and reordered into matrix operations. Such algorithms have the same stability properties as their scalar counterparts. A **block algorithm** is a generalization of a scalar algorithm where the scalar elements of the matrix has been replaced by matrices and scalar operations exchanged for matrix operations. Although such a block algorithm may have good numerical stability properties this cannot be taken for granted, since in general it does not perform the same arithmetic operations as the corresponding scalar algorithm.

As a first example, consider computing the inverse of a block lower triangular matrix

$$L = \begin{pmatrix} L_{11} & & & \\ L_{21} & L_{22} & & \\ \vdots & \vdots & \ddots & \\ L_{n,1} & L_{n,2} & \cdots & L_{nn} \end{pmatrix}, \quad (7.6.1)$$

where the diagonal blocks L_{ii} , $i = 1 : 2$, are assumed to be nonsingular but not necessarily lower triangular. It is easily verified that the inverse will also be block lower triangular,

$$L^{-1} = \begin{pmatrix} Y_{11} & & & \\ Y_{21} & Y_{22} & & \\ \vdots & \vdots & \ddots & \\ Y_{n,1} & Y_{n,2} & \cdots & Y_{nn} \end{pmatrix}, \quad (7.6.2)$$

In Section 7.1.2 we showed that the inverse in the 2×2 case is

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}.$$

For $n > 2$ the blocks in the inverse can be computed a block column at a time from a straightforward extension of the scalar algorithm (7.2.40). Identifying blocks in

the j th block column of the matrix equation $LY = I$, we get

$$L_{jj}Y_{jj} = I, \quad L_{ii}Y_{ij} = -\sum_{k=j}^{i-1} L_{ik}Y_{kj}, \quad j = 1 : n, \quad i = j + 1 : n. \quad (7.6.3)$$

These equations can be solved for Y_{jj}, \dots, Y_{nj} , by the scalar algorithms described in Section 7.2. The main arithmetic work will take place in the matrix-matrix multiplications $L_{ik}Y_{kj}$. This is an example of a true block algorithm. Such algorithms are obtained by substituting in a scalar algorithm operations on blocks of partitioned matrices regarded as non-commuting scalars.

In the special case that the diagonal blocks of L are lower triangular matrices, the equations in (7.6.3) can be solved by back substitution. Then all blocks Y_{ii} , $i = 1 : n$, are also lower triangular and the algorithm is just a rearrangement of a scalar algorithm, that is, a partitioned algorithm.

In Section 7.1.2 we gave, using slightly different notations, the block LU factorization

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & S \end{pmatrix}, \quad (7.6.4)$$

for a 2×2 block matrix, with square diagonal blocks. Here $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is the Schur complement. Note that the diagonal blocks in the block lower triangular factor in (7.6.4) are the identity matrix. This is a true block algorithm. In a partitioned LU factorization algorithm, the LU factors should have the form

$$A = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix},$$

where L_{11} , L_{22} are unit lower triangular and U_{11} , U_{22} upper triangular. Such a factorization can be computed as follows. First compute the scalar LU factorization $A_{11} = L_{11}U_{11}$, and then compute

$$L_{21} = A_{21}U_{11}^{-1}, \quad U_{12} = L_{11}^{-1}A_{12}, \quad S_{22} = A_{22} - L_{21}U_{12}.$$

Finally, compute the scalar factorization $S_{22} = L_{22}U_{22}$.

In the general case a partitioned algorithm for the LU factorization of a block matrix

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{pmatrix}, \quad (7.6.5)$$

with square diagonal blocks. Let L and U be partitioned conformally with A . Equating blocks in the product $A = LU$, we obtain, assuming that all inverses exist, the following block LU algorithm:

Algorithm 7.9. Partitioned LU Factorization.

```

for  $k = 1 : N$ 
     $S_{kk} = A_{kk} - \sum_{p=1}^{k-1} L_{kp}U_{pk};$ 
     $S_{kk} = L_{kk}U_{kk}$ 
    for  $j = k + 1 : N$ 
         $L_{jk} = \left( A_{jk} - \sum_{p=1}^{k-1} L_{jp}U_{pk} \right) U_{kk}^{-1};$ 
    end
    for  $j = 1 : k - 1$ 
         $U_{jk} = L_{kk}^{-1} \left( A_{jk} - \sum_{p=1}^{k-1} L_{jp}U_{pj} \right);$ 
    end
end

```

Here the LU-decompositions $S_{kk} = L_{kk}U_{kk}$ of the modified diagonal blocks are computed by a scalar LU factorization algorithm. Note that the dominating part of the work is performed in matrix-matrix multiplications. The inverse of the triangular matrices L_{kk}^{-1} and U_{kk}^{-1} are *not* formed but the off-diagonal blocks U_{kj} and L_{jk} (which in general are full matrices) are computed by triangular solves. Pivoting can be used in the factorization of the diagonal blocks. As described the algorithm does not allow for row interchanges between blocks. This point is addressed in the next section.

There are many possible ways of sequencing the computations corresponding to those outlined Section 7.2.6. The partitioned algorithm above computes in the k th major step the k th block column of L and U . In this variant at step k only the k th block column of A is accessed, which is advantageous from the standpoint of data access. In another variant in the k th major step the k th block row of U and the k th block column of L are computed.

A block LU factorization algorithm differs from the partitioned algorithm above in that the lower block triangular matrix L has diagonal blocks equal to unity 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 block tridiagonal matrices that are *block diagonally dominant*.

Definition 7.6.1 ((Demmel et al. [149])).

A general matrix $A \in \mathbf{R}^{n \times n}$ is said to be block diagonally dominant by columns, with respect to a given partitioning, if it holds i.e.

$$\|A_{jj}^{-1}\|^{-1} \geq \sum_{i \neq j} \|A_{ij}\|, \quad j = 1 : n. \quad (7.6.6)$$

It is said to be strictly block diagonally dominant if (7.6.6) holds with strict inequality.

A is said to be (strictly) block diagonally dominant by rows, if A^T is (strictly) diagonally dominant by columns.

Note that for block size 1 the usual property of (point) diagonal dominance is obtained. For the 1 and ∞ -norms diagonal dominance does not imply block diagonal dominance. Neither does and the reverse implications hold.

Analogous to the Block LU Algorithm in Section 7.6.2 block versions of the Cholesky algorithm can be developed. If A is partitioned into $N \times N$ blocks with square diagonal blocks, then using a block columnwise order, we obtain the following algorithm:

Algorithm 7.10. *Partitioned Cholesky Algorithm.*

```

for  $j = 1 : N$ 
     $S_{jj} = A_{jj} - \sum_{k=1}^{j-1} R_{jk}^T R_{jk};$ 
     $S_{jj} = R_{jj}^T R_{jj}$ 
    for  $i = j + 1 : N$ 
         $R_{ij}^T = \left( A_{ij} - \sum_{k=1}^{j-1} R_{ik}^T R_{jk} \right) (R_{jj})^{-1};$ 
    end
end

```

Note that the diagonal blocks R_{jj} are obtained by computing the Cholesky factorizations of matrices of smaller dimensions. The right multiplication with $(R_{jj})^{-1}$ in the computation of R_{ij}^T is performed by solving the triangular equations of the form $R_{jj}^T R_{ij} = S^T$. The matrix multiplications dominate the arithmetic work in the block Cholesky algorithm.

Note that in some cases a block factorization may exists even when a partitioned factorization does not. For example, a symmetric indefinite matrix may not have an LDL^T factorization unless we allow 2×2 diagonal blocks in D .

In deriving the block LU and Cholesky algorithms we assumed that the block sizes were determined in advance. However, this is by no means necessary. A more flexible way is to advance the computation by deciding at each step the size of the current pivot block. The corresponding dynamically partitioned algorithms uses a 3×3 block structure, but the partitioning changes after each step.

Suppose that an LU factorization of the first n_1 columns has been computed. We can write the result in the form

$$P_1 A = \begin{pmatrix} L_{11} & & \\ L_{21} & I & \\ L_{31} & 0 & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & \tilde{A}_{22} & \tilde{A}_{23} \\ & \tilde{A}_{32} & \tilde{A}_{33} \end{pmatrix}. \quad (7.6.7)$$

where P_1 is a permutation matrix and $L_{11}, U_{11} \in \mathbf{R}^{n_1 \times n_1}$ has been obtained. The

remaining $n-n_1$ columns are partitioned into blocks of n_2 and $n-(n_1+n_2)$ columns.

To advance the factorization an LU factorization with row pivoting is performed

$$P_2 \begin{pmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{pmatrix} = \begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} U_{22}, \quad (7.6.8)$$

where $L_{22}, U_{22} \in \mathbf{R}^{n_2 \times n_2}$. The permutation matrix P_2 has to be applied also to

$$\begin{pmatrix} \tilde{A}_{23} \\ \tilde{A}_{33} \end{pmatrix} := P_2 \begin{pmatrix} \tilde{A}_{23} \\ \tilde{A}_{33} \end{pmatrix}, \quad \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} := P_2 \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix}.$$

We then solve for U_{23} and update A_{33} using

$$L_{22}U_{23} = \tilde{A}_{23}, \quad \tilde{A}_{33} = A_{33} - L_{32}U_{23}.$$

The factorization has now been advanced one step to become

$$P_2 P_1 A = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & A_{33} \end{pmatrix}.$$

We can now repartition so that the first two block-columns in L are joined into a block of n_1+n_2 columns and similarly the first two block-rows in U joined into one block of n_1+n_2 rows. The blocks I and A_{33} in L and U are partitioned into 2×2 block matrices and we advance to the next block-step. This describes the complete algorithm since we can start the algorithm by taking $n_1 = 0$.

The above algorithm is sometimes called **right-looking**, referring to the way in which the data is accessed. The corresponding **left-looking** algorithm goes as follows. Assume that we have computed the first block column in L and U in the factorization (7.6.7). To advance the factorization we solve the triangular system $L_{11}U_{12} = A_{12}$ to obtain U_{12} and compute

$$\begin{pmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{pmatrix} = \begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix} - \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} U_{12},$$

We then compute the partial LU factorization (7.6.8) and replace

$$\begin{pmatrix} \tilde{A}_{23} \\ \tilde{A}_{33} \end{pmatrix} = P_2 \begin{pmatrix} A_{23} \\ A_{33} \end{pmatrix}, \quad \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} := P_2 \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix}.$$

The factorization has now been advanced one step to become

$$P_2 P_1 A = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & A_{13} \\ & U_{22} & A_{23} \\ & & A_{33} \end{pmatrix}.$$

Note that in this version the blocks in the last block column of A are referenced only in the pivoting operation, but this can be postponed.

Block LU factorizations appears to have been first proposed for **block tridiagonal** matrices, which often arise from the discretization of partial differential

equations. For a symmetric positive definite matrix the recursion (7.4.15) is easily generalized to compute the following block-factorization:

$$A = U^T D^{-1} U, \quad D = \text{diag}(\Sigma_1, \dots, \Sigma_n),$$

of a symmetric positive definite **block-tridiagonal** matrix with square diagonal blocks. We obtain

$$A = \begin{pmatrix} D_1 & A_2^T & & \\ A_2 & D_2 & A_3^T & \\ & A_3 & \ddots & \ddots & \\ & & \ddots & \ddots & A_N^T \\ & & & A_N & D_N \end{pmatrix}, \quad U^T = \begin{pmatrix} \Sigma_1 & & & \\ A_2 & \Sigma_2 & & \\ & A_3 & \ddots & \\ & & \ddots & \ddots & \\ & & & A_N & \Sigma_N \end{pmatrix},$$

where

$$\Sigma_1 = D_1, \quad \Sigma_k = D_k - A_k \Sigma_{k-1}^{-1} A_k^T, \quad k = 2 : N. \quad (7.6.9)$$

To perform the operations with Σ_k^{-1} , $k = 1 : N$ the Cholesky factorization of these matrices are computed by a scalar algorithm. After this factorization has been computed the solution of the system

$$Ax = U^T D^{-1} Ux = b$$

can be obtained by block forward- and back substitution $U^T z = b$, $Ux = Dz$.

Note that the blocks of the matrix A may again have band-structure, which should be taken advantage of! A similar algorithm can be developed for the unsymmetric block-tridiagonal case.

For block tridiagonal matrices the following result is known:

Theorem 7.6.2 (Varah [592]).

Let the matrix $A \in \mathbf{R}^{n \times n}$ be block tridiagonal and have the block LU factorization $A = LU$, where L and U are block bidiagonal, and normalized so that $U_{i,i+1} = A_{i,i+1}$. Then if A is block diagonally dominant by columns

$$\|L_{i,i-1}\| \leq 1, \quad \|U_{i,i}\| \leq \|A_{i,i}\| + \|A_{i-1,i}\|. \quad (7.6.10)$$

If A is block diagonally dominant by rows

$$\|L_{i,i-1}\| \leq \frac{\|A_{i-1,i}\|}{\|A_{i,i-1}\|}, \quad \|U_{i,i}\| \leq \|A_{i,i}\| + \|A_{i-1,i}\|. \quad (7.6.11)$$

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.

7.6.3 A Recursive Fast Matrix Multiply

Recursive algorithms introduces an automatic blocking, where the block size changes during the execution and can targets several different levels of memory hierarchy; Gustavson [297].

As a first example of a recursive matrix algorithm we consider the algorithm for fast matrix multiplication of Strassen [558]. This 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

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

Then, as can be verified by substitution, the product C can be computed using the following formulas:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 + P_3 - P_2 + P_6 \end{pmatrix}, \quad (7.6.12)$$

where

$$\begin{aligned} P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}), & P_2 &= (A_{21} + A_{22})B_{11}, \\ P_3 &= A_{11}(B_{12} - B_{22}), & P_4 &= A_{22}(B_{21} - B_{11}), \\ P_5 &= (A_{11} + A_{12})B_{22}, & P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}), \\ P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}). \end{aligned}$$

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 (n^3) than addition (n^2) this will lead to a saving in operations.

If Strassen's algorithm is used recursively to multiply two square matrices of dimension $n = 2^k$, then the number of multiplications is reduced from n^3 to $n^{\log_2 7} = n^{2.807\dots}$. (The number of additions is of the same order.) Even with just one level of recursion Strassen's method is faster in practice when n is larger than about 100, see Problem 7.6.4. However, there is some loss of numerical stability compared to conventional matrix multiplication, see Higham [328, Ch. 23]. By using the block formulation recursively, and Strassen's method for the matrix multiplication it is possible to perform the LU factorization. In practice recursion is only performed down to some level at which the gain in arithmetic operations is outweighed by overheads in the implementation.

The following MATLAB program shows how Strassen's algorithm can be implemented in a simple but efficient recursive MATLAB program. The program uses the fast matrix multiplication as long as n is a power of two and $n > n_{min}$ when it switches to standard matrix multiplication.

Algorithm 7.11. Recursive Matrix Multiplication by Strassen's Algorithm.

```

function C = smult(A,B,nmin);
% SMULT computes the matrix product A*B of two square
% matrices using Strassen's algorithm
n = size(A,1);
if rem(n,2) == 0 & n > nmin
    Recursive multiplication
    n = n/2; u = 1:n; v = n+1:2*n;
    P1 = smult(A(u,u) + A(v,v),B(u,u) + B(v,v),nmin);
    P2 = smult(A(v,u) + A(v,v),B(u,u),nmin);
    P3 = smult(A(u,u),B(u,v) - B(v,v),nmin);
    P4 = smult(A(v,v),B(v,u) - B(u,u),nmin);
    P5 = smult(A(u,u) + A(u,v),B(v,v),nmin);
    P6 = smult(A(v,u) - A(u,u),B(u,u) + B(u,v),nmin);
    P7 = smult(A(u,v) - A(v,v),B(v,u) + B(v,v),nmin);
    C(u,u) = P1 + P4 - P5 + P7;
    C(u,v) = P3 + P5;
    C(v,u) = P2 + P4;
    C(v,v) = P1 + P3 - P2 + P6;
else
    C = A*B;
end

```

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 transversed in preorder during the execution; see Knuth [378, Sec. 2.3]. Figure 7.5.1 shows the tree and the order in which the nodes are visited for $n = 8$.

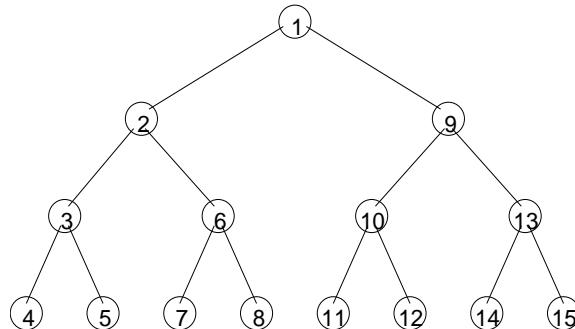


Figure 7.6.1. Binary tree with nodes in preorder.

7.6.4 Recursive Cholesky and LU Factorizations

From 2005 almost all computer manufacturers have changed their computer architecture to multi-core. The number of processors is expected to double about every 15 month. These new designs lead to poor performance for traditional block methods in matrix computations. For this new architectures recursively blocked matrix algorithms and data structures have several advantages.

By using recursive blocking in matrix factorizations we obtain algorithms which express the computations entirely in level 3 BLAS matrix-matrix operations. In this section we exemplify this by looking at the Cholesky and LU factorizations. The idea can applies more generally to other matrix algorithms.

We now develop a recursive algorithm for the Cholesky factorization. of a symmetric positive definite matrix $A \in \mathbf{R}^{n \times n}$. Then no pivoting need to be performed. We assume that A is stored in the upper triangular part of a square matrix. The matrix is 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

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{12}^T & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{12} \\ 0 & L_{22}^T \end{pmatrix}. \quad (7.6.13)$$

gives the following matrix equations for computing the three nonzero blocks in the Cholesky factor L :

$$\begin{aligned} L_{11}L_{11}^T &= A_{11}, \\ L_{21}^T &= L_{11}^{-1}A_{12}, \\ \tilde{A}_{22} &= A_{22} - L_{12}^T L_{12}, \\ L_{22}L_{22}^T &= \tilde{A}_{22}. \end{aligned}$$

We recall that 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 $L_{21}L_{21}^T$. 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 $n/2 \times n/2$ requires $n^3/12$ flops, which is 1/4 of the total number of $n^3/3$ flops. The triangular solve and modification step each take $n/8$ flops. (Note that only the upper the upper triangular part of \tilde{A}_{22} needs to be computed.)

Using these equations recursively a recursive algorithm for Cholesky factorization is obtained. The algorithm below does not take advantage of the symmetry in the matrices, r.g. in the modification of the $(2, 2)$ block.

Algorithm 7.12. *Recursive Cholesky Factorization.*

Let $A \in \mathbf{R}^{n \times n}$ be a symmetric positive definite matrix. The following recursive algorithm computes the Cholesky factorization of A .

```
function L = rchol(A);
```

```
%RCHOL Recursive Cholesky Factorization
[n,n] = size(A);
if n > 1
    n1 = floor(n/2); n2 = n-n1;
    j1 = 1:n1; j2 = n1+1:n;
    L11 = rchol(A(j1,j1)); %recursive call
    L12 = L11\A(j1,j2); %triangular solve
    A(j2,j2) = A(j2,j2) - L12'*L12; %modify %%(2,2) block
    L22 = rchol(A(j2,j2)); %recursive call
    L = [L11, zeros(n1,n2); L12', L22];
else
    L = sqrt(A);
end
```

Note that in the recursive algorithm *all the work is done in the triangular solves and matrix multiplication*. At each 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 as $1/4$. For large problems this does not affect the total efficiency. But for small problems, where most of the calls to level 3 BLAS has small problem size, the efficiency is deteriorated. This can be avoided by calling a standard Cholesky routine if the problem size satisfies $n > n_{min}$. A recursive algorithm for Cholesky factorization of a matrix in packed storage format is described in [9].

A recursive algorithm for the LU factorization of a matrix A can be obtained similarly. In order to accommodate partial pivoting we need to consider the LU factorization of a rectangular matrix $A \in \mathbf{R}^{m \times n}$ into a product of $L \in \mathbf{R}^{m \times n}$ and $U \in \mathbf{R}^{n \times n}$. The total number of flops required for this factorization is $n^2m - n^3/3$.

We partition the matrix as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}, \quad (7.6.14)$$

where $A_{11} \in \mathbf{R}^{n_1 \times n_1}$ and $A_{22} \in \mathbf{R}^{n_2 \times (m-n_1)}$. Then the size of the blocks in the factorization are

$$L_{11}, U_{11} \in \mathbf{R}^{n_1 \times n_1}, \quad U_{22} \in \mathbf{R}^{n_2 \times n_2}, \quad L_{22} \in \mathbf{R}^{(m-n_1) \times n_1}.$$

Equating blocks on the left and right hand side gives

$$\begin{aligned} \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} U_{11} &= \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}, & n^2m/4 - n^3/24 \text{ flops} \\ U_{12} &= L_{11}^{-1} A_{12}, & n^3/8 \text{ flops} \\ \tilde{A}_{22} &= A_{22} - L_{21} U_{12}, & n^2m/2 - n^3/4 \text{ flops} \\ L_{22} U_{22} &= \tilde{A}_{22}. & n^2m/4 - n^3/6 \text{ flops} \end{aligned} \quad (7.6.15)$$

The flop counts above are for the case that n is even and $n_1 = n_2 = n/2$. Hence, the LU factorization of $A \in \mathbf{R}^{m \times n}$ is reduced to an LU factorization of the first block of n_1 columns of A 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.

For a recursive algorithm for LU factorization with partial pivoting the same approach as in (7.6.15) can be used. We then perform the two LU factorizations with partial pivoting; (see Gustavson [297] and Toledo [569]). As before the recursion will produce a binary tree with of depth $k + 1$ where $2^{k-1} < n \leq 2^k$. At each level i , 2 calls to level 3 BLAS are made.

Algorithm 7.13. Recursive LU Factorization.

The following recursive algorithm computes the LU factorization of the matrix $A \in \mathbf{R}^{m \times n}$, $m \geq n$.

```

function [LU,p] = rlu(A);
% Recursive LU factorization with partial pivoting
% of A. p holds the final row ordering
[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; % modify (2,2) block
    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)/A(1,1);
end

```

In going from level i to $i + 1$, the number of BLAS calls doubles. The problem size in the recursive call are now $m \times n/2$ and $n/2 \times n/2$. From the flop count above it follows that the total number of flops done at each level now goes down by more than a factor of two.

7.6.5 Kronecker Structure and Linear Systems

The **Kronecker product**²⁵ arise in several application areas such as signal and image processing, photogrammetry, computer vision, multidimensional data fitting, etc. Such systems can be solved with great savings in storage and operations. Since often the size of the matrices A and B is large, resulting in models involving several hundred thousand equations and unknowns, such savings may be essential.

Definition 7.6.3.

The **Kronecker product** of two matrices $A \in \mathbf{R}^{m \times n}$ and $B \in \mathbf{R}^{p \times q}$ is the $mp \times nq$ block matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}. \quad (7.6.16)$$

We state without proofs some elementary facts about Kronecker products. From the definition (7.6.16) it easily follows that

$$\begin{aligned} (A + B) \otimes C &= (A \otimes C) + (B \otimes C), \\ A \otimes (B + C) &= (A \otimes B) + (A \otimes C), \\ A \otimes (B \otimes C) &= (A \otimes B) \otimes C, \\ (A \otimes B)^T &= A^T \otimes B^T. \end{aligned}$$

From the last identity it follows that if the factors A and B are symmetric the product $A \otimes B$ shares this property. We next show a mixed-product relation, which is less obvious:

Lemma 7.6.4.

Let $A \in \mathbf{R}^{m \times n}$, $B \in \mathbf{R}^{p \times q}$, $C \in \mathbf{R}^{n \times k}$, and $D \in \mathbf{R}^{q \times r}$. Then the ordinary matrix products AC and BD are defined, and

$$(A \otimes B)(C \otimes D) = AC \otimes BD. \quad (7.6.17)$$

Proof. Let $A = (a_{ik})$ and $C = (c_{kj})$. Partitioning according to the sizes of B and D , $A \otimes B = (a_{ik}B)$ and $C \otimes D = (c_{kj}D)$. Hence, , the (i, j) th block of $(A \otimes B)(C \otimes D)$

²⁵Leopold Kronecker (1823–1891) German mathematician. He is known also for his remark “God created the integers, all else is the work of man”.

equals

$$\sum_{k=1}^n a_{ik} B c_{kj} D = \left(\sum_{k=1}^n a_{ik} c_{kj} \right) BD,$$

which is the (i, j) th element of AC times BD , which is the (i, j) th block of $(A \otimes B)(C \otimes D)$. \square

As an application of this result we show that if $P, Q \in \mathbf{R}^{n \times n}$ are orthogonal then so is $P \otimes Q$. We have

$$(P \otimes Q)^T (P \otimes Q) = (P^T \otimes Q^T)(P \otimes Q) = (P^T P \otimes Q^T Q) = I_n \otimes I_n = I_{n^2}.$$

An important fact for computational work is that in many cases the Kronecker product inherits the structure of its factors. For example, if A and B are lower (upper) triangular then $A \otimes B$ is lower (upper) triangular. If A and B are banded or Toeplitz, then $A \otimes B$ is block banded or block Toeplitz.

If $A \in \mathbf{R}^{n \times n}$ and $B \in \mathbf{R}^{p \times p}$ are nonsingular, then by Lemma 7.6.4

$$(A^{-1} \otimes B^{-1})(A \otimes B) = I_n \otimes I_p = I_{np}.$$

It follows that $A \otimes B$ is nonsingular and

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \quad (7.6.18)$$

In working with Kronecker products, matrices are sometimes regarded as vectors and vectors are sometimes made into matrices. We now introduce an operator which makes this precise.

Definition 7.6.5. Given a matrix $C = (c_1, c_2, \dots, c_n) \in \mathbf{R}^{m \times n}$ we define

$$\text{vec}(C) = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \in \mathbf{R}^{mn},, \quad (7.6.19)$$

that is, the vector formed by **stacking** the columns of C into one long vector.

We now state an important result which shows how the vec-function is related to the Kronecker product.

Lemma 7.6.6.

If $A \in \mathbf{R}^{m \times n}$, $B \in \mathbf{R}^{p \times q}$, and $X \in \mathbf{R}^{q \times n}$, then

$$(A \otimes B)\text{vec } X = \text{vec}(BXA^T). \quad (7.6.20)$$

Proof. Denote the k th column of a matrix M by M_k . Then

$$\begin{aligned} (BXA^T)_k &= BX(A^T)_k = B \sum_{i=1}^n a_{ki} X_i \\ &= (a_{k1}B \ a_{k2}B \ \dots a_{kn}B) \text{vec } X, \end{aligned}$$

where $A = (a_{ij})$. But this means that $\text{vec}(BXA^T) = (A \otimes B)\text{vec} X$. \square

Linear systems where the matrix is a Kronecker product are ubiquitous in applications. Let $A \in \mathbf{R}^{n \times n}$ and $B \in \mathbf{R}^{p \times p}$ be nonsingular, and $C \in \mathbf{R}^{p \times n}$. Consider the Kronecker linear system

$$(A \otimes B)x = c, \quad c = \text{vec}(C), \quad (7.6.21)$$

which is of order np . Solving this by LU factorization would require $O(n^3p^3)$ flops. Using (7.6.18) the solution can be written

$$x = (A \otimes B)^{-1}\text{vec}(C) = (A^{-1} \otimes B^{-1})\text{vec}(C). \quad (7.6.22)$$

Using Lemma 7.6.6 this is equivalent to

$$x = \text{vec}(X), \quad X = B^{-1}CA^{-T}. \quad (7.6.23)$$

Here X can be computed by solving two matrix equations

$$BY = C, \quad A^TX = Y.$$

A consequence of this result is that linear systems of the form (7.6.21) can be solved fast. The operation count is reduced from to $O(n^3p^3)$ to $O(n^3 + p^3)$ flops.

Review Questions

- 6.1** How many operations are needed (approximately) for
- (a) The LU factorization of a square matrix?
 - (b) The solution of $Ax = b$, when the triangular factorization of A is known?
- 6.2** To compute the matrix product $C = AB \in \mathbf{R}^{m \times p}$ we can either use an outer- or inner-product formulation. Discuss the merits of the two resulting algorithms when A and B have relatively few nonzero elements.
- 6.3** Is the Hadamard product $A.*B$ a submatrix of the Kronecker product $A \otimes B$?

Problems

- 6.1** Assume that for the nonsingular matrix $A_{n-1} \in \mathbf{R}^{(n-1) \times (n-1)}$ we know the LU factorization $A_{n-1} = L_{n-1}U_{n-1}$. Determine the LU factorization of the **bordered matrix** $A_n \in \mathbf{R}^{n \times n}$,

$$A_n = \begin{pmatrix} A_{n-1} & b \\ c^T & a_{nn} \end{pmatrix} = \begin{pmatrix} L_{n-1} & 0 \\ l^T & 1 \end{pmatrix} \begin{pmatrix} U_{n-1} & u \\ 0 & u_{nn} \end{pmatrix}.$$

Here $b, c \in \mathbf{R}^{n-1}$ and a_{nn} are given and $l, u \in \mathbf{R}^{n-1}$ and u_{nn} are to be determined.

- 6.2** 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 that the diagonal blocks U_{11} and U_{22} are square and nonsingular.

- 6.3** Write a recursive LU Factorization algorithm based on the 2×2 block LU algorithm.

- 6.4** (a) Let $A \in \mathbf{R}^{m \times n}$, $B \in \mathbf{R}^{n \times p}$, with m and n even. Show that, whereas conventional matrix multiplication requires mnp multiplications (M) and $m(n - 1)p$ additions (A) to form the product $C = AB \in \mathbf{R}^{m \times p}$, Strassen's algorithm, using conventional matrix multiplication at the block level, requires

$$\frac{7}{8}mnp \text{ M} + \frac{7}{8}m(n-2)p + \frac{5}{4}n(m+p) + 2mp \text{ A.}$$

(b) Show, using the result in (a), that if we assume that “M \approx A”, Strassen's algorithm is cheaper than conventional multiplication when

$$mnp \leq 5(mn + np + mp).$$

- 6.5** (a) Verify the identity $(A \otimes B)^T = A^T \otimes B^T$.
(b) Show the identity $\text{vec}(A)^T \text{vec}(B) = \text{trace}(A^T B)$.

7.7 Sparse Linear Systems

7.7.1 Introduction

A matrix $A \in \mathbf{R}^{n \times n}$ is called **sparse** if only a small fraction of its elements are nonzero. Similarly, a linear system $Ax = b$ is called sparse if its matrix A is sparse. (the right-hand side may be dense or sparse.) There is no precise definition of what constitutes a sparse matrix. Following Wilkinson and Reinsch [614], a matrix A will be called sparse if the percentage of nonzero elements is small and their distribution is such that it is economical (either in computer time or storage) to take advantage of their presence.

Large sparse linear systems arise in numerous areas of application, such as the numerical solution of partial differential equations, mathematical programming, structural analysis, chemical engineering, electrical circuits, and networks. The car you drive and the airplane in which you fly has been designed using sparse matrix technology. Large could imply a value of n in the range 1,000–10,000,000. Typically, A will have relatively few (say, 10–100) nonzero elements in each row, regardless of the value of n . 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 equals 0.625% of the elements. Without exploitation of sparsity, such a problem would be totally intractable.

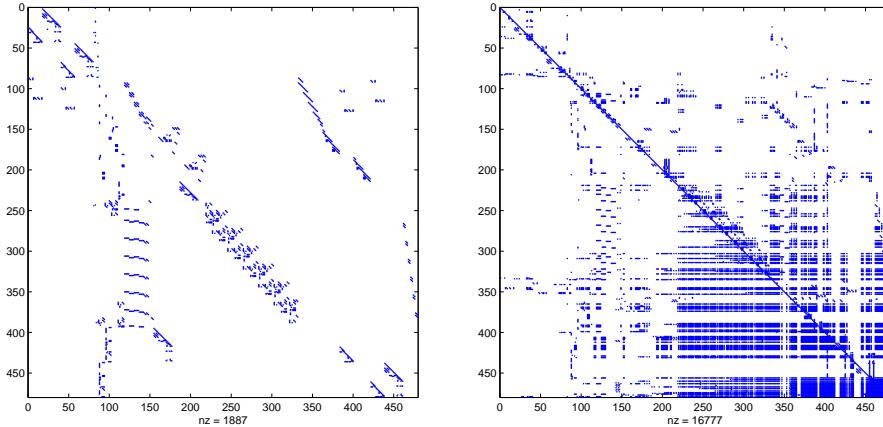


Figure 7.7.1. Nonzero pattern of a matrix W and its LU factors.

The aim of this section is to present the basic theory of sparse matrix techniques. The sparse matrix packages now in use are huge and rely on decades of sophisticated evolution. For example, the LU, Cholesky, and QR factorizations codes in MATLAB total about 100,000 lines of code. Clearly it is beyond the scope of this book to consider all details of such codes. For the origin and details of the sparse matrix codes in MATLAB we refer to Gilbert, Moler, and Schreiber [247].

A simple case of sparsity, already considered in Section 7.4, is band matrices. Here the nonzero elements are concentrated on a narrow band centered on the main diagonal. Matrices of small bandwidth occur naturally, since they correspond to situations where only variables "close" to each other are coupled by observations. However, often the nonzero elements are distributed in a less systematic manner. Figure 7.7.1 was produced by MATLAB using the function `spy`, which visualizes the sparsity pattern of a matrix. The figure shows a sparse matrix W of order $n = 479$ with 1887 nonzero elements (or 0.9%) and its LU factors. The matrix is taken from the Harwell–Boeing sparse matrix test collection; see Duff, Grimes, and Lewis [177]. It comes from a model of an eight stage chemical distillation column due to Westerberg. Other applications may give a pattern with quite different characteristics.

Large sparse systems coming from partial differential equations in two and three dimensions are often treated by iterative methods; see Chapter 10. Iterative methods often have to be specially designed for a particular class of problems. Direct methods for sparse systems are easier to develop as "black box" algorithms. The boundary between iterative and direct methods is not sharp. Sparse (approximate) factorizations are often used as a so-called **preconditioners** to improve the rate of convergence of iterative methods. Indeed the only way to solve really large linear systems often is by using a combination of direct and iterative methods.

When solving sparse linear systems by direct methods it is important to avoid storing and operating on the elements which are known to be zero. One should try

also to minimize **fill** as the computation proceeds, which is the term used to denote the creation of new nonzero elements during the elimination. Recall that in step k of Gaussian elimination the elements are transformed according to

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{kj}^{(k)} a_{ik}^{(k)}}{a_{kk}^{(k)}} \quad k+1 \leq i \leq j \leq n.$$

Thus, a zero entry $a_{ij}^{(k)}$ can become nonzero only if the three elements $a_{kj}^{(k)}$, $a_{ik}^{(k)}$, and $a_{kk}^{(k)}$ are nonzero. As shown in Figure 7.7.1, the LU factors of W contain 16777 nonzero elements, or about nine times as many as in the original matrix.

Example 7.7.1.

When considering the fill that occurs during Gaussian elimination the nonzero structure of the sparse matrix can be represented by a schematic diagram introduced by J. H. Wilkinson and which we call a **Wilkinson diagram**. For the matrix in (7.7.1) this is

$$\begin{pmatrix} \times & & \times \\ \times & \times & \times \\ & \times & \times & \times \\ \times & & \times \\ & \times & \times \end{pmatrix}.$$

We now perform Gaussian elimination now *symbolically*, i.e., we do not operate on any numerical values. In the first step only the two first rows take part. The element in position (2,1) is zeroed and a fill 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.

$$\begin{pmatrix} \times & \times \\ \otimes & \times & + & \times \\ \times & \times & & \times \\ \times & & \times \\ \times & \times \end{pmatrix}, \quad \begin{pmatrix} \times & \times \\ \otimes & \times & + & \times \\ \otimes & \times & + & \times \\ \otimes & + & \times & + \\ & \times & \times \end{pmatrix}, \quad \begin{pmatrix} \times & \times \\ \otimes & \times & + & \times \\ \otimes & \times & + & \times \\ \otimes & \otimes & \times & + \\ \otimes & & \times \end{pmatrix}.$$

The structure of the LU factors then are

$$L = \begin{pmatrix} 1 & & & \\ \times & 1 & & \\ \times & & 1 & \\ \times & \times & & 1 \\ & & \times & \end{pmatrix}, \quad U = \begin{pmatrix} \times & \times & + & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{pmatrix}.$$

We find that the LU factors have 16 nonzero elements compared to 12 for A , a modest increase.

Note that it could happen due to numerical cancellation that an element $a_{ij}^{(k+1)}$ becomes zero even when $a_{ij}^{(k)} \neq 0$. Since we cannot detect this when working only with the structure of the original matrix we will ignore this possibility in the following. This is called the **no-cancellation assumption**.

7.7.2 Storage Schemes for Sparse Matrices

A simple scheme to store a sparse matrix is to store the nonzero elements in an unordered one-dimensional array AC together with two integer vectors ix and jx containing the corresponding row and column indices.

$$ac(k) = a_{i,j}, \quad ix(k) = i, \quad jx(k) = j, \quad k = 1 : nnz.$$

Hence, A is stored in “coordinate form” as an unordered set of triples consisting of a numerical value and two indices. For example, the matrix

$$A = \begin{pmatrix} a_{11} & 0 & a_{13} & 0 & 0 \\ a_{21} & a_{22} & 0 & a_{24} & 0 \\ 0 & a_{32} & a_{33} & 0 & a_{35} \\ 0 & a_{42} & 0 & a_{44} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} \end{pmatrix}, \quad (7.7.1)$$

is stored in coordinate form as

$$\begin{aligned} AC &= (a_{13}, a_{22}, a_{21}, a_{33}, a_{35}, a_{24}, a_{32}, a_{42}, a_{44}, a_{55}, a_{54}, a_{11}) \\ ix &= (1, 2, 2, 3, 3, 2, 3, 4, 4, 5, 5, 1) \\ jx &= (3, 2, 1, 3, 5, 4, 2, 2, 4, 5, 4, 1) \end{aligned}$$

We will denote the number of nonzero elements in a matrix A by $nnz(A)$. In this example $nnz(A) = 12$.

The coordinate storage scheme is very convenient for the initial representation of a general sparse matrix. One advantage is that more nonzero elements are easily added to the structure. However, except for this it is not useful. A drawback is that there is no efficient way to access a row or a column of A , which is needed for implementing Gaussian elimination. This illustrates an important point about storage schemes for sparse matrices. It must allow an efficient implementation of the operations we need to perform on the matrix. In this context it should permit rapid execution of the basic elimination steps in Gaussian elimination.

Usually the pattern of nonzero elements is very irregular, as illustrated in Figure 7.7.1. Consider first a storage scheme for a sparse vector x . The nonzero elements of x can be stored in **compressed form** in a vector xc with dimension $nnz(z)$. Additionally an integer vector ix is stored, which contains the indices of the corresponding nonzero elements in xc . Thus, a sparse vector x is represented by the triple (nnz, xc, ix) , where

$$xc_k = x_{ix(k)}, \quad k = 1 : nnz.$$

For example, the vector $x = (0, 4, 0, 0, 1, 0, 0, 0, 6, 0)$ can be stored as

$$xc = (1, 4, 6), \quad ix = (5, 2, 9), \quad nnz = 3.$$

Operations on sparse vectors are simplified if *one* of the vectors is first **un-compressed**, i.e., stored in a full vector of dimension n . Clearly this operation can

be done in time proportional to the number of nonzeros, and allows direct random access to specified element in the vector. Vector operations, e.g., adding a multiple a of a sparse vector x to an uncompressed sparse vector y , or computing the inner product $x^T y$ can then be performed in *constant time per nonzero element*. Assume, for example, that the vector x is held in compressed form as nnz pairs of values and indices, and y is held in a full length array. Then the operation $y := a * x + y$ may be expressed as

$$\text{for } k = 1 : nnz, \quad y(ix(k)) := a * xc(k) + y(ix(k));$$

A matrix can be stored as a in 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 a_{ik} is given in $ix(k)$. Finally, we need a third vector $ja(i)$, which gives the position in the array AC of the first element in the j th column of A . For example, the 5×5 matrix in (7.7.1) is stored as:

$$\begin{aligned} AC &= (a_{11}, a_{21} \mid a_{22}, a_{32}, a_{42} \mid a_{13}, a_{33}, a_{24} \mid a_{44}, a_{54} \mid a_{35}, a_{55}), \\ ix &= (1, 2, 2, 3, 4, 1, 3, 2, 4, 5, 3, 5), \\ ja &= (1, 3, 6, 8, 11, 13). \end{aligned}$$

This is essentially the storage scheme used in MATLAB for a sparse real matrix. (If the matrix is complex, the imaginary parts are stored in another array.)

Alternatively, a similar scheme storing A as a sequence of row vectors may be used. A drawback with these schemes is that it is expensive to insert new nonzero elements in the structure when fill occurs. Note that a last element equal to $nnz + 1$ is stored in $ja(n)$ to indicate the end of this vector.

The components in each row 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. However, more usual is that a complete row of A has to be retrieved, and this can be done quite efficiently. This scheme can be used unchanged for storing the lower triangular part of a symmetric positive definite matrix.

In the general sparse storage scheme only nonzero elements are stored. This saving is however bought at the cost of storage for the vector ix of column subscripts. This overhead storage can often be decreased by using a clever compressed scheme due to Sherman, see George and Liu [239, pp. 139–142].

If the matrix is stored as a sequence of sparse row vectors, the entries in a particular column cannot be retrieved without a search of nearly all elements. This is needed, for instance, to find the rows which are involved in a stage of Gaussian elimination. A solution is then to store also the structure of the matrix as a set of column vectors. If a matrix is input in coordinate form it the conversion to this storage form requires a sorting of the elements, since they may be in arbitrary order. This can be done very efficiently in $O(n) + O(\tau)$ time, where τ is the number of nonzero elements in the factors and n is the order of the matrix.

Another way to avoid extensive searches in data structures is to use a linked list to store the nonzero elements. Associated with each element is a pointer to the location of the next element in its row and a pointer to the location of the next element in its column. If also pointer to the first nonzero in each row and column are stored there is a total overhead of integer storage of $2(\tau + n)$. This allows fills to be added to the data structure with only two pointers being altered. Also, the fill can be placed anywhere in storage so no reorderings are 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.

An important distinction is between **static** storage structures that remain fixed and **dynamic** structures that can accommodate fill. If only nonzeros 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 nonzeros in the factors can be predicted in advance, as is the case for the Cholesky factorization.

7.7.3 Graphs and Sparse Matrices.

In sparse matrix methods the representation of the structure of a sparse matrix by a **graph** plays an important role. We introduce here some basic concepts of graph theory. An unsymmetric matrix $A \in \mathbf{R}^{n \times n}$ can be associated with a directed graph $G = (X, E)$. Here X is a set of n nodes (or vertices) labeled x_i , (or simply i) $i = 1 : n$, and E is a set of edges, i.e., *ordered pairs* of nodes. There is a directed edge from x_i to x_j , $i \neq j$, if $a_{ij} \neq 0$.²⁶ (Usually self loops corresponding to $a_{ii} \neq 0$ are not included.) A graph $G' = (X', E')$ is said to be a subgraph of $G = (X, E)$ if $X' \subset X$ and $E' \subset E$.

The directed graph corresponding to A is constructed as follows. Let x_1, \dots, x_n be n distinct nodes in the plane. For each $a_{ij} \neq 0$ in A we connect node x_i to node x_j by means of directed edge $(\overrightarrow{x_i}, \overrightarrow{x_j})$ from node x_i to node x_j . 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

$$A = \begin{pmatrix} \times & \times & & \times \\ & \times & & \\ \times & & \times & \\ & \times & \times & \times \end{pmatrix}$$

has nodes x_i $i = 1 : 4$ and directed edges

$$E = \{(\overrightarrow{x_1}, \overrightarrow{x_2}), (\overrightarrow{x_1}, \overrightarrow{x_4}), (\overrightarrow{x_3}, \overrightarrow{x_1}), (\overrightarrow{x_4}, \overrightarrow{x_2}), (\overrightarrow{x_4}, \overrightarrow{x_3})\}.$$

For a symmetric matrix $A \in \mathbf{R}^{n \times n}$ if an element $a_{ij} \neq 0$, then $a_{ji} \neq 0$. 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

²⁶A nonsymmetric (even nonsquare) matrix $A \in \mathbf{R}^{m \times n}$ can be represented by a bipartite graph; see Section 7.7.7.

of nodes labeled $1 : n$ and undirected edges $(x_i, x_j) \in E$ if and only if $a_{ij} = a_{ji} \neq 0$, $i \neq j$. For example, the undirected graph corresponding to the symmetric matrix

$$A = \begin{pmatrix} \times & \times & \times & 0 \\ \times & \times & & \times \\ \times & & \times & \times \\ & \times & \times & \times \end{pmatrix}$$

has nodes x_i , $i = 1 : 4$ and 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 $P \in \mathbf{R}^{n \times n}$ the graphs $G(A)$ and $G(PAP^T)$ are the same except that the labeling of the nodes are different. Hence, the unlabeled graph represents the structure of A without any particular ordering. Finding a good permutation for A is equivalent to finding a good labeling for its graph.

Let i and j be nodes in the undirected graph $G(X, E)$ of a symmetric matrix A . Two nodes x_i and x_j , $i \neq j$, 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

$$\text{Adj}(x) = \{y \in X \mid x \text{ and } y \text{ are adjacent}\}. \quad (7.7.2)$$

The number of nodes adjacent to x is called the **degree** of x and is denoted by $|\text{Adj}(x)|$.

A sequence of edges $(x_1, x_2), (x_2, x_3), \dots, (x_k, x_{k+1})$ in a graph $G(X, E)$ such that

$$(x_i, x_{i+1}) \in E, \quad i = 1 : k,$$

is a **path** of length k from node x_1 to x_{k+1} . If $x_1 = x_{k+1}$ the path is a **cycle**. Every cycle consists of at least one edge. If there is a path between every pair of distinct nodes, then we say that the graph is **connected**. A directed graph is **strongly connected**, if there is a path between every pair of distinct nodes 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, that is, all vertices of the subgraph are pairwise connected by edges in the graph.

A disconnected graph consists of at least two separate connected subgraphs. 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.

A **tree** T is a (directed or undirected) graph with a distinguished node x_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 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 . A graph whose strongly connected components are trees is called a **forest**. Trees form a class of data structure that is easy to store and manipulate and play an important role in many aspects of sparse matrix factorization.

Graph Model of Cholesky Factorization

The structure of the Cholesky factor L of a symmetric positive definite matrix A can be determined using a graph model of Gaussian elimination. In this the undirected $G(A)$ is recursively transformed into a sequence of **elimination graphs** as follows.

Algorithm 7.14. *Graph Model of 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 , $i = 1 : n - 1$. Here G_i is obtained from $G_{(i-1)}$ by removing the pivot node x_i and its incident edges and adding fill edges

$$\{(j, k) \mid (j, k) \in \text{Adj}(v_i), j \neq k\}.$$

The fill edges correspond to the set of edges required to make the adjacent nodes of v_i pairwise adjacent. The filled graph $G_F(A)$ of A is a 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 ,

$$G(L + L^T) \subset G_F(A). \quad (7.7.3)$$

Under the no-cancellation assumption, the relation (7.7.3) holds with equality.

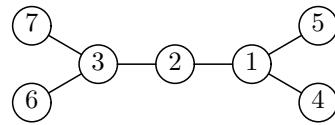
The ordering of the rows and columns of a matrix can, sometimes drastically, affect the amount of fill in the factorization. Elimination orderings that will produce no fill are referred to as **perfect elimination orderings**. Let $A \in \mathbf{R}^{n \times n}$ be a symmetric irreducible matrix whose graph $G(A)$ is a tree. In this case such orderings are 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 $k - 1$.

Example 7.7.2.

Consider the sparse symmetric matrix A (only the structure of the lower triangular half is shown).

$$A = \begin{pmatrix} \times & \times & & \times & \times \\ \times & \times & \times & & \\ & \times & \times & & \times & \times \\ \times & & & \times & & \\ & & & & \times & \\ & \times & & & \times & \\ & & \times & & & \times \end{pmatrix}, \quad L = \begin{pmatrix} \times & & & & \\ \times & \times & & & \\ & \times & \times & & \\ & & \times & + & + & \times \\ & & & \times & + & + & \times \\ & & & & \times & + & + & \times \\ & & & & & \times & + & + & + & \times \end{pmatrix}. \quad (7.7.4)$$

The structure of its Cholesky factor L is also shown. The ten elements marked $+$ are the fill that occur in the Cholesky factorization. It can be verified that the labeled graph of A is the tree



If the rows and columns of the matrix A are reordered in postorder $4, 5, 7, 6, 3, 1, 2$, then there is no fill in the Cholesky factor L and the matrix PAP^T and $L + L^T$ both have the structure

$$\begin{pmatrix} \times & & & & \times & \\ & \times & & & & \times \\ & & \times & & \times & \\ & & & \times & \times & \\ & & & & \times & \times \\ \times & \times & & & & \times \\ & & & & \times & \times \end{pmatrix}, \quad (7.7.5)$$

At the elimination of each node (except the root) this will be connected only to one uneliminated node, namely its parent. Hence, in the lower triangular part of A , each column has exactly one off-diagonal element. Such a structure can be represented by the subscripts of the off-diagonal nonzeros in the columns.

7.7.4 The Elimination Tree

The dependency between the columns of L can be characterized by considering the columnwise version of Cholesky given in Section 7.3.2. For a sparse matrix it can be written in slightly modified form as follows:

```

for  $i = 1 : n$ 
  for  $j = 1 : i - 1$ 
    if  $l_{i,j} \neq 0$ 
       $l_{i:n,i} := l_{i:n,i} - l_{i,j}l_{i:n,j};$ 
    end
  end
end
  
```

This shows that the i th column $l_{i:n,i}$ is only modified by the j th column $l_{i:n,j}$, $j = 1 : i - 1$ if the element $l_{i,j}$ is nonzero. The nonzero pattern of the i th column equals that of the i th column of A and the direct sum of the nonzero patterns of the previous columns for which $l_{i,j} \neq 0$. We have shown the following result.

Proposition 7.7.1.

For $i > j$, the j th column of the Cholesky factor L depends on the i th column if and only if $l_{ij} \neq 0$.

We now introduce a tree structure that plays an important role in sparse Cholesky factorization. Let $A \in \mathbf{R}^{n \times n}$ be an irreducible symmetric matrix and

L its Cholesky factor. Since A is irreducible, it is easy to see that the first $n - 1$ columns in L must have at least one nonzero element. For each column in L , remove all nonzeros below the diagonal except the first. Let L_t be the resulting matrix. Then the undirected graph $G(L_t^T L_t)$ has a tree structure which depends only on the structure of A and its initial reordering. This tree is a spanning tree of the filled graph $G_F(A)$, i.e., it contains all the nodes of $G(F)$ and a subset of its edges so that it forms a tree connecting all nodes. We denote this tree by $T(A)$ and refer to as the **elimination tree** of A .

Definition 7.7.2.

Let L be the Cholesky factor of the symmetric matrix $A \in \mathbf{R}^{n \times n}$. The elimination tree $T(A)$ of A is a rooted tree with n nodes labeled from 1 to n , where node $p(j) > j$ is the parent node of node j , if and only if

$$p(j) = \min\{i > j \mid l_{ij} \neq 0\}. \quad (7.7.6)$$

The elimination tree is conveniently represented by the parent vector $p(j)$, $j = 1 : n - 1$. For example, for the matrix PAP^T in (7.7.5) the parent vector is given by

j	1	2	3	4	5	6	7
$p(j)$	6	6	5	5	7	7	7

The following theorem partly characterizes the structure of L in terms of its elimination tree.

Theorem 7.7.3.

Assume that $l_{ij} \neq 0$, $i > j$, in the Cholesky factor L of a symmetric matrix A . Then the node x_i is an ancestor of node x_j in the elimination tree $T(A)$.

We now consider a necessary and sufficient condition for an element l_{ij} to be nonzero. It is based on the special type of path in the original graph $G(A)$. For proofs of this and the following results on elimination trees we refer to Liu [413].

Theorem 7.7.4. [Liu [413, Theorem 3.3]]

Let $i > j$. Then $l_{ij} \neq 0$ if and only if there exists a path

$$x_i, x_{p_1}, \dots, x_{p_i}, x_j$$

in the graph $G(A)$ such that all subscripts in $\{p_1, \dots, x_{p_i}\}$ are less than j .

This result can be used to construct the elimination tree from the filled graph $G_F(A)$ as follows. Modify $G_F(A)$ by using a *directed edge* from node x_j to node x_k , $k > j$, to indicate that column k depends on column j . This gives a directed graph that is the graph $G(L^T)$. This directed graph is simplified by a so-called transitive reduction. That is, if there is a directed edge from x_j to x_k , and also

a directed path of length greater than one from x_j to x_k , then the edge from x_j to x_k is removed. The removal of all such edges gives a tree, which is exactly 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 rooted at node x_i in the elimination tree $T(A)$. Thus, it 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 7.7.5. [Liu [413, 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 $T[x_i]$ 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 .

Liu [413, Sec. 5.1] gives an algorithm that to determine the elimination tree that 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 $p(k)$, $k = 1 : i.1$. The algorithms 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. 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 ordering of the elimination tree is equivalent. However, the class of postordering is particularly suitable for many purposes.

7.7.5 Orderings Algorithms for Cholesky Factorization

Consider a symmetric positive definite sparse linear system $Ax = b$. After a symmetric permutation of rows and columns the system becomes

$$(PAP^T)z = Pb, \quad x = P^T z,$$

where P is a permutation matrix. If $PAP^T = LL^T$ is the Cholesky factorization, then the solution is obtained by substitution in the two triangular systems

$$Ly = Pb, \quad L^T z = y.$$

When computing a sparse Cholesky factorization an important first task is to find a symmetric reordering of the rows and columns that reduces fill and the number of arithmetic operations.

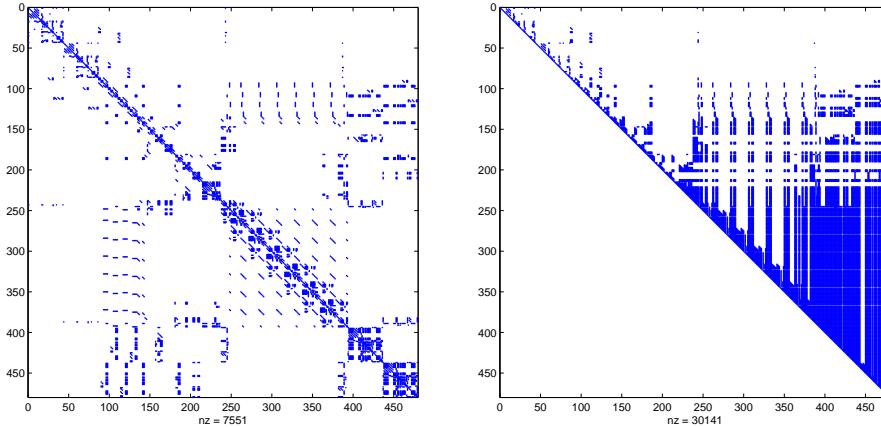


Figure 7.7.2. Nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor L^T ; $nnz(A_L) = 4015$; $nnz(L) = 30,141$.

We will use as a test example the symmetric matrix $A = WW^T$, where W is the matrix west0479 in Figure 7.7.1. Figure 7.7.2 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 4015. 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.

For a symmetric positive definite matrix A the Cholesky factorization is numerically stable for any choice of pivots along the diagonal. Hence, the permutation matrix P can be chosen with regard only to preserving sparsity. To find the *optimal ordering* which minimizes the number of nonzeros in the Cholesky factor L is known to be computationally intractable: an NP-complete problem; see Yannakakis [621]. We are therefore forced to rely on heuristic algorithms. These usually can give orderings which nearly minimize fill as well as the arithmetic operation count.

In a sparse symmetric solver the algorithm can be divided into 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 $P^T AP$ 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 $P^T AP$ in the data structure for L . Perform the numerical Cholesky factorization.
4. Solve the two triangular systems $Ly = Pb$, and $L^T zx = y$ and set $x = P^T z$.

The static storage structure for L generated in step 2 will lead 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 different heuristic ordering algorithms for sparse Cholesky factorization.

Reverse Cuthill–McKee Ordering

We first consider an ordering algorithm that aims at minimizing the size of the envelope of a symmetric matrix A . By Definition 7.4.7 the **envelope** of a matrix A is the index set

$$\text{Env}(A) = \{(i, j) \mid f_i \leq j \leq i; \text{ or } l_j \leq i < j; \}.$$

The envelope of a symmetric matrix is defined by the envelope of its lower triangular part including the main diagonal. In Section 7.4.6 we showed that in the Cholesky factorization all zeros outside the envelope of a matrix A are preserved. The **Cuthill–McKee algorithm** uses a local minimization and works on the graph $G(A)$ as follows:

1. Determine a starting node 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. Before discussing this choice we remark that as discovered in 1971 (see George and Liu [239, Section 4.3]) the reversal of the Cuthill–McKee ordering is never inferior and often much superior to the original ordering. If the ordering of the nodes given by Cuthill–McKee is x_1, x_2, \dots, x_n then the reverse Cuthill–McKee ordering is x_n, \dots, x_2, x_1 .

Experience has shown that good starting nodes are nodes which are near the maximum distance apart in the graph $G(A) = (X, E)$. This can be made more precise by defining the **eccentricity** of a node x to be

$$\ell(x) = \max\{d(x, y) \mid y \in X\}, \quad (7.7.7)$$

where the distance $d(x, y)$ is the length of the shortest path in $G(A)$. The diameter of the graph is then given by

$$\delta(G) = \max\{d(x, y) \mid x, y \in X\}, \quad (7.7.8)$$

A node $x \in X$ is a **peripheral** node if its eccentricity equals the diameter of the graph. A heuristic algorithm to find an approximate peripheral node based on so-called level structures in the graph is given in George and Liu [239, Section 4.3]).

Figure 7.7.3 shows the nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor after a reverse Cuthill–McKee reordering. The number of nonzero elements in the Cholesky factor has decreased to 23,866. This still is a reduction of 20%, but still not very satisfactory.

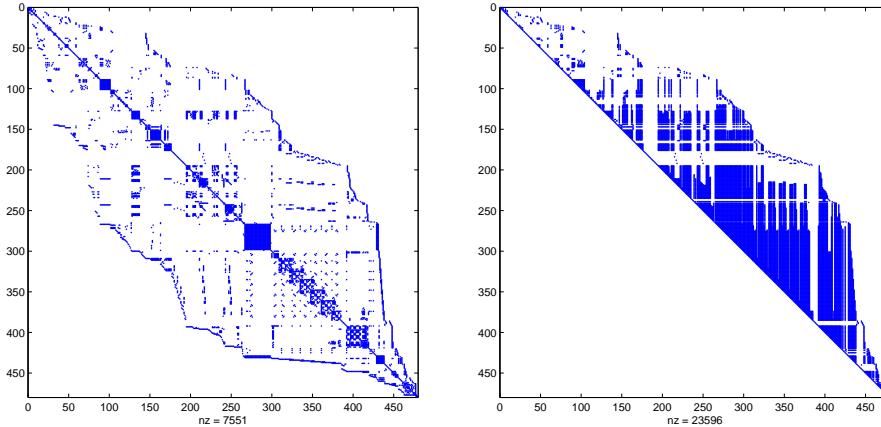


Figure 7.7.3. Nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor after reverse Cuthill–McKee reordering; $nnz(A) = 4015$; $nnz(L) = 23,596$.

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 algorithm is best described using a graph model of the Cholesky factorization. At the same time the nonzero structure of the Cholesky factor L can be determined and its storage structure generated. The minimum degree algorithm has been subject to an extensive development and very efficient implementations now exist. For details we refer to George and Liu [239, Chapter 5] and [241].

Typically, the minimum degree ordering tend to give a scattered distribution of the nonzeros throughout the matrix. For many problems, such orderings can reduce fill by one or more orders of magnitude over the corresponding minimum envelope ordering. This is illustrated in Figure 7.7.4, which shows the structure of the matrix WW^T and its Cholesky factor after minimum degree reordering. The number of nonzero elements in the Cholesky factor is reduced to 9,911, a substantial reduction over the reverse Cuthill–McKee ordering.

In terms of the Cholesky factorization the minimum degree algorithm is equivalent to choosing the i th pivot column as one with the minimum number of nonzero elements in the unreduced part of the matrix. This will minimize the number of entries that will be modified in the next elimination step, and hence tend to minimize the arithmetic and amount of fill that occurs in this step. Although this, of course, will not in general provide a minimization of the *global* arithmetic or fill, it has proved to be very effective in reducing both of these objectives. The name “minimum degree” comes from the graph-theoretic formulation of the algorithm, which is given below.

Algorithm 7.15. Minimum Degree Ordering.

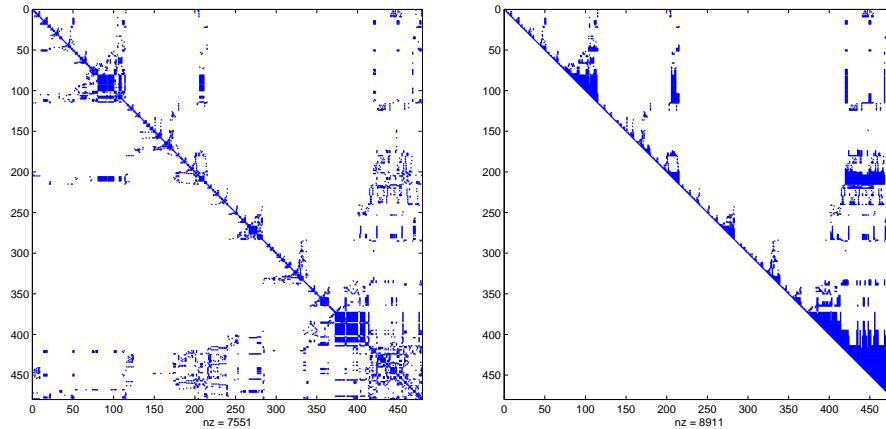


Figure 7.7.4. Nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor after minimum degree reordering. $nnz(A) = 4015$; $nnz(L) = 8,911$.

Let $G^{(0)} = G(A)$.

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

The way tie-breaking is done may have an important influence on the goodness of the ordering. One can, e.g., 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.

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 $G^{(i-1)}$, since 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 the degree of the remaining nodes in the set will decrease by one, and they all become of minimum degree. This allows us to eliminate all nodes in Y simultaneously and perform the graph transformation

and node update only once. Indeed, indistinguishable nodes can in the minimum degree algorithm be merged and treated as one **supernode**. With this and other 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.

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 we set $c = 1$, $k = 0$. For one-off problems of moderate size thresholds like $1.5d + 2$ can be used. The default in MATLAB is $1.2d + 1$.

Nested Dissection Ordering

The minimum degree algorithm is a local ordering strategy. We now describe an ordering algorithm called **nested dissection**, which takes a global view of the graph.

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 corresponding to a symmetric matrix $A \in \mathbf{R}^{n \times n}$. 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 nodes X_1 and X_2 . If the separator nodes are ordered last after those of X_1 and X_2 then the correspondingly ordered matrix A has the block form

$$A = \begin{pmatrix} A_1 & 0 & S_1 \\ 0 & A_2 & S_2 \\ S_1 & S_2 & A_3 \end{pmatrix}, \quad L = \begin{pmatrix} L_1 & & \\ 0 & L_2 & \\ L_{31} & L_{32} & L_3 \end{pmatrix}. \quad (7.7.9)$$

The important observation is that the zero blocks are preserved during the factorization. The dissection process can now be repeated on the two subgraphs $G(A_1)$ and $G(A_2)$ and again the separator nodes 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 operator on a square regular two-dimensional grid. Let the grid be dissected into k congruent rectangles, by separators parallel to the y -axis, $k \gg 1$. Permuting the separator variables 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 $S = R - Q^T P Q$ will have a block tridiagonal form, which is important for the efficiency of this scheme. For $k = n$ it can be shown that the bandwidth of S is $2n^2$. The operation count for the solution of this system equals n^7 , which compares with n^8 for the original ordering.

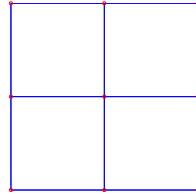


Figure 7.7.5. A 3×3 element in a Regular grid.

We will illustrate nested dissection on a regular two-dimensional grid G of size $n \times n$ on a square. This is built up by 3×3 elements where the nodes are connected as shown in Figure 7.7.5, the so-called 9-point stencil. With such an $(n+1)$ by $(n+1)$ grid is associated a symmetric positive definite linear system $Ax = b$ of dimension $N = (n+1)^2$. Each node x_i is associated with a node and $a_{ij} \neq 0$ if and only if the nodes x_i and x_j belong to the same subsquare.

Example 7.7.3.

Consider a regular 7×7 grid. A nested dissection ordering for the 49 nodes is shown below:

$$\begin{bmatrix} 35 & 39 & 32 & 49 & 14 & 18 & 11 \\ 36 & 38 & 33 & 48 & 15 & 17 & 12 \\ 34 & 37 & 31 & 47 & 13 & 16 & 10 \\ 42 & 41 & 40 & 46 & 21 & 20 & 19 \\ 26 & 30 & 23 & 45 & 5 & 9 & 2 \\ 27 & 29 & 24 & 44 & 6 & 8 & 3 \\ 25 & 28 & 22 & 43 & 4 & 7 & 1 \end{bmatrix}.$$

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 that a great improvement is achieved compared to the $O(n^3)$ storage and $O(n^4)$ operations required using the natural ordering.

Theorem 7.7.6 (George [235]).

The number of nonzeros in the Cholesky factor L of a matrix associated with a regular n by n grid ordered by nested dissection equals

$$nnz(L) = \frac{31}{4}n \log_2 n + O(n^2).$$

The number of operations required to compute L is given by

$$\frac{829}{84}n^3 + O(n^2 \log_2 n).$$

These results are optimal in the order of magnitude sense.

Nested dissection can also be applied to more general problems. A **planar graph** is a graph which can be drawn in a plane without two edges crossing. An algorithms for finding a separator of size $O(\sqrt{n})$ for a planar graph with n nodes that splits the graph into two approximately equal subgraphs is given in [411]. George and Liu [239, Chapter8] give an alternative algorithm based on level sets. For these orderings bounds $O(n^3)$ on operations and $O(n^2 \log_2 n)$ on storage can be guaranteed.

The Multifrontal Method

A significant improvement of the efficiency of sparse Cholesky factorization was realized by the introduction of the **multifrontal method** by Duff and Reid [179]. The name of the method comes from its relation to the frontal method of Irons [349] used in finite element calculations. In this context the two phases of the solution, namely the assembly of the matrix (from integral computations) and the factorization, were merged together. However, a variable can not be eliminated until it has been fully assembled.

The multifrontal method can be explained purely from the matrix point of view. It uses 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 multifrontal method can reorganize the numerical Cholesky factorization into a sequence of partial factorizations of dense smaller matrices. This can lead to large speedup on machines with vector and parallel architectures.

The choice of frontal and update matrices is governed by the elimination tree. The class of postorderings of this is particularly suitable. The following theorem due to Duff shows how to find independent tasks in multifrontal Cholesky factorization.

Theorem 7.7.7.

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 an effective but crude measure for the amount of work in parallel elimination.

7.7.6 LU Factorization of 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 stability of the elimination process. The choice of pivots cannot be determined from the structure of A , and the row interchanges will not be known until the numerical factorization is performed. Since the fill in the factors L and U depend on the choice of pivots. This means that the storage structure for L and U and the total size of storage needed can not be predicted in advance. Instead of having a separate symbolic phase as is the case for symmetric positive definite matrices, there needs to be a combined analyze-factorize step.

To find an ordering such that fill and the number of arithmetic operations are reduced, one has to rely on heuristic ordering algorithms. One of the most used ordering algorithms in the unsymmetric case is an ordering initially used by Markowitz [425] for linear programming problems. To motivate this, suppose that Gaussian elimination has proceeded through k stages and let $A^{(k)}$ be the remaining active submatrix. Denote by r_i the number of nonzero elements in the i th row and c_j is the number of nonzero elements in the j th column of $A^{(k)}$. In the Markowitz algorithm one performs a row and column interchange so that the product

$$(r_i - 1)(c_j - 1),$$

is minimized. (Some rules for tie-breaking are needed also.) This is equivalent to a *local minimization* of the fill at the next stage, assuming that all entries modified were zero beforehand. This choice minimizes also the number of multiplications required for this stage. Notice that if A is symmetric this is exactly the minimum degree algorithm. However, historically the Markowitz criterion predates the minimum degree algorithm. For a detailed discussion of the implementation of the Markowitz criterion, see Duff, Erisman, and Reid [176].

The Markowitz criterion may not give pivotal elements which are acceptable from the point of numerical stability. Usually a **threshold pivoting** scheme is used to minimize the reorderings. This means that the chosen pivot is restricted to elements that satisfy an inequality of the form

$$|a_{ij}^{(k)}| \geq \tau \max_r |a_{rj}^{(k)}|, \quad (7.7.10)$$

where τ , $0 < \tau \leq 1$, is a predetermined threshold value. A value of $\tau = 0.1$ is usually recommended as a good compromise between sparsity and stability. (Note that the usual partial pivoting strategy is obtained for $\tau = 1$.) The condition (7.7.10) 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. Furthermore, it is possible to monitor stability by computing the relative backward error, see Section 7.5.2. A widely used implementation based on the Markowitz strategy is Duff's Harwell code MA28.

It would be desirable to be able to produce a data structure for L and U that is large enough to contain the factors for *any choice of pivots* in the factorization.

Then the numerical factorization could be performed using a static storage scheme such as for the Cholesky factorization. This would also give a bound on the total storage and tell us if it is feasible to perform the factorization. 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 . Assume that A has a zero-free main diagonal. Then it has been shown by George and Ng [243] that the structure of the Cholesky factor L_C of $A^T A$ 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, we can use the structure of L_C to get an upper bound for the size of the LU factors. There are cases when this will produce a gross overestimate. For example, even if A is sparse the matrix $A^T A$ may not be sparse. There is an efficient algorithm due to George and Ng [244] to perform the symbolic factorization of $A^T A$ directly using only the structure of A . This removes the step of determining the structure of $A^T A$. This is also of interest for solving sparse least squares problem; see Section 8.5.5.

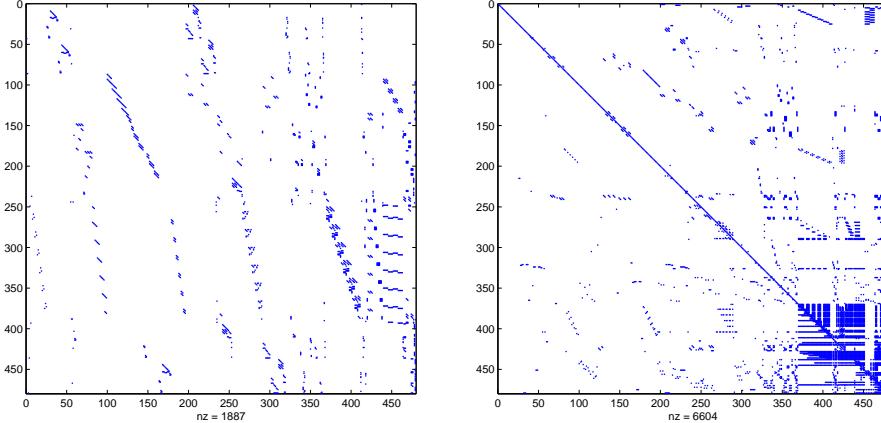


Figure 7.7.6. Nonzero pattern of the matrix W *west0479* and its LU factors after reordering by increasing column count; $\text{nnz}(W) = 1887$; $\text{nnz}(L+U) = 6604$.

Other sparse LU algorithms reorders only the columns for sparsity and perform partial pivoting by rows. For example, the LU factorization in MATLAB uses a sparse version of the column sweep method due to Gilbert and Peierls [249] (cf. Section 7.2.5). The LU factorization is computed a column of L and U at a time. 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. The total size of the factors cannot be predicted in advance. The routine makes an initial guess of the storage required,

and expands that storage by a factor of 1.5 whenever needed. The total time for this algorithm can be shown to be proportional to the number of arithmetic operations plus the size of the result.

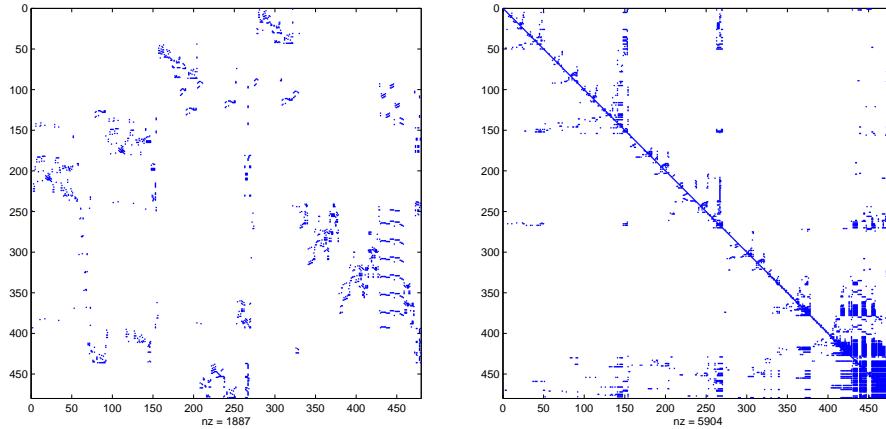


Figure 7.7.7. Nonzero pattern of the matrix *west0479* and its LU factors after reordering by and its LU factors after column minimum degree ordering; $nnz(W) = 1887$; $nnz(L + U) = 5904$.

A simple column ordering strategy is to *sort the columns by increasing column count*, i.e. by the number of nonzeros in each column. This can often give a substantial reduction of the fill in Gaussian elimination. Figure 7.7.6 shows the LU factorization of the matrix *W* reordered after column count and its LU factors. The number of nonzeros in *L* and *U* now are 6604, which is a substantial reduction.

In MATLAB this column ordering is computed by the function `p = colperm(A)`. This computation can be expressed very compactly as(see [247, Sec. 3.3.1]):

```
[i,j] = find(A);
[ignore,p] = sort(diff(find(diff([0 j' inf]))));
```

Here `[i,j] = find(A)` returns the row and column indices of the nonzero entries of *A*. The vector `y = diff(x)`, where *x* is a row or column vector, is a vector with entries equal to the first differences $x_2 - x_1, \dots, x_n - x_{n-1}$. Hence, the inner `diff` gives a vector with ones at the starts of columns and zero elsewhere. Hence, the outer `diff` converts this to a vector of column start indices and the outer `diff` then gives the vector of column length. Finally, `[y,p] = sort(x)` returns in *y* the entries of *x* sorted in ascending order and a permutation vector *p* such that $y = x(p)$.

The reverse Cuthill–MacKey ordering described for the Cholesky factorization can be used for unsymmetric matrices by applying it to $A + A^T$. This ordering sometimes performs well for matrices that come for one-dimensional problem or are somehow long and thin. 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 $\min_x \|Ax - b\|_2$; see Section 8.5.5. Minimum degree ordering often performs much better than `colperm` or `symrcm`. Results for this ordering applied to the matrix `west0479` are shown in Figure 7.7.7. The LU factors of the reordered matrix now contains only 5904 nonzeros.

MATLAB uses an implementation of the column minimum degree ordering that does not actually form the matrix $A^T A$. The code has the 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 $p = \text{symamd}(A)$ takes only time proportional to the size of A . For the many other features of this MATLAB algorithm we refer to [247, Sec. 3.3.1]).

7.7.7 Nonzero Diagonal and Block Triangular Form

Recall that a matrix A is called irreducible if and only if its graph is strongly connected. Otherwise it is reducible and can be permuted into block triangular form

$$PAQ = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1,t} \\ & A_{22} & \dots & A_{2,t} \\ & & \ddots & \vdots \\ & & & A_{tt} \end{pmatrix} \quad (7.7.11)$$

with square irreducible diagonal blocks A_{11}, \dots, A_{tt} , $t > 1$. The off-diagonal blocks are possibly nonzero matrices of appropriate dimensions. In the symmetric positive definite case a similar reduction to block upper triangular form can be considered, where $Q = vP^T$. Some authors reserve the term reducible for this case, and use the terms bi-reducible and bi-irreducible for the general case.

Before performing a factorization of a sparse matrix it can be advantageous to perform some preprocessing. For an arbitrary square nonsingular matrix $A \in \mathbf{R}^{n \times n}$ there always is a row permutation P such that PA has nonzero elements on its diagonal. Further, there is a row permutation P and column permutation Q such that PAQ has a nonzero diagonal and the block triangular form (7.7.11). Using this structure a linear system $Ax = b$ or $PAQy = c$, where $y = Q^T x$, $c = Pb$, reduces to

$$A_{ii}y_i = c_i - \sum_{j=i+1}^n A_{ij}x_j, \quad j = n : -1 : 1. \quad (7.7.12)$$

Hence, we only need to factorize the diagonal blocks A_{ii} , $i = 1 : n$, and then use block back substitution. This can lead to significant savings.

If the diagonal blocks are required to be irreducible, then the block triangular form (7.7.11) can be shown to be essentially unique. Any one 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 a symmetric permutations that reorder the blocks.

An arbitrary rectangular matrix $A \in \mathbf{R}^{m \times n}$ has a block triangular form called the **Dulmage–Mendelsohn form**. If A is square and nonsingular this is the form (7.7.11). The general case is based on a canonical decomposition of bipartite graphs

discovered by Dulmage and Mendelsohn. In the general case the first diagonal block may have more columns than rows, the last diagonal block more rows than column. All the other diagonal blocks are square and nonzero diagonal entries. This block form can be used for solving least squares problems by a method analogous to back substitution; see Section 8.5.4.

The **bipartite graph** associated with A is denoted by $G(A) = \{R, C, E\}$, where $R = (r_1, \dots, r_m)$ is a set of vertices corresponding to the rows of A and $C = (c_1, \dots, c_m)$ a set of vertices corresponding to the columns of A . E is the set of edges, and $\{r_i, c_j\} \in E$ if and only if $a_{ij} \neq 0$. A **matching** in $G(A)$ is a subset of its edges with no common end points. In the matrix A this corresponds to a subset of nonzeros, no two of which belong to the same row or column. A **maximum matching** is a matching with a maximum number $r(A)$ of edges. The **structural rank** of A equals $r(A)$. Note that the mathematical rank is always less than or equal to its structural rank. For example, the matrix

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

has structural rank 2 but numerical rank 1.

For the case when A is a structurally nonsingular matrix there is a two-stage algorithm for permuting A to block upper triangular form. see Tarjan [562]. The implementation of this is discussed by Duff and Reid [178]. In the first stage of the program MC13D a maximum matching in the bipartite graph $G(A)$ with row set R and column set C is found. In the second stage the block upper triangular form of each submatrix determined from the strongly connected components in the graph $G(A)$, with edges directed from columns to rows.

If A has structural rank n but is *numerically* rank deficient it will not be possible to factorize all the diagonal blocks in (7.7.11). In this case the block triangular structure given by the Dulmage–Mendelsohn form cannot be preserved, or some blocks may become severely ill-conditioned.

Review Questions

- 7.1** Describe the coordinate form of storing a sparse matrix. Why is this not suitable for performing the numerical LU factorization?
- 7.2** Give an example of a sparse matrix A , which suffers extreme fill in Gaussian elimination.
- 7.3** Describe the Markowitz criterion for ordering rows and columns of a nonsymmetric matrix before factorization.
- 7.4** Describe threshold pivoting. Why is this used instead of partial pivoting in some schemes for LU factorization?
- 7.5** What does the reverse Cuthill–McKee ordering minimize?

Problems

- 7.1** Let $A, B \in \mathbf{R}^{n \times n}$ be sparse matrices. Show that the number of multiplications to compute the product $C = AB$ is $\sum_{i=1}^n \eta_i \theta_i$, 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 formulation $C = \sum_{i=1}^n a_i b_i^T$.

- 7.2** (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 thus:

for $k = 1 : nx$

$$y(\text{index}(k)) = a * x(k) + y(\text{index}(k));$$

- (b) Give an efficient algorithm for computing the inner product of two compressed vectors.

- 7.3** The symmetric matrices

$$A = \begin{pmatrix} \times & \times & \times & \dots & \times \\ \times & \times & & & \\ \times & & \times & & \\ & & & \ddots & \\ \vdots & & & & \\ \times & & & & \times \end{pmatrix}, \quad PAP^T = \begin{pmatrix} \times & & & & \times \\ & \ddots & & & \vdots \\ & & \times & & \times \\ & & & \times & \times \\ \times & \dots & \times & \times & \times \end{pmatrix}.$$

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.

- (a) Show that if the $(1, 1)$ element in A is chosen as the first pivot the fill will be total and $O(n^3)$ flops required for its factorization. In contrast, for PAP^T there is no fill except in the last step, when pivots are chosen in natural order. Only about $O(n)$ flops are required to perform this factorization.

- (b) Show that the graph $G(A)$ is a tree.

- 7.4** Use the graph model of Cholesky factorization (see Algorithm 7.17) to find the filled graph of the matrix A given in (7.7.4). Verify your result by comparing with the graph $G(L + L^T)$, where L is the Cholesky factor given in (7.7.4).

- 7.5** (a) Let A be 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 then the band structure of A is destroyed. There will now be some fill in the Cholesky factorization. How is the graph changed? Show that if the row 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.

7.6 Consider a matrix with the symmetric structure

$$A = \begin{pmatrix} \times & & \times & & \\ & \times & & \times & \times \\ \times & & \times & \times & \\ & \times & \times & & \\ & & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix}.$$

- (a) What is the envelope of A ? Where will fill occur during Gaussian elimination?
- (b) Draw the undirected graph G that represents the sparsity structure of A .

7.8 Structured Linear Equations

The coefficient matrices in systems of linear equations arising from signal processing, control theory and linear prediction are often dense but highly structured so that the matrix is defined by only $O(n)$ quantities. In many cases the special structure can be taken advantage of. Such structured systems can often be solved in $O(n^2)$ operations rather than $O(n^3)$ otherwise required by Gaussian elimination. This has important implications for many problems in signal restoration, acoustics, seismic exploration and many other application areas. Sometimes super-fast methods exist, which take only $O(n \log n)$ operations. However, since the numerical stability of super-fast methods are either bad or unknown, we will not consider such methods in the following.

7.8.1 Toeplitz and Hankel Matrices

A **Toeplitz**²⁷ matrix $T = (t_{i-j})_{1 \leq i,j \leq n}$ is a matrix whose entries are constant along each diagonal;

$$T_n = \begin{pmatrix} t_0 & t_1 & \cdots & t_{n-1} \\ t_{-1} & t_0 & \cdots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{-n+1} & t_{-n+2} & \cdots & t_0 \end{pmatrix} \in \mathbb{C}^{n \times n}. \quad (7.8.1)$$

T_n is defined by the $2n - 1$ values of $t_{-n+1}, \dots, t_0, \dots, t_{n-1}$. Toeplitz matrices are fundamental in signal processing and time series analysis. The structure reflects the invariance in time or in space. Toeplitz matrices arise also directly from partial differential equations with constant coefficients and from discretizations of integral equations with convolution kernels. Positive definite Toeplitz matrices arise as covariance matrices of stationary random processes.

²⁷Otto 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 is now known as Toeplitz operators. He emigrated to Palestine in 1939.

A **Hankel matrix** is a matrix, whose elements are constant along each antidiagonal, i.e., $H = (h_{i+j-2})_{1 \leq i,j \leq n}$

$$H = \begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \cdots & h_n \\ \vdots & \vdots & & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{pmatrix} \in \mathbf{C}^{n \times n}.$$

Note that H is a complex symmetric matrix. It should be noted that complex symmetric matrices have special properties; see [338, Sec. 4.4].

Reversing the rows (or columns) of a Toeplitz matrix gives a Hankel matrix. That if $J = (e_n, e_{n-1}, \dots, e_1)$, then JT (and TJ) are Hankel matrices, and vice versa. Methods developed for solving Hankel systems thus apply also to Toeplitz systems. Fast algorithms can be developed for the sum of a Toeplitz and a Hankel matrix.

Toeplitz linear systems arising in applications are often large, and dimensions of 10,000 or more are not uncommon. The original matrix T only requires $2n - 1$ storage. However, the inverse of a Toeplitz matrix is not Toeplitz, and if standard factorization methods are used, at least $n(n + 1)/2$ storage is needed. Special algorithms which exploit the Toeplitz structure are much faster and require only $O(n^2)$ flops and $O(n)$ storage.

We now describe the **Levinson–Durbin algorithm** (see [406, 181]) that solves a Toeplitz linear system $T_n x = y$ in about $2n^2$ flops. We assume that all principal minors 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 satisfy as solutions of the linear systems

$$T_k f_k = e_1, \quad T_k b_k = e_k, \quad k = 1 : n, \quad (7.8.2)$$

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 the leading principal submatrix of T_k , we have

$$T_k \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} = \begin{pmatrix} e_1 \\ \epsilon_k \end{pmatrix}, \quad \epsilon_k = T_k e_k \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix}. \quad (7.8.3)$$

Similarly, for the backward vectors we have the recursion.

$$T_k \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} = \begin{pmatrix} \delta_k \\ e_{k-1} \end{pmatrix}, \quad \delta_k = T_k^T e_k \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix}. \quad (7.8.4)$$

If we now take a linear combination of these two equations

$$T_k \left(\alpha \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} \right) = \alpha \begin{pmatrix} e_1 \\ \epsilon_k \end{pmatrix} + \beta \begin{pmatrix} \delta_k \\ e_{k-1} \end{pmatrix}.$$

and α and β are chosen so that the right-hand side becomes e_1 , this will give us the forward vector f_k . Similarly, if α and β are 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 satisfy the 2×2 linear system

$$\begin{pmatrix} 1 & \delta_k \\ \epsilon_k & 1 \end{pmatrix} \begin{pmatrix} \alpha_f & \alpha_b \\ \beta_f & \beta_b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \quad (7.8.5)$$

If $\delta_k \neq 1$ 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 built up as follows. Assume the vector $x^{(k-1)} \in \mathbf{R}^{k-1}$ satisfies the first $k-1$ equations and write

$$T_k \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{k-1} \\ \eta_k \end{pmatrix}, \quad \eta_k = T_k^T e_k \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix}. \quad (7.8.6)$$

Then the backward vector b_k can be used to modify the last element in the right-hand side. The solution can be built up using 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 $x^{(k)}$ are needed.

When the Toeplitz matrix is symmetric there are important simplifications. Then from (7.8.4)–(7.8.4) it follows that the backward and forward vectors are the row-reversals of each other, i.e.

$$b_k = J_k f_k, \quad J_k = (e_k, e_{k-1}, \dots, e_1).$$

Further, $\epsilon_k = \delta_k$, so the auxiliary 2×2 subsystems (7.8.5) are symmetric. Taking this into account roughly halves the operation count and storage requirement.

It is important to note that even if the Toeplitz matrix T_n is nonsingular, its principal minors 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 minor T_2 is singular. Likewise, if T_n is well-conditioned, its principal minors 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 [129].

7.8.2 Cauchy-Like Matrices

A **Cauchy matrix** is of the form:

$$C(x, y) = \left(\frac{1}{y_i - z_j} \right)_{1 \leq i, j \leq n} = \begin{pmatrix} \frac{1}{y_1 - z_1} & \cdots & \frac{1}{y_1 - z_n} \\ \vdots & \ddots & \vdots \\ \frac{1}{y_n - z_1} & \cdots & \frac{1}{y_n - z_n} \end{pmatrix}, \quad (7.8.7)$$

where we assume that $y_i \neq z_j$ for $1 \leq i, j \leq n$. Cauchy matrices are encountered in many applications, including solution of integral equations, particle simulation, various rational interpolation problems, the pole placement problem in system theory, etc.

Many algebraic properties of Cauchy matrices are similar to those of Vandermonde matrices. The problem of solving the associated linear Cauchy system was treated in 1841 by Cauchy [90], who gave the following well-known explicit expression for the determinant

$$\det(C) = \frac{\prod_{1 \leq i < j \leq n} (y_j - y_i)(z_j - z_i)}{\prod_{1 \leq i \leq j \leq n} (y_j - z_i)}.$$

A well known example of a Cauchy matrix is the **Hilbert matrix** $H_n \in \mathbf{R}^{n \times n}$ with $h_{ij} = 1/(i + j - 1)$, obtained by taking $y_i = z_i = i - 1/2$. For example,

$$H_4 = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{pmatrix}.$$

The Hilbert matrix is symmetric and positive definite Hankel matrix. It is also **totally positive**. The inverse of H_n is known explicitly and has integer elements. Hilbert matrices of high order are known to be very ill-conditioned; for large n it holds that $\kappa_2(H_n) \sim e^{3.5n}$.

Example 7.8.1.

Cauchy matrices are related to rational interpolation. Given distinct points x_i , $i = 1 : n$ and numbers f_i , $i = 1 : n$, find the coefficients of a rational function

$$r(x) = \sum_{j=1}^n a_j \frac{1}{x - y_j},$$

which satisfies the interpolation conditions $r(x_i) = f_i$, $i = 1 : n$. With $a = (a_1, \dots, a_n)$, $f = (f_1, \dots, f_n)$ this leads to the linear system $Ca = f$, where C is the Cauchy matrix in (7.8.7).

Any row or column permutation of a Cauchy matrix is again a Cauchy matrix. This property allows fast and stable version of Gaussian elimination with partial pivoting to be developed for Cauchy systems. These methods, first suggested by Heinig [315], apply also in the more general case of **Cauchy-like** of the form

$$C = \left(\frac{a_i^T b_j}{y_i - z_j} \right)_{1 \leq i, j \leq n}, \quad a_i, b_j \in \mathbf{R}^r, \quad r \leq n. \quad (7.8.8)$$

A Cauchy-like matrix can alternatively be defined to be the solution of the **displacement equation**

$$\Omega C - C\Lambda = G \quad (7.8.9)$$

with $G = A^T B$, and

$$A = (a_1, \dots, a_n) \in \mathbf{R}^{r \times n}, \quad B = (b_1, \dots, b_n) \in \mathbf{R}^{r \times n}, \\ \Omega = \text{diag}(\omega_1, \dots, \omega_n), \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n).$$

The matrix pair (A, B) is the generator of C and r the **displacement rank**. C is said to have a displacement structure with respect to Ω and Λ if $r \ll n$.

The equation (7.8.9) 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 10.1.19. Multiplying (7.8.9) 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 .

A Cauchy-like system can be solved in $O(n^2)$ flops using Gaussian elimination with partial pivoting. The first step is to zero out elements in the first column below the diagonal

$$C = \begin{pmatrix} \gamma_1 & u_1^T \\ r_1 & C_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l_1 & I \end{pmatrix} \begin{pmatrix} 1 & u_1^T \\ 0 & S_2 \end{pmatrix} \quad (7.8.10)$$

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 .

The following result shows that a Toeplitz matrix can be transformed into Cauchy-like matrix with using fast Fourier or transforms (requiring only $O(n \log n)$ flops).

Theorem 7.8.1.

Let C be a matrix with $\gamma_1 \neq 0$ that satisfies the displacement equation (7.8.9) 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

$$\tilde{A}_2 = A_2 - a_1 l_1^T, \quad \tilde{B}_2 = B_2 - b_1 u_1^T / \gamma_1.$$

The first step of Gaussian elimination involves computing the first row and column γ_1 , r_1 , and u_1 of C from (7.8.9) 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

$$Z_\delta = \begin{pmatrix} 0 & 0 & \dots & 0 & \delta \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & \ddots & & \vdots \\ 0 & 1 & \ddots & & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}$$

and let $\Omega = Z_1$ and $\Lambda = Z_{-1}$. Then every Toeplitz matrix satisfies the displacement equation (7.8.9) with G having nonzero entries only in its first row and last column. Indeed, it is easy to verify that for the Toeplitz matrix (7.8.1)

$$Z_1 T - TZ_1 = e_1 \begin{pmatrix} t_{n-1} - t_{-1} \\ \vdots \\ t_1 - t_{-n+1} \\ t_0 \end{pmatrix}^T + \begin{pmatrix} t_0 \\ t_{-n+1} + t_1 \\ \vdots \\ t_{-1} + t_{n-1} \end{pmatrix} e_n^T, \quad (7.8.11)$$

so the displacement rank is at most equal to 2.

Theorem 7.8.2 (Gohberg, Kailath, and Olshevsky [259, Prop. 3.1]).

Let $T \in \mathbf{C}^{n \times n}$ by 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

$$\mathcal{D}_1 C - C \mathcal{D}_{-1} = (\mathcal{F} A^H)(B \mathcal{D}_0 \mathcal{F}^H),$$

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 discrete Fourier transform 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

$$Z_1 = \mathcal{F}^H D_1 \mathcal{F}, \quad Z_{-1} = D_0^{-1} \mathcal{F}^H D_{-1} \mathcal{F} D_0.$$

□

The GKO algorithm of Gohberg, Kailath, and Olshevsky [259] uses this transformation and fast Gaussian elimination with partial pivoting to solve Toeplitz linear systems. Toeplitz-plus-Hankel matrices can be shown to have displacement rank $r \leq 4$. Fast algorithms for such matrices can be developed using similar techniques. see Gu [288].

7.8.3 Vandermonde systems

In Chapter 4 the problem of interpolating given function values $f(\alpha_i)$, $i = 1 : n$ at distinct points α_i with a polynomial of degree $\leq n - 1$ was shown to lead to a linear system of equations with matrix $M = [p_j(\alpha_i)]_{i,j=1}^m$. In the case of the power basis $p_j(z) = z^{j-1}$, the matrix M equals V^T , where V is the **Vandermonde matrix**

$$V = [\alpha_j^{i-1}]_{i,j=1}^n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{pmatrix}. \quad (7.8.12)$$

Hence, the unique polynomial $P(z)$ satisfying the interpolating conditions $P(\alpha_i) = f_i$, $i = 1, \dots, n$ is given by

$$P(z) = (1, z, \dots, z^{n-1})a,$$

where a is the solution of the dual Vandermonde system.

$$V^T a = f \quad (7.8.13)$$

An efficient algorithm for solving primal and dual Vandermonde systems is the **Björck–Pereyra algorithm** described in Volume I, Section 3.5.4. It is related to Newton's interpolation method for determining the polynomial $P(x)$, and solves primal and dual Vandermonde systems in $\frac{5}{2}n(n + 1)$ flops. No extra storage is needed.

The Björck–Pereyra algorithms give almost full relative accuracy in the solution of some Vandermonde systems which are so ill-conditioned that Gaussian elimination with complete pivoting fails to produce a single correct digit. A forward error analysis given by Higham [324] explains the surprisingly favorable results. If the points are positive and monotonically ordered,

$$0 < x_1 < x_2 < \cdots < x_n, \quad (7.8.14)$$

then the error in the solution \bar{a} of a Vandermonde system $Vy = b$ computed by the primal algorithm can be bounded as

$$|\bar{a} - a| \leq 5u|V^{-1}| |b| + O(u^2). \quad (7.8.15)$$

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

$$|\bar{a} - a| \leq 5u|a| + O(u^2); \quad (7.8.16)$$

i.e., the solution is computed with small relative error independent of the conditioning of V . A similar result holds for the dual algorithm. These good results can be shown to be related to the fact that when (7.8.14) holds, the matrix $V(x_1, x_2, \dots, x_n)$ is **totally positive**, i.e., the determinant of every square submatrix of V is positive; see [230, 74] and [151].

Fast Björck–Pereyra-type algorithms for **Vandermonde-like** matrices of the form

$$P = (p_i(\alpha_j))_{i,j=1}^n,$$

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 [328, Sec. 22.2]. Cauchy linear systems $Cx = b$, where 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, Kailath, and Olshevsky [74].

7.8.4 Semiseparable Matrices

Semiseparable matrices were first defined as inverses of unreducible tridiagonal matrices. In Section 7.4.5 it was shown that the elements of such a matrix $S = T^{-1}$ can be written in the form.

$$s_{ij} = \begin{cases} u_i^T v_j & \text{if } 1 \leq i \leq j \leq n, \\ p_i^T q_j & \text{if } 1 \leq j < i \leq n, \end{cases} \quad (7.8.17)$$

Thus, S can be represented by order $O(n)$ information. We can use MATLAB notations and write $\text{triu}(A, k)$ for a matrix that is identical to the matrix 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. Using this notation a semiseparable matrix can be written as

$$S = \text{triu}(uv^T, 0) + \text{tril}(pq^T, -1). \quad (7.8.18)$$

The vectors u and v are called the **generators** of S .

Lemma 7.8.3.

Let $S \in \mathbf{R}^{n \times n}$ be the semiseparable matrix (7.8.18). Then the matrix-vector product Sx can be computed in $7n$ flops.

Proof. Write $Sx = y + z$, where

$$y = \text{triu}(uv^T, 0)x, \quad z = \text{tril}(pq^T, -1)x.$$

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

A more general class of semiseparable matrices is the following: S is a semiseparable matrix of semiseparability rank r if there exist two matrices R_1 and R_2 both of rank r such that

$$S = \text{triu}(UV^T) + \text{tril}(PQ^T).$$

where U , V , P , and Q are $n \times r$ matrices.

Semiseparable matrices or matrices which 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 be computed in $O(nr^2)$ flops; see Gohberg, Kailath, and Koltracht [258]. However, when A is not symmetric positive definite, stable methods for computing the LU factorization are not known.

Consider the problem of solving a linear system $Ax = b$, where the matrix A is the sum of a diagonal and a semiseparable rank one matrix

$$A = D + \text{triu}(uv^T, 1) + \text{tril}(pq^T, -1).$$

or for ($n = 5$)

$$A = \begin{pmatrix} d_1 & u_1 v_2 & u_1 v_3 & u_1 v_4 & u_1 v_5 \\ p_2 q_1 & d_2 & u_2 v_3 & u_2 v_4 & u_2 v_5 \\ p_3 q_1 & p_3 q_2 & d_3 & u_3 v_4 & u_3 v_5 \\ p_4 q_1 & p_4 q_2 & p_4 q_3 & d_5 & u_4 v_5 \\ p_5 q_1 & p_5 q_2 & p_5 q_3 & p_5 q_4 & d_5 \end{pmatrix}.$$

Chandrasekaran and Gu [104] have given an algorithms for solving linear systems where the matrix is the sum of a band and a semiseparable rank r matrix that has complexity $O(nr^2)$ flops. The key is to compute a two-sided decomposition of the form

$$A = WLH,$$

where L is lower triangular and W and H are products of Givens matrices. Following [104] W_0 is taken to be a such that

$$W_1 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ \tilde{u}_2 \end{pmatrix}.$$

When forming $W_1 A$ the first two rows of A are transformed into

$$\begin{pmatrix} d_{11} & d_{12} & 0 & 0 & 0 \\ d_{21} & d_{22} & \tilde{u}_2 v_3 & \tilde{u}_2 v_4 & \tilde{u}_2 v_5 \end{pmatrix}.$$

The transformation is also applied to the rhs to get $\tilde{b} = W_1 b$. Next a Givens transformation H_1 is applied from the right so that

$$(d_{11} \quad d_{12}) H_1 = (\tilde{d}_{11} \quad 0).$$

The linear system $Ax = b$ now becomes

$$(W_1 A H_1) H_1^T x = W_1 b.$$

where the transformed matrix has the form

$$A_1 = W_1 A H_1 = \begin{pmatrix} \tilde{d}_{11} & 0 & 0 & 0 & 0 \\ \tilde{d}_{21} & \tilde{d}_{22} & \tilde{u}_2 v_3 & \tilde{u}_2 v_4 & \tilde{u}_2 v_5 \\ p_3 \tilde{q}_1 & p_3 \tilde{q}_2 & d_3 & u_3 v_4 & u_3 v_5 \\ p_4 \tilde{q}_1 & p_4 \tilde{q}_2 & p_4 q_3 & d_5 & u_4 v_5 \\ p_5 \tilde{q}_1 & p_5 \tilde{q}_2 & p_5 q_3 & p_5 q_4 & d_5 \end{pmatrix}.$$

Using the first equation we can now solve for the first component in $\tilde{x} = H_1^T x$ and hence $x = H_1 \tilde{x}$. Striking out the first column and row of the system we are left with a reduced system for the remaining components of \tilde{x} . This system has the same form as the original system. Hence, the reduction can be continued until a 2×2 system is obtained, which can be solved directly. The solution of the original system is then found from

$$x = H_1 H_2 \cdots H_{n-2} x.$$

The method is backward stable since only orthogonal transformations have been used. We stress that the full matrix is never formed. Only the diagonal elements and the generators u, v, p and q are transformed. Therefore, both the storage requirement and operation count is of order $O(n)$.

Review Questions

- 8.1** (a) What characterizes a Toeplitz matrix and how many numbers are needed to specify it?
 (b) How are Hankel matrices related to Toeplitz matrices?
 (c) Is the inverse of a Toeplitz matrix also Toeplitz?
- 8.2** Approximately how many flops are needed to solve a Toeplitz system $T_n x = y$ by the Levinson–Durbin method?
- 8.3** The Hilbert matrix with elements $h_{i,j} = 1/(i+j+1)$ is a both a Hankel and a Cauchy matrix. What is the general form of a Cauchy-like matrix?

Problems

- 8.1** (a) Show that the inverse of a Toeplitz matrix is persymmetric.

- (b) Show that if a matrix M is both symmetric and persymmetric all elements are defined by those in a wedge, as illustrated below 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 equals $n^2/4 + n/2$.

- 8.2** (a) Let the matrix $W \in \mathbf{R}^{(n+1) \times n}$ have the QR factorization

$$W = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Show that R is the diagonal plus semiseparable matrix

$$R = \text{diag}(R) + \text{triu}(qv^T),$$

where the last row of Q equals $e_{n+1}^T Q = (q, \gamma)$.

- (b) Show that the upper triangular system $Rx = b$, where R has the structure in (a) can be solved in $O(n)$ arithmetic operations.

Notes and Further References

The literature on linear algebra is extensive. The classic survey of matrix computations by Golub and Van Loan [277] was originally based on lectures by the authors at Johns Hopkins University. This book has become a bestseller and been thoroughly revised several times. It should be within reach of anyone interested in matrix computations. The two books by Stewart [549] on basic decompositions and [549] on eigensystems are the first in a planned multivolume series on the state of the art. Two other modern textbooks that can be highly recommended are Demmel [145] and Watkins [602]. Higham [328] is superbly written book, which is indispensable source book for anyone seriously interested in the accuracy and stability of matrix algorithms. A very complete treatment of perturbation theory and related topics is found in Stewart and Sun [552], which contains an elegant treatise on norms and metrics and also many historical comments.

Marcus and Minc [423] is a nice and handy survey of matrix theory and matrix inequalities. An influential and style setting text is Householder [343]. A newer authoritative treatise and an excellent source of information are the two volumes by Horn and Johnson [338] and [339]. Many nonstandard topics are covered in Gantmacher [227, 228] and in Lancaster and Tismenetsky [390]. Bellman [41] is an original and readable complementary text.

Section 7.1

The development of the theory of matrices in the 19th century is treated by Hawkins [313]. An interesting account of numerical methods in linear algebra before the advent of computers can be found in Faddeev and Faddeeva [195], but many of the methods treated here are now dated. Bellman [41] and Gastinel [231] are other readable complementary texts. A text that is still very useful on matrix computation is Stewart [543].

The Schur complement was so named by Emilie Haynsworth in 1968, and the name was chosen because of a lemma in the paper [526]. An interesting historical account of the Schur complement is given in [624]. The Banachiewicz inversion formula was introduced in 1937.

Section 7.2

Although the history of Gaussian elimination goes back at least to Chinese mathematicians about 250 B.C., there was no practical experience of solving large linear systems until the advent of computers in the 1940s. Gaussian elimination was the first numerical algorithm to be subjected to a rounding error analysis. In 1946 there was a mood of pessimism about the stability of Gaussian elimination. In 1943 Hotelling [340] had 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 J. von Neumann and Goldstein [450] (reprinted in [563, pp. 479–557]) published more relevant error bounds. In 1948 Turing wrote a remarkable paper [580], where he formulated the LU factorization and introduced matrix condition numbers. In 1951 a symposium on “Simultaneous Linear Equations and Determination of Eigenvalues” was held on UCLA campus. One of the principal papers was delivered by George Forsythe, who had assembled and organized a list of roughly 500 references [212]. A detailed historical perspective of Gaussian elimination is given by Higham [328, Sec. 9.13].

Rook pivoting for nonsymmetric matrices was introduced by Neal and Poole in [447]; see also [448]. Related pivoting strategies for symmetric indefinite matrices had been used earlier by Fletcher [206].

The Matrix Computation Toolbox by N. J. Higham is a collection of MATLAB m-files containing functions for constructing test matrices, computing matrix factorizations, visualizing matrices and other miscellaneous functions is available from the Web; see [328, Appendix D].

Section 7.3

The idea of doing only half the elimination for symmetric systems, while preserving symmetry is probably due to Gauss, who first sketched his elimination algorithm in 1809. The Cholesky method is named after the French military officer Andre-Louis Cholesky. He devised his method in order to solve symmetric, positive definite system arising in a geodetic survey in Crete and North Africa just before World War I.

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 [85]. 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 in [326], where normwise backward stability is shown.

Section 7.4

The inverse of any band matrix has a special structure related to low rank matrices. The first to study inverses of general band matrices was Asplund [15]. A history of cyclic reduction and its applications are given by Gander and Golub [224].

Section 7.5

An introduction to the implementation of algorithms for vector and parallel computers is given by Dongarra et al. [167]. Many important practical details on implementation of algorithms can be found in the documentation of LINPACK and EISPACK software given in Dongarra et al. [164] and Smith et al. [536]. Software developments in linear algebra is surveyed in Dongarra and Ward [171] and Dongarra and Eijkhout [168].

LAPACK95 is a Fortran 95 interface to the Fortran 77 LAPACK library documented in [10]. It is relevant for anyone who writes in the Fortran 95 language and needs reliable software for basic numerical linear algebra. It improves upon the original user-interface to the LAPACK package, taking advantage of the considerable simplifications that Fortran 95 allows. LAPACK95 Users' Guide [28] 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. For more information on LAPACK95 go to <http://www.netlib.org/lapack95/>.

Section 7.6

Bauer [39] in 1966 was the first to study componentwise perturbation theory. This did not catch on in English publications until Skeel took it up in two papers [533, 534]. Experiments by Poole and Neal [489] indicate that scaling can improve the solution if it makes the matrix more diagonal dominant.

Properties and applications of the Kronecker product are surveyed in [589]. For a review of the vec operator and Kronecker products see [318].

Section 7.7

The use of graphs to symbolically model Gaussian elimination for a symmetric positive definite matrix $A \in \mathbf{R}^{n \times n}$ is due to Parter [483]. A symmetric version of a strategy proposed by Markowitz [425] and called S2 was used by Tinney and Walker [565] in the analysis of power systems. Rose [505] subsequently developed a graph-theoretic model of the algorithm and named it the minimum degree algorithm. Perhaps the first to formally define the elimination tree was Schreiber [523]. An exhaustive treatment of their role in sparse Cholesky factorization is given by Liu [413]. George and Liu [241] survey the extensive development of efficient versions of the minimum degree algorithm. For an exposition of multifrontal methods we refer to Liu [414]. Structure prediction in sparse matrix computations is surveyed by Gilbert [246].

Most implementations of sparse Cholesky factorization use some variant of the column sweep scheme. They include the Harwell series of routines, the Waterloo SPARSPAK [238] and the Yale sparse matrix package [185].

Direct methods for sparse symmetric positive definite systems are treated in the book by George and Liu [239]. Duff, Erisman, and Reid [176] treat also the unsymmetric case. A good, more recent survey, is given by Duff [174]. The current state of the art is found in Davis [140]. This excellent monograph demonstrates a wide range of sparse matrix algorithms in a concise code. Available at <http://www.cise.ufl.edu/research/sparse/codes> is an up to date overview of available software with links to high performance sparse LU, Cholesky, and QR factorizations codes. An extensive comparison of sparse direct symmetric solvers on a wide range of problems from real applications are documented in Scott and Hu [528].

Section 7.8

Norbert Wiener and A. N. Kolmogorov independently analyzed the continuous case for linear filtering in 1941. In 1947 N. Levinson considered the discrete case, which yields a Toeplitz system of linear equations to solve. He gave an $O(n^2)$ algorithm, which was later improved by Durbin and others. Trench [578] has given an $O(n^2)$ algorithm for computing the inverse of a Toeplitz matrix.

A number of “superfast” methods for solving Toeplitz systems have been devised, e.g. the Schur method of Ammar and Gragg [7]. 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 [83]. The theory of displacement structures and their applications are surveyed by Kailath and Sayed [364].

A comprehensive overview of mathematical and numerical properties of semiseparable matrices is presented in [590]. Several alternative definitions of semiseparable matrices are discussed in [581].

Chapter 8

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 Maria Legendre, Nouvelles méthodes pour la détermination des orbites des comètes. Paris 1805

8.1 Preliminaries

8.1.1 The Least Squares Principle

A fundamental task in scientific computing is to estimate parameters in a mathematical model from collected data which are subject to errors. The influence of the errors can be reduced by using a greater number of data than the number of unknowns. If the model is linear, the resulting problem is then to “solve” an in general inconsistent linear system $Ax = b$, where $A \in \mathbf{R}^{m \times n}$ and $m \geq n$. In other words, we want to find a vector $x \in \mathbf{R}^n$ such that Ax is in some sense the “best” approximation to the known vector $b \in \mathbf{R}^m$.

There are many possible ways of defining the “best” solution to an inconsistent linear system. A choice which can often be motivated for statistical reasons (see Theorem 8.1.6) and leads also to a simple computational problem is the following: Let x be a vector which minimizes the Euclidian length of the **residual vector** $r = b - Ax$; i.e., a solution to the minimization problem

$$\min_x \|Ax - b\|_2, \quad (8.1.1)$$

where $\|\cdot\|_2$ denotes the Euclidian vector norm. Note that this problem is equivalent to minimizing the sum of squares of the residuals $\sum_{i=1}^m r_i^2$. Hence, we call (8.1.1) a **linear least squares problem** and any minimizer x a **least squares solution** of the system $Ax = b$.

Example 8.1.1.

Consider a model described by a scalar function $y(t) = f(x, t)$, where $x \in \mathbf{R}^n$ is a parameter vector to be determined from measurements (y_i, t_i) , $i = 1 : m$, $m > n$. In particular, let $f(x, t)$ be *linear* in x ,

$$f(x, t) = \sum_{j=1}^n x_j \phi_j(t).$$

Then the equations $y_i = \sum_{j=1}^n x_j \phi_j(t_i)$, $i = 1 : m$ form an overdetermined system, which can be written in matrix form $Ax = b$, where $a_{ij} = \phi_j(t_i)$, and $b_i = y_i$.

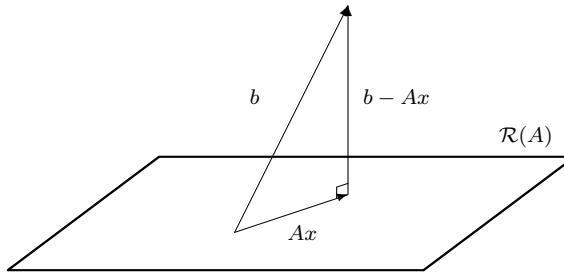


Figure 8.1.1. Geometric characterization of the least squares solution.

We shall see that a least squares solution x is characterized by $r \perp \mathcal{R}(A)$, where $\mathcal{R}(A)$ the range space of A . The residual vector r is always uniquely determined and the solution x is unique if and only if $\text{rank}(A) = n$, i.e., when A has linearly independent columns. If $\text{rank}(A) < n$, we seek the unique least squares solution of minimum Euclidean norm.

We now show a necessary condition for a vector x to minimize $\|b - Ax\|_2$.

Theorem 8.1.1.

Given the matrix $A \in \mathbf{R}^{m \times n}$ and a vector $b \in \mathbf{R}^m$. The vector x minimizes $\|b - Ax\|_2$ if and only if the residual vector $r = b - Ax$ is orthogonal to $\mathcal{R}(A)$, i.e. $A^T(b - Ax) = 0$, or equivalently x satisfies the **normal equations**.

$$A^T A x = A^T b \quad (8.1.2)$$

Proof. Let x be a vector for which $A^T(b - Ax) = 0$. Then for any $y \in \mathbf{R}^n$ $b - Ay = (b - Ax) + A(x - y)$. Squaring this and using (8.1.2) we obtain

$$\|b - Ay\|_2^2 = \|b - Ax\|_2^2 + \|A(x - y)\|_2^2 \geq \|b - Ax\|_2^2.$$

On the other hand assume that $A^T(b - Ax) = z \neq 0$. Then if $x - y = -\epsilon z$ we have for sufficiently small $\epsilon \neq 0$,

$$\|b - Ay\|_2^2 = \|b - Ax\|_2^2 - 2\epsilon\|z\|_2^2 + \epsilon^2\|Az\|_2^2 < \|b - Ax\|_2^2$$

so x does not minimize $\|b - Ax\|_2$. \square

The matrix $A^T A \in \mathbf{R}^{n \times n}$ is symmetric and positive semidefinite since

$$x^T A^T A x = \|Ax\|_2^2 \geq 0.$$

The normal equations $A^T A x = A^T b$ are always *consistent* since $A^T b \in \mathcal{R}(A^T) = \mathcal{R}(A^T A)$ and therefore a least squares solution always exists. Any solution to the normal equations is a least squares solution.

By Theorem 8.1.1 any least squares solution x will decompose the right hand side b into two orthogonal components

$$b = Ax + r, \quad r \perp Ax. \quad (8.1.3)$$

Here $Ax = b - r = P_{\mathcal{R}(A)} b$ is the orthogonal projection of b (see Section 8.1.3) onto $\mathcal{R}(A)$ and $r \in \mathcal{N}(A^T)$ (cf. Figure 8.1.1). Note that although the least squares solution x may not be unique the decomposition in (8.1.3) always is unique.

We now introduce a related problem. Suppose that the vector $y \in \mathbf{R}^m$ is required to satisfy exactly $n < m$ linearly independent equations $A^T y = c$. We want to find the **minimum norm solution**, i.e. to solve the problem

$$\min \|y\|_2 \quad \text{subject to } A^T y = c. \quad (8.1.4)$$

Let y be any solution of $A^T y = c$, and write $y = y_1 + y_2$, where $y_1 \in \mathcal{R}(A)$, $y_2 \in \mathcal{N}(A^T)$. Then $A^T y_2 = 0$ and hence y_1 is also a solution. Since $y_1 \perp y_2$ we have

$$\|y_1\|_2^2 = \|y\|_2^2 - \|y_2\|_2^2 \leq \|y\|_2^2,$$

with equality only if $y_2 = 0$. This shows that the minimum norm solution lies in $\mathcal{R}(A)$, i.e., $y = Az$ for some $z \in \mathbf{R}^n$. Substituting this in (8.1.4) gives the normal equations $A^T A z = c$. Since A has full column rank the matrix $A^T A$ is nonsingular and the solution is given by

$$y = A(A^T A)^{-1} c \quad (8.1.5)$$

A slightly more general problem is the **conditional least squares** problem

$$\min_y \|y - b\|_2 \quad \text{subject to } A^T y = c. \quad (8.1.6)$$

By a similar argument as used above the solution satisfies $y - b \in \mathcal{R}(A)$. Setting $y = b - Az$, and substituting in $A^T y = c$ we find that z satisfies the equation

$$A^T A z = A^T b - c. \quad (8.1.7)$$

Hence, the unique solution to problem (8.1.6) is

$$y = (I - A(A^T A)^{-1} A^T) b + A(A^T A)^{-1} c. \quad (8.1.8)$$

where $P_{\mathcal{N}(A^T)} = I - A(A^T A)^{-1} A^T$ is the orthogonal projection onto $\mathcal{N}(A^T)$.

Example 8.1.2.

The height $h_k = h(t_k)$ of a falling body is measured at times $t_k = t_0 + k\Delta t$, $k = 1 : m$. The adjusted values $\hat{h}_k = h_k - y_k$ should lie on a parabola, that is, the third differences must vanish. This leads to the problem minimizing

$$\min_y \|y - h\|_2 \quad \text{subject to} \quad A^T y = 0$$

where ($m = 7$)

$$A^T = \begin{pmatrix} 1 & -3 & 3 & -1 & 0 & 0 & 0 \\ 0 & 1 & -3 & 3 & -1 & 0 & 0 \\ 0 & 0 & 1 & -3 & 3 & -1 & 0 \\ 0 & 0 & 0 & 1 & -3 & 3 & -1 \end{pmatrix}.$$

which is a conditional least squares problem.

The solution to the standard linear least squares problem $\min_x \|Ax - b\|_2$ is characterized by the two conditions $A^T r = 0$ and $r = b - Ax$. These are $n + m$ equations

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}. \quad (8.1.9)$$

for the unknowns x and the residual r . This special case of is often called the **augmented system** for the least squares problem. The following theorem gives a unified formulation of the least squares and conditional least squares problems in terms of an augmented system.

Theorem 8.1.2.

Let the matrix if $A \in \mathbf{R}^{m \times n}$ have full column rank and consider the symmetric linear system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}, \quad (8.1.10)$$

Then the system is nonsingular and gives the first order conditions for the following two optimization problems:

1. Linear least squares problems

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 + c^T x. \quad (8.1.11)$$

2. Conditional least squares problem

$$\min_r \frac{1}{2} \|y - b\|_2^2, \quad \text{subject to} \quad A^T y = c, \quad (8.1.12)$$

Proof. The system (8.1.10) can be obtained by differentiating (8.1.11) to give

$$A^T(Ax - b) + c = 0,$$

and setting $y = r = b - Ax$.

The system can also be obtained by differentiating the Lagrangian

$$L(x, y) = \frac{1}{2}y^T y - y^T b + x^T (A^T y - c)$$

of (8.1.12), and equating to zero. Here x is the vector of Lagrange multipliers . \square

The augmented system plays a key role in the perturbation analysis of least squares problems (Section 8.2.3) as well as in the iterative refinement of least squares solutions (Section 8.3.7)

8.1.2 The Gauss–Markov Model

Gauss claims he discovered the method of least squares in 1795. He used it for analyzing surveying data and for astronomical calculation. A famous example is when Gauss successfully predicted the orbit of the asteroid Ceres in 1801.

Gauss [232] in 1821 put the method of least squares on a sound theoretical basis. To describe his results we first need to introduce some concepts from statistics. Let the probability that random variable $y \leq x$ be equal to $F(x)$, where $F(x)$ is nondecreasing, right continuous, and satisfies

$$0 \leq F(x) \leq 1, \quad F(-\infty) = 0, \quad F(\infty) = 1.$$

Then $F(x)$ is called the **distribution function** for y .

The **expected value** and the **variance** of y are defined as the Stieltjes integrals

$$\mathcal{E}(y) = \mu = \int_{-\infty}^{\infty} y dF(y), \quad \mathcal{E}(y - \mu)^2 = \sigma^2 = \int_{-\infty}^{\infty} (y - \mu)^2 dF(y),$$

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

$$\begin{aligned} \sigma_{ij} &= \mathcal{E}[(y_i - \mu_i)(y_j - \mu_j)] = \int_{-\infty}^{\infty} (y_i - \mu_i)(y_j - \mu_j) dF(y_i, y_j) \\ &= \mathcal{E}(y_i y_j) - \mu_i \mu_j. \end{aligned}$$

The covariance matrix $V \in \mathbf{R}^{n \times n}$ of y is defined by

$$V = \mathcal{V}(y) = \mathcal{E}[(y - \mu)(y - \mu)^T] = \mathcal{E}(yy^T) - \mu\mu^T.$$

where the diagonal element σ_{ii} is the variance of y_i .

Definition 8.1.3.

Let $A \in \mathbf{R}^{m \times n}$ be a known matrix, $b \in \mathbf{R}^m$ a vector of observations, and $x \in \mathbf{R}^n$ an unknown parameter vector. The **Gauss–Markov model** is a linear statistical model, where it is assumed that a linear relationship

$$Ax = b + e, \quad \mathcal{E}(e) = 0, \quad \mathcal{V}(e) = \sigma^2 W, \quad (8.1.13)$$

holds.²⁸ Here e is a vector of random errors, $W \in \mathbf{R}^{m \times n}$ a symmetric nonnegative definite matrix and σ^2 , an unknown constant. In the **standard case** the errors are assumed to be independently and identically distributed, i.e., $W = I_m$.

We now prove some properties which will be useful in the following.

Lemma 8.1.4.

Let $B \in \mathbf{R}^{r \times n}$ be a matrix and y a random vector with $\mathcal{E}(y) = \mu$ and covariance matrix V . Then the expected value and covariance matrix of By is

$$\mathcal{E}(By) = B\mu, \quad \mathcal{V}(By) = BVB^T. \quad (8.1.14)$$

In the special case that $B = b^T$ is a row vector $\mathcal{V}(b^T y) = \mu \|b\|_2^2$.

Proof. The first property follows directly from the definition of expected value. The second follows from the relation

$$\begin{aligned} \mathcal{V}(By) &= \mathcal{E}[(B(y - \mu)(y - \mu)^T B^T)] \\ &= B\mathcal{E}[(y - \mu)(y - \mu)^T]B^T = BVB^T. \end{aligned}$$

□

We make the following definitions:

Definition 8.1.5.

Let $g = c^T y$, where c is a constant vector, be a linear function of the random vector y . Then $c^T y$ is an **unbiased estimate** of the parameter θ if $\mathcal{E}(c^T y) = \theta$. When such a function exists, θ is called an **estimable parameter**. Further, it is a minimum variance (best) linear unbiased estimate of θ if $\mathcal{V}(g)$ is minimized over all such linear estimators. .

Theorem 8.1.6 (The Gauss–Markov Theorem).

Consider the linear model (8.1.13) with covariance matrix $\sigma^2 I$. Let \hat{x} be the least square estimator, obtained by minimizing over x the sum of squares $\|Ax - b\|_2^2$. Then the best linear unbiased estimator of any linear functional $\theta = c^T x$ is $c^T \hat{x}$. The covariance matrix of the estimate \hat{x} equals

$$\mathcal{V}(\hat{x}) = V = \sigma^2 (A^T A)^{-1} \quad (8.1.15)$$

²⁸In statistical literature the Gauss–Markov model is written $X\beta = y + e$. We choose another notation in order to be consistent throughout the book.

Furthermore, the residual vector $r = b - Ax$ is uncorrelated with \hat{x} and the quadratic form

$$s^2 = \frac{1}{m-n} \hat{r}^T \hat{r} \quad (8.1.16)$$

is an unbiased estimate of σ^2 , that is, $\mathcal{E}(s^2) = \sigma^2$.

Proof. If we set $\hat{b} = b + e$, then $\mathcal{E}(\hat{b}) = b = Ax$ and $\mathcal{V}(\hat{b}) = \sigma^2 I$. Consider the estimate $\hat{\theta} = d^T \hat{b}$ of the linear functional $\theta = c^T x$. Since θ is unbiased, we have

$$\mathcal{E}(\hat{\theta}) = d^T \mathcal{E}(\hat{b}) = d^T Ax = c^T x,$$

which shows that $A^T d = c$. From Lemma 8.1.4 it follows that $\mathcal{V}(\hat{\theta}) = \sigma^2 d^T d$. Thus, we wish to minimize $d^T d$ subject to $A^T d = c$. Using the method of Lagrange multipliers, let

$$Q = d^T d - 2\lambda^T (A^T d - c),$$

where λ is a vector of Lagrange multipliers. A necessary condition for Q to be a minimum is that

$$\frac{\partial Q}{\partial d} = 2(d^T - \lambda^T A^T) = 0,$$

or $d = A\lambda$. Premultiplying this by A^T results in $A^T A \lambda = A^T d = c$ and since $A^T A$ is nonsingular this gives $d = A\lambda = A(A^T A)^{-1}c$. This gives

$$\hat{\theta} = d^T \hat{b} = c^T (A^T A)^{-1} A^T \hat{b} = c^T \hat{x}$$

where \hat{x} is the solution to the normal equations. But the solution of the normal equations minimizes the sum of squares $(b - Ax)^T (b - Ax)$ with respect to x . \square

Remark 8.1.1. In the literature Gauss–Markov theorem is sometimes stated in less general forms. It is important to note that in the theorem errors are *not* assumed to be normally distributed, nor are they assumed to be independent (only uncorrelated—a weaker condition). They are also *not* assumed to be identically distributed, but only having zero mean and the same variance.

Remark 8.1.2. It is fairly straight-forward 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 in complex stochastic processes; see Miller [438].

If $\text{rank}(A) < n$, then the normal equations are singular but consistent. In this case a linear functional $c^T x$ is estimable if and only if

$$c \in \mathcal{R}(A^T).$$

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 are uncorrelated with \hat{x} , i.e.,

$$\mathcal{V}(\hat{r}, \hat{x}) = 0, \quad \mathcal{V}(s^2, \hat{x}) = 0.$$

An estimate of the covariance of the linear functional $c^T x$ is given by $s^2(c^T(A^TA)^{-1}c)$. In particular, for the components $x_i = e_i^T x$,

$$s^2(e_i^T(A^TA)^{-1}e_i) = s^2(A^TA)_{ii}^{-1}.$$

the i th diagonal element of $(A^TA)^{-1}$.

It is often the case that the errors have a positive definite covariance matrix different from $\sigma^2 I$. The above results are easily modified to cover this case.

Theorem 8.1.7.

Consider a linear model with the error covariance matrix equal to the symmetric positive definite matrix $\mathcal{V}(e) = \sigma^2 V$. If A has full column rank, then the best unbiased linear estimate is given by the solution to

$$\min_x (Ax - b)^T V^{-1} (Ax - b). \quad (8.1.17)$$

The covariance matrix of the estimate \hat{x} is

$$\mathcal{V}(\hat{x}) = \sigma^2 (A^T V^{-1} A)^{-1} \quad (8.1.18)$$

and

$$s^2 = \frac{1}{m-n} (b - A\hat{x})^T V^{-1} (b - A\hat{x}), \quad (8.1.19)$$

is an unbiased estimate of σ

A situation that often occurs is that the error covariance is a diagonal matrix

$$V = \text{diag}(v_{11}, v_{22}, \dots, v_{mm})$$

Then (8.1.17) is called a **weighted least squares** problem. It can easily be transformed to the standard case by scaling the i th equation by $1/\sqrt{v_{ii}}$. Note that the smaller the variance the larger weight should be given to a particular equation. It is important to note that different scalings will give different solutions, unless the system is $Ax = b$ is consistent.

In the general Gauss-Markov model no assumption is made on the dimension or rank of A or the rank of the covariance matrix except that A and W have the same number of rows. Assume that $\text{rank}(W) = k \leq n$ and given in factored form

$$W = BB^T, \quad B \in \mathbf{R}^{m \times k} \quad (8.1.20)$$

If W is initially given, the B can be computed as the Cholesky factor of W . Then the Guass-Markov model can be replaced by the equivalent model

$$Ax = b + Bu, \quad \mathcal{V}(u) = \sigma^2 I. \quad (8.1.21)$$

The best linear estimate of x is a solution to the **constrained** linear least squares problem.

$$\min_{x,u} u^T u \quad \text{subject to} \quad b = Ax + Bu \quad (8.1.22)$$

Here we must require that the consistency condition

$$b \in \mathcal{R}(A, B)$$

is satisfied. If this does not hold, then b could not have come from the linear model (8.1.21). The solution \hat{x} to (8.1.22) may not be unique. In this case we should take \hat{x} to be the solution of minimum norm. For a full analysis of the general models we refer to Korouklis and Paige [385]. Solution methods for **constrained** linear least squares problem are treated in Section 8.6.3.

8.1.3 Orthogonal and Oblique Projections

We have seen that the least squares solution is characterized by the property that its residual is orthogonal to its projection onto $\mathcal{R}(A)$. In this section make a systematic study of both orthogonal and more general projection matrices.

Any matrix $P \in \mathbf{C}^{n \times n}$ such that $P^2 = P$ is called **idempotent** and a **projector**. An arbitrary vector $v \in \mathbf{C}^n$ is decomposed in a unique way as

$$v = Pv + (I - P)v = v_1 + v_2. \quad (8.1.23)$$

Here $v_1 = Pv$ is a projection of v onto $\mathcal{R}(P)$, the column space of P . Since $Pv_2 = (P - P^2)v = 0$ it follows that $(I - P)$ is a projection onto $\mathcal{N}(P)$, the null space of P .

If λ is an eigenvalue of a projector P , then from $P^2 = P$ it follows that $\lambda^2 = \lambda$. Hence, the eigenvalues of P are either 1 or 0 and $k = \text{trace}(P)$ is the rank of P .

If P is Hermitian, $P^H = P$, then

$$v_1^H v_2 = (Pv)^H (I - P)v = v^H P(I - P)v = v^H (P - P^2)v = 0.$$

In this case v_2 lies in the orthogonal complement of $\mathcal{R}(P)$; and P is an **orthogonal projector**.

It can be shown that the orthogonal projector P onto a given subspace \mathcal{S} is unique, see Problem 8.1.1. The following property follows immediately from the Pythagorean theorem.

Lemma 8.1.8.

Let the orthogonal projection of vector $x \in \mathbf{C}^n$ onto a subspace $\mathcal{S} \subset \mathbf{C}^n$ be $z = Px \in \mathcal{S}$. Then z is the point in \mathcal{S} closest to x .

Let P be an orthogonal projector onto \mathcal{S} and U_1 a unitary basis for \mathcal{S} . Then we can always find a unitary basis U_2 for the orthogonal complement of \mathcal{S} . Then $U = (U_1 \ U_2)$ is unitary and $U^H U = U U^H = I$, and P can be expressed in the form

$$P = U_1 U_1^H, \quad I - P = U_2 U_2^H; \quad (8.1.24)$$

For an orthogonal projector we have

$$\|Pv\|_2 = \|U_1^H v\|_2 \leq \|v\|_2 \quad \forall v \in \mathbf{C}^m, \quad (8.1.25)$$

where equality holds for all vectors in $\mathcal{R}(U_1)$ and thus $\|P\|_2 = 1$. The converse is also true; P is an orthogonal projection only if (8.1.25) holds.

A projector P such that $P \neq P^H$ is called an **oblique projector**. We can write the spectral decomposition

$$P = (X_1 \ X_2) \begin{pmatrix} I_k & 0 \\ 0 & 0_{n-k} \end{pmatrix} \begin{pmatrix} \hat{Y}_1^H \\ \hat{Y}_2^H \end{pmatrix} = X_1 \hat{Y}_1^H. \quad (8.1.26)$$

where

$$\begin{pmatrix} \hat{Y}_1^H \\ \hat{Y}_2^H \end{pmatrix} (X_1 \ X_2) = \begin{pmatrix} \hat{Y}_1^H X_1 & \hat{Y}_1^H X_2 \\ \hat{Y}_2^H X_1 & \hat{Y}_2^H X_2 \end{pmatrix} = \begin{pmatrix} I_k & 0 \\ 0 & I_{n-k} \end{pmatrix}. \quad (8.1.27)$$

In particular, $\hat{Y}_1^H X_2 = 0$ and $\hat{Y}_2^H X_1 = 0$. Hence, the columns of X_2 form a basis for the orthogonal complement of $\mathcal{R}(\hat{Y}_1)$ and, similarly, the columns of \hat{Y}_2 form a basis for the orthogonal complement of $\mathcal{R}(X_1)$.

In terms of this spectral decomposition $I - P = X_2 \hat{Y}_2^H$ and the splitting (8.1.23) can be written

$$v = X_1(\hat{Y}_1^H v) + X_2(\hat{Y}_2^H v) = v_1 + v_2. \quad (8.1.28)$$

Here v_1 is the oblique projection of v onto $\mathcal{R}(P)$ along $\mathcal{N}(P)$.

Let Y_1 be an orthogonal matrix whose columns span $\mathcal{R}(\hat{Y}_1)$. Then there is a nonsingular matrix G_1 such that $\hat{Y}_1 = Y_1 G_1$. From (8.1.27) it follows that $G_1^H Y_1^H X_1 = I_k$, and hence $G_1^H = (Y_1^H X_1)^{-1}$. Similarly, $Y_2 = (Y_2^H X_2)^{-1} Y_2$ is an orthogonal matrix whose columns span $\mathcal{R}(\hat{Y}_2)$. Hence, we can write

$$P = X_1(Y_1^H X_1)^{-1} Y_1^H, \quad I - P = X_2(Y_2^H X_2)^{-1} Y_2^H. \quad (8.1.29)$$

This shows that $\|P\|$ can be large when the matrix $Y_1^H X_1$ is ill-conditioned.

Example 8.1.3.

We illustrate the case when $n = 2$ and $n_1 = 1$. Let the vectors x_1 and y_1 be normalized so that $\|x_1\|_2 = \|y_1\|_2 = 1$ and let $y_1^H x_1 = \cos \theta$, where θ is the angle between x_1 and y_1 . Since

$$P = x_1(y_1^H x_1)^{-1} y_1^H = \frac{1}{\cos \theta} x_1 y_1^H.$$

Hence, $\|P\|_2 = 1/\cos \theta \geq 1$, and $\|P\|_2$ becomes very large when y_1 is almost orthogonal to x_1 . When $y_1 = x_1$ we have $\theta = 0$ and P is an orthogonal projection.

8.1.4 Generalized Inverses and the SVD

The SVD introduced in Section 7.1.5 is a powerful tool both for analyzing and solving linear least squares problems. The reason for this is that the orthogonal matrices that transform A to diagonal form do not change the l_2 -norm. We have the following fundamental result.

Theorem 8.1.9.

Let $A \in \mathbf{R}^{m \times n}$ have the singular value decomposition

$$A = (U_1 \ U_2) \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}, \quad (8.1.30)$$

where U_1 and V_1 have $r = \text{rank}(A)$ columns. The least squares problem

$$\min_{x \in S} \|x\|_2, \quad S = \{x \in \mathbf{R}^n \mid \|b - Ax\|_2 = \min\}. \quad (8.1.31)$$

always has a unique solution, which can be written as

$$x = V_1 \Sigma_1^{-1} U_1^T b. \quad (8.1.32)$$

Proof. Using the orthogonal invariance of the l_2 norm we have

$$\begin{aligned} \|b - Ax\|_2 &= \|U^T(b - AVV^T x)\|_2 \\ &= \left\| \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} - \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} c_1 - \Sigma_1 z_1 \\ c_2 \end{pmatrix} \right\|_2. \end{aligned}$$

where $z_1, c_1 \in \mathbf{R}^r$ and

$$c = U^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}.$$

The residual norm will attain its minimum value equal to $\|c_2\|_2$ for $z_1 = \Sigma_1^{-1} c_1$, z_2 arbitrary. Obviously the choice $z_2 = 0$ minimizes $\|x\|_2 = \|Vz\|_2 = \|z\|_2$. \square

Note that problem (8.1.31) includes as special cases the solution of both overdetermined and underdetermined linear systems. We set

$$A^\dagger = (V_1 \ V_2) \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} = V_1 \Sigma_1^{-1} U_1^T \in \mathbf{R}^{n \times m} \quad (8.1.33)$$

We call A^\dagger the **pseudo-inverse** of A and $x = A^\dagger b$ the pseudo-inverse solution of $Ax = b$. The pseudo-inverse solution (8.1.33) can also be written

$$x = \sum_{i=1}^r \frac{u_i^T b}{\sigma_i} \cdot v_i. \quad (8.1.34)$$

The following two cases are important special cases:

- In an overdetermined case where A has full column rank ($r = n$) the submatrix V_2 is empty and the pseudo-inverse becomes

$$A^\dagger = V \Sigma_1^{-1} U_1^T. \quad (8.1.35)$$

- In an underdetermined case where A has full row rank ($r = m$) the submatrix U_2 is empty and the pseudo-inverse becomes

$$A^\dagger = V_1 \Sigma_1^{-1} U^T. \quad (8.1.36)$$

Note that for computing the pseudo-inverse solution we only need to compute the “thin” SVD, i.e. the nonzero singular values, the matrix V_1 and the vector $U_1^T b$. Methods for computing the SVD are described in Section 10.5.3 and Section 10.6.4.

The matrix A^\dagger is often called the **Moore–Penrose inverse**. Moore [443] developed the concept of the general reciprocal, which was rediscovered by Bjerhammar [55]. Penrose [1955], gave an elegant algebraic characterization and showed that $X = A^\dagger$ is uniquely determined by the four **Penrose conditions** :

$$(1) \quad AXA = A, \quad (2) \quad XAX = X, \quad (8.1.37)$$

$$(3) \quad (AX)^T = AX, \quad (4) \quad (XA)^T = XA. \quad (8.1.38)$$

It can be directly verified that $X = A^\dagger$ given by (8.1.33) 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. (See also Problem 8.1.2.) The following properties of the pseudoinverse easily follow from (8.1.36).

Theorem 8.1.10.

1. $(A^\dagger)^\dagger = A;$
2. $(A^\dagger)^H = (A^H)^\dagger;$
3. $(\alpha A)^\dagger = \alpha^\dagger A^\dagger;$
4. $(A^H A)^\dagger = A^\dagger (A^\dagger)^H;$
5. if U and V are unitary $(UAV^H)^\dagger = VA^\dagger U^H;$
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 $\text{trace}(A^\dagger A)$.

The orthogonal projections onto the four fundamental subspaces of A have the following simple expressions in terms of the pseudo-inverse :

$$\begin{aligned} P_{\mathcal{R}(A)} &= AA^\dagger = U_1 U_1^T, & P_{\mathcal{N}(A^T)} &= I - AA^\dagger = U_2 U_2^T, & (8.1.39) \\ P_{\mathcal{R}(A^T)} &= A^\dagger A = V_1 V_1^T, & P_{\mathcal{N}(A)} &= I - A^\dagger A = V_2 V_2^T. \end{aligned}$$

These expressions are easily verified using the definition of an orthogonal projection and the Penrose conditions.

Another useful characterization of the pseudo-inverse solution is the following:

Theorem 8.1.11.

The pseudo-inverse solution $x = A^\dagger b$ is uniquely characterized by the two geometrical conditions

$$x \in \mathcal{R}(A^T), \quad r = b - Ax \in \mathcal{N}(A^T). \quad (8.1.40)$$

Proof. These conditions are easily verified from (8.1.34). \square

In the special case that $A \in \mathbf{R}^{m \times n}$ and $\text{rank}(A) = n$ it holds that

$$A^\dagger = (A^T A)^{-1} A^T, \quad (A^T)^\dagger = A (A^T A)^{-1} \quad (8.1.41)$$

These expressions follow from the normal equations (8.2.2) and (8.1.5). Some properties of the usual inverse can be extended to the pseudo-inverse, e.g., the relations

$$(A^\dagger)^\dagger = A, \quad (A^T)^\dagger = (A^\dagger)^T,$$

easily follow from (8.1.33). In general $(AB)^\dagger \neq B^\dagger A^\dagger$. The following theorem gives a useful *sufficient* conditions for the relation $(AB)^\dagger = B^\dagger A^\dagger$ to hold.

Theorem 8.1.12.

If $A \in \mathbf{R}^{m \times r}$, $B \in \mathbf{R}^{r \times n}$, and $\text{rank}(A) = \text{rank}(B) = r$, then

$$(AB)^\dagger = B^\dagger A^\dagger = B^T (BB^T)^{-1} (A^T A)^{-1} A^T. \quad (8.1.42)$$

Proof. The last equality follows from (8.1.41). The first equality is verified by showing that the four Penrose conditions are satisfied. \square

Any matrix A^- satisfying the first Penrose condition

$$AA^-A = A \quad (8.1.43)$$

is called a **generalized inverse** of A . It is also called an **inner inverse** or $\{1\}$ -inverse. If it satisfies the second condition $AA^-A = A^-AA^- = A^-$ 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

$$x = A^-b + (I - A^-A)y, \quad y \in \mathbf{C}^n.$$

We have also

$$(AA^-A^-)^2 = AA^-AA^- = AA^-, \quad (A^-A)^2 = A^-AA^-A = A^-A.$$

This shows that AA^- and A^-A are idempotent and therefore (in general oblique) projectors

$$AX = P_{\mathcal{R}(A),S}, \quad XA = P_{T,\mathcal{N}(A)},$$

where S and T are some subspaces complementary to $\mathcal{R}(A)$ and $\mathcal{N}(A)$, respectively.

Let $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$. Then $\|Ax - b\|_2$ is the minimized when x satisfies the normal equations $A^T Ax = A^T b$. Suppose now that a generalized inverse A^- satisfies

$$(AA^-)^T = AA^-. \quad (8.1.44)$$

Then AA^- is the orthogonal projector onto $\mathcal{R}(A)$ and A^- is called a **least squares inverse**. We have

$$A^T = (AA^-A)^T = A^TAA^-,$$

which shows that $x = A^-b$ satisfies the normal equations and therefore is a least squares solution. Conversely, if $A^- \in \mathbf{R}^{n \times m}$ has the property that for all b , $\|Ax-b\|_2$ is smallest when $x = A^-b$, then A^- is a least squares inverse.

The following dual result holds also: If A^- is a generalized inverse, and

$$(A^-A)^T = A^-A$$

then A^-A is the orthogonal projector orthogonal to $\mathcal{N}(A)$ 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

$$x = A^Tz, \quad AA^Tz = b.$$

For a minimum norm inverse we have

$$A^T = (AA^-A)^T = A^-AA^T,$$

and hence $x = A^Tz = A^-(AA^T)z = A^-b$, which shows that $x = A^-b$ is the solution of smallest norm. Conversely, if $A^- \in \mathbf{R}^{n \times m}$ is such that, whenever $Ax = b$ has a solution, then $x = A^-b$ is a minimum norm solution, then A^- is a minimum norm inverse.

We now derive some perturbation bounds for the pseudo-inverse of a matrix $A \in \mathbf{R}^{m \times n}$. 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 obtained by taking

$$A = \begin{pmatrix} \sigma & 0 \\ 0 & 0 \end{pmatrix}, \quad E = \begin{pmatrix} 0 & 0 \\ 0 & \epsilon \end{pmatrix},$$

where $\sigma > 0$, $\epsilon \neq 0$. Then $1 = \text{rank}(A) \neq \text{rank}(A+E) = 2$,

$$A^\dagger = \begin{pmatrix} \sigma^{-1} & 0 \\ 0 & 0 \end{pmatrix}, \quad (A+E)^\dagger = \begin{pmatrix} \sigma^{-1} & 0 \\ 0 & \epsilon^{-1} \end{pmatrix},$$

and $\|(A+E)^\dagger - A^\dagger\|_2 = |\epsilon|^{-1} = 1/\|E\|_2$. This example shows that formulas derived by operating formally with pseudo-inverses may have no meaning numerically.

The perturbations for which the pseudo-inverse is well behaved can be characterized by the condition

$$\text{rank}(A) = \text{rank}(B) = \text{rank}(P_{\mathcal{R}(A)}BP_{\mathcal{R}(A^T)}); \tag{8.1.45}$$

The matrix B is said to be an **acute perturbation** of A if this condition holds; see Stewart [545, 1977]. In particular, we have the following result.

Theorem 8.1.13.

If $\text{rank}(A + E) = \text{rank}(A) = r$, and $\eta = \|A^\dagger\|_2 \|E\|_2 < 1$, then

$$\|(A + E)^\dagger\|_2 \leq \frac{1}{1 - \eta} \|A^\dagger\|_2. \quad (8.1.46)$$

Proof. From the assumption and Theorem 1.2.7 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 (8.1.46). \square

Let $A, B \in \mathbf{R}^{m \times n}$, 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 pseudo-inverse identity** (see [605]) holds

$$B^\dagger - A^\dagger = -B^\dagger EA^\dagger + (B^T B)^\dagger E^T P_{N(A^T)} + P_{N(B)} E^T (AA^T)^\dagger, \quad (8.1.47)$$

This identity can be proved by expressing the projections in terms of pseudo-inverses using the relations in (8.1.39).

Let $A = A(\alpha)$ be a matrix, where α is a scalar parameter. Under the assumption that $A(\alpha)$ has local constant rank the following formula for the derivative of the pseudo-inverse $A^\dagger(\alpha)$ follows from (8.1.47):

$$\frac{dA^\dagger}{d\alpha} = -A^\dagger \frac{dA}{d\alpha} A^\dagger + (A^T A)^\dagger \frac{dA^T}{d\alpha} P_{N(A)} + P_{N(A^T)} \frac{dA^T}{d\alpha} (AA^T)^\dagger. \quad (8.1.48)$$

This formula is due to Wedin [605, p 21]. We observe that if A has full column rank, then the second term vanishes; if A has full row rank, then it is the third term that vanishes. The variable projection algorithm for separable nonlinear least squares is based on a related formula for the derivative of the orthogonal projection matrix $P_{\mathcal{R}(A)}$; see Section 11.2.5.

For the case when $\text{rank}(B) = \text{rank}(A)$ the following theorem applies.

Theorem 8.1.14. If $B = A + E$ and $\text{rank}(B) = \text{rank}(A)$, then

$$\|B^\dagger - A^\dagger\| \leq \mu \|B^\dagger\| \|A^\dagger\| \|E\| \quad (8.1.49)$$

where $\mu = 1$ for the Frobenius norm $\|\cdot\|_F$, and for the spectral norm $\|\cdot\|_2$,

$$\mu = \begin{cases} \frac{1}{2}(1 + \sqrt{5}) & \text{if } \text{rank}(A) < \min(m, n), \\ \sqrt{2} & \text{if } \text{rank}(A) = \min(m, n). \end{cases}$$

Proof. For the $\|\cdot\|_2$ norm, see Wedin [606]. The result that $\mu = 1$ for the Frobenius norm is due to van der Sluis and Veltkamp [583]. \square

8.1.5 Matrix Approximation and the SVD

The singular values decomposition (SVD) plays a very important role in a number of least squares matrix approximation problems. In this section we have collected a number of results that will be used extensively in the following.

In the proof of Theorem 7.1.16 we showed that the largest singular value of A could 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 8.1.15.

Let $A \in \mathbf{R}^{m \times n}$ have singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$, $p = \min(m, n)$, and S be a linear subspace of \mathbf{R}^n of dimension $\dim(S)$. Then

$$\sigma_i = \max_{\dim(S)=i} \max_{\substack{x \in S \\ x^H = 1}} \|Ax\|_2, \quad i = 1 : p, \quad (8.1.50)$$

and

$$\sigma_i = \min_{\dim(S)=p-i+1} \max_{\substack{x \in S \\ x^H = 1}} \|Ax\|_2, \quad i = 1 : p. \quad (8.1.51)$$

Proof. The result follows from the relationship shown in Theorem 10.5.2 and the corresponding result for the Hermitian eigenvalue problem in Theorem 10.2.8 (Fischer's theorem). \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 8.1.16.

Let $A, B \in \mathbf{R}^{m \times n}$ 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

$$\max_i |\sigma_i - \tau_i| \leq \|A - B\|_2, \quad (8.1.52)$$

$$\sum_{i=1}^p |\sigma_i - \tau_i|^2 \leq \|A - B\|_F^2. \quad (8.1.53)$$

Proof. See Stewart [543, pp. 321–322]. \square

By the inequality (8.1.53) 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 result 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 below.

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 8.1.17.

Let

$$\hat{A} = (A, u) \in \mathbf{R}^{m \times n}, \quad m \geq n, \quad u \in \mathbf{R}^m.$$

Then the ordered singular values σ_i of A interlace the ordered singular values $\hat{\sigma}_i$ of \hat{A} as follows

$$\hat{\sigma}_1 \geq \sigma_1 \geq \hat{\sigma}_2 \geq \sigma_2 \dots \geq \hat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \hat{\sigma}_n.$$

Similarly, if A is bordered by a row,

$$\hat{A} = \begin{pmatrix} A \\ v^T \end{pmatrix} \in \mathbf{R}^{m \times n}, \quad m > n, \quad v \in \mathbf{R}^n,$$

then

$$\hat{\sigma}_1 \geq \sigma_1 \geq \hat{\sigma}_2 \geq \sigma_2 \dots \geq \hat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \hat{\sigma}_n \geq \sigma_n.$$

Proof. The theorem is a consequence of the Cauchy interlacing theorem for Hermitian matrices to be proved in Chapter 9; see Theorem 10.2.11. This says that the eigenvalues of the leading principal minor of order $n - 1$ of a Hermitian matrix B interlace those of B . Since

$$\begin{aligned} \begin{pmatrix} A^T \\ u^T \end{pmatrix} (A \quad u) &= \begin{pmatrix} A^T A & A^T u \\ u^T A & u^T u \end{pmatrix}, \\ \begin{pmatrix} A \\ v^T \end{pmatrix} (A^T \quad v) &= \begin{pmatrix} A A^T & A v \\ v^T A^T & v^T v \end{pmatrix} \end{aligned}$$

The result now follows from the observation that the singular values of A are the positive square roots of the eigenvalues of $A^T A$ and $A A^T$. \square

The best approximation of a matrix A by another matrix of lower rank can be expressed in terms of the SVD of A .

Theorem 8.1.18.

Let $\mathcal{M}_k^{m \times n}$ denote the set of matrices in $\mathbf{R}^{m \times n}$ of rank k . Assume that $A \in \mathcal{M}_r^{m \times n}$ and consider the problem

$$\min_{X \in \mathcal{M}_k^{m \times n}} \|A - X\|, \quad k < r.$$

Then the SVD expansion of A truncated to k terms $X = B = \sum_{i=1}^k \sigma_i u_i v_i^T$, solves this problem both for the l_2 norm and the Frobenius norm. Further, the minimum distance is given by

$$\|A - B\|_2 = \sigma_{k+1}, \quad \|A - B\|_F = (\sigma_{k+1}^2 + \dots + \sigma_r^2)^{1/2}.$$

The solution is unique for the Frobenius norm but not always for the l_2 norm.

Proof. Eckhard and Young [183] proved it for the Frobenius norm. Mirsky [439] generalized it to unitarily invariant norms, which includes the l_2 -norm. \square

Let $A \in \mathbf{C}^{m \times n}$, be a matrix of rank n with the “thin” SVD $A = U_1 \Sigma V^H$. Since $A = U_1 \Sigma V^H = U_1 \Sigma U_1^H U_1 V^H$, we have

$$A = PH, \quad P = U_1 V^H, \quad H = V \Sigma V^H, \quad (8.1.54)$$

where P is unitary, $P^H P = I$, and $H \in \mathbf{C}^{n \times n}$ is Hermitian positive semidefinite. The decomposition (8.1.54) is called the **polar decomposition** of A . If $\text{rank}(A) = n$, then H is positive definite and the polar decomposition is unique. If the polar decomposition $A = PH$ is given, then from a spectral decomposition $H = V \Sigma V^H$ one can construct the singular value decomposition $A = (PV)\Sigma V^H$. The polar decomposition is also related to the matrix square root and sign functions; see Section 10.8.4.

The significance of the factor P in the polar decomposition is that it is the unitary matrix closest to A .

Theorem 8.1.19.

Let $A \in \mathbf{C}^{m \times n}$ be a given matrix and $A = UH$ its polar decomposition. Then for any unitary matrix $U \in \mathcal{M}_{m \times n}$,

$$\|A - U\|_F \geq \|A - P\|_F.$$

Proof. This theorem was proved for $m = n$ and general unitarily invariant norms by Fan and Hoffman [197]. The generalization to $m > n$ follows from the additive property of the Frobenius norm. \square

Less well known is that the optimal properties of the Hermitian polar factor H . Let $A \in \mathbf{C}^{n \times n}$ 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 8.1.20.

Let $A \in \mathbf{C}^{m \times n}$ be Hermitian and $A = UH$ its polar decomposition. Set

$$B = A + E = \frac{1}{2}(H + A), \quad E = \frac{1}{2}(H - A).$$

Then for any positive semidefinite Hermitian matrix X it holds that

$$\|A - B\|_2 \leq \|A - X\|_2.$$

Proof. See Higham [323]. \square

8.1.6 Principal Angles and Distance Between Subspaces

In many applications the relationship between two given subspaces needs to be investigated. For example, in statistical models canonical correlations measure how “close” two set of observations are.

This and similar questions can be answered by computing angles between subspaces. Let \mathcal{F} and \mathcal{G} be subspaces of \mathbf{C}^n and assume that

$$p = \dim(\mathcal{F}) \geq \dim(\mathcal{G}) = q \geq 1.$$

The smallest angle $\theta_1 = \theta_1(\mathcal{F}, \mathcal{G}) \in [0, \pi/2]$ between \mathcal{F} and \mathcal{G} is defined by

$$\theta_1 = \max_{\substack{u \in \mathcal{F} \\ \|u\|_2=1}} \max_{\substack{v \in \mathcal{G} \\ \|v\|_2=1}} \theta(u, v).$$

where $\theta(u, v)$ is the acute angle between u and v . Assume that the maximum is attained for $u = u_1$ and $v = v_1$. Then θ_2 is defined as 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 . Continuing in this way until one of the subspaces is empty, we are led to the following definition:

Definition 8.1.21.

The principal angles $\theta_k \in [0, \pi/2]$ between two subspaces of \mathbf{C}^n are recursively defined for $k = 1 : q$, by

$$\theta_k = \max_{\substack{u \in \mathcal{F} \\ \|u\|_2=1}} \max_{\substack{v \in \mathcal{G} \\ \|v\|_2=1}} \theta(u, v). = \theta(u_k, v_k), \quad (8.1.55)$$

subject to the constraints

$$u^H u_j = 0, \quad v^H v_j = 0, \quad j = 1 : k - 1.$$

The vectors u_k and v_k , $k = 1 : q$, are called **principal vectors** of the pair of spaces.

The principal vectors are not always uniquely defined, but the principal angles are. 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} . It will be shown that it holds also that

$$u_j^H v_k = 0, \quad j \neq k, \quad j = 1 : p, \quad k = 1 : q.$$

Assume that the subspaces \mathcal{F} and \mathcal{G} are defined as the range of the unitary matrices $Q_F \in \mathbf{C}^{n \times p}$ and $Q_G \in \mathbf{C}^{n \times q}$. The following theorem shows the relationship between the principal angles and the SVD of the matrix $Q_F^H Q_G$.

Theorem 8.1.22.

Assume that the columns of $Q_F \in \mathbf{C}^{n \times p}$ and $Q_G \in \mathbf{C}^{n \times q}$, $p \geq q$, form unitary bases for two subspaces of \mathbf{C}^n . Let the thin SVD of the matrix $M = Q_F^H Q_G \in \mathbf{C}^{p \times q}$ be

$$M = Y C Z^H, \quad C = \text{diag}(\sigma_1, \dots, \sigma_q), \quad (8.1.56)$$

where $y^H Y = Z^H Z = ZZ^H = I_q$ 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

$$U = Q_F Y, \quad V = Q_G Z. \quad (8.1.57)$$

Proof. The singular values and vectors of M can be characterized by the property

$$\sigma_k = \max_{\|y\|_2 = \|z\|_2 = 1} y^H M z = y_k^H M z_k, \quad (8.1.58)$$

subject to $y^H y j = z^H z_j = 0$, $j = 1 : k$. If we put $u = Q_F y \in \mathcal{F}$ and $v = Q_G z \in \mathcal{G}$, then 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 $y^H M z = y^H Q_F^H Q_G z = u^H v$, (8.1.58) 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 (8.1.57) follows directly from definition 8.1.21. \square

The principal angles can be used to define the distance between two subspaces of the same dimension.

Definition 8.1.23.

The distance between two subspaces \mathcal{F} and \mathcal{G} of \mathbf{C}^n , both of dimension p , equals

$$\text{dist}(\mathcal{F}, \mathcal{G}) = \sin \theta_{\max}(\mathcal{F}, \mathcal{G})$$

where $\theta_{\max}(\mathcal{F}, \mathcal{G})$ is the largest principal angle between \mathcal{F} and \mathcal{G} . Equivalently

$$\theta_{\max}(\mathcal{F}, \mathcal{G}) = \max_{\substack{u \in \mathcal{F} \\ \|u\|_2=1}} \min_{\substack{v \in \mathcal{G} \\ \|v\|_2=1}} \theta(u, v). \quad (8.1.59)$$

where $\theta(u, v) = \arccos(u^H u)$ is the acute angle between u and v .

Clearly $0 \leq \text{dist}(\mathcal{F}, \mathcal{G}) \leq 1$, and $\text{dist}(\mathcal{F}, \mathcal{F}) = 0$ if and only if $\mathcal{F} = \mathcal{G}$. The distance can also be expressed using the orthogonal projectors $P_{\mathcal{F}}$ and $P_{\mathcal{G}}$ onto \mathcal{F} and \mathcal{G}

$$\text{dist}(\mathcal{F}, \mathcal{G}) = \|P_{\mathcal{F}} - P_{\mathcal{G}}\|_2; \quad (8.1.60)$$

see Golub and Van Loan [277, Theorem 2.6.1].

In principle, a unitary basis for the intersection of two subspaces is obtained by taking the vectors u_k that correspond to $\theta_k = 0$ or $\sigma_k = 1$. However, numerically small angles θ_k are well defined from $\sin \theta_k$ but not from $\cos \theta_k$. We now show how to compute $\sin \theta_k$.

We now change the notations slightly and write the SVD in (8.1.56) and the principal vectors as

$$M = Y_F C Y_G^H, \quad U_F = Q_F Y_F, \quad U_G = Q_G Y_G.$$

Since Q_F is unitary it follows that $P_F = Q_F Q_F^H$ is the orthogonal projector onto \mathcal{F} . Then we have

$$P_F Q_G = Q_F Q_F^H Q_G = Q_F M = U_F C Y_G. \quad (8.1.61)$$

Squaring $Q_G = P_F Q_G + (I - P_F) Q_G$, using (8.1.61) and $P_F(I - P_F) = 0$ gives

$$Q_G^H (I - P_F)^2 Q_G = Y_G (I - C^2) Y_G^H,$$

which shows that the SVD of $(I - P_F) Q_G = Q_G - Q_F M$ can be written

$$(I - P_F) Q_G = W_F S Y_G^H, \quad S^2 = I - C^2,$$

and thus $S = \pm \text{diag}(\sin \theta_k)$.

We assume for convenience in the following that $p + q \leq n$. Then the matrix $W_F \in \mathbf{R}^{n \times q}$ can be chosen so that $W_F^H U_F = 0$.

$$(I - P_G) Q_F = Q_F - Q_G M = W_G S Y_F^H. \quad (8.1.62)$$

Combining this with $P_F Q_G = U_F C Y_G^H$ we can write

$$U_G = Q_G Y_F = (U_F C + W_F S) = (U_F \ W_F) \begin{pmatrix} C \\ S \end{pmatrix}.$$

If we put

$$P_{A,B} = U_G U_F^H = (U_F \ W_F) \begin{pmatrix} C \\ S \end{pmatrix} U_F^H$$

then the transformation $y = P_{A,B} x$, rotates a vector $x \in R(A)$ into a vector $y \in R(B)$, and $\|y\|_2 = \|x\|_2$. By analogy we have also the decomposition

$$(I - P_F) Q_G = Q_G - Q_F M = W_F S Y_G^H. \quad (8.1.63)$$

8.1.7 The CS Decomposition

More information about the relationship between two subspaces can be obtained from the **CS decomposition**. This is a special case a decomposition of a partitioned orthogonal matrix related to the SVD.

Theorem 8.1.24 (Thin CS Decomposition).

Let $Q_1 \in \mathbf{R}^{(m \times n)}$ have orthonormal columns, that is $Q_1^T Q_1 = I$, and be partitioned as

$$Q_1 = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} \}_{m_2}^{m_1}, \quad (8.1.64)$$

where $m_1 \geq n$, and $m_2 \geq n$. Then there are orthogonal matrices $U_1 \in \mathbf{R}^{m_1 \times m_1}$, $U_2 \in \mathbf{R}^{m_2 \times m_2}$, and $V_1 \in \mathbf{R}^{n \times n}$ such that

$$\begin{pmatrix} U_1 & 0 \\ 0 & U_2 \end{pmatrix}^T \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} V_1 = \begin{pmatrix} C \\ 0 \\ S \\ 0 \end{pmatrix} \quad (8.1.65)$$

where

$$C = \text{diag}(c_1, \dots, c_n), \quad S = \text{diag}(s_1, \dots, s_n), \quad (8.1.66)$$

are square nonnegative diagonal matrices satisfying $C^2 + S^2 = I_n$. The diagonal elements in C and S are

$$c_i = \cos(\theta_i), \quad s_i = \sin(\theta_i), \quad i = 1 : n,$$

where without loss of generality, we may assume that

$$0 \leq \theta_1 \leq \theta_2 \leq \dots \leq \theta_n \leq \pi/2.$$

Proof. To construct U_1 , V_1 , and C , note that since U_1 and V_1 are orthogonal and C is a nonnegative diagonal matrix, $Q_{11} = U_1 C V_1^T$ is the SVD of Q_{11} . Hence, the elements c_i are the singular values of Q_{11} , and since $\|Q_{11}\|_2 \leq \|Q\|_2 = 1$, we have $c_i \in [0, 1]$.

If we put $\tilde{Q}_{21} = Q_{21}V_1$, then the matrix

$$\begin{pmatrix} C \\ 0 \\ \tilde{Q}_{21} \end{pmatrix} = \begin{pmatrix} U_1^T & 0 \\ 0 & I_{m_2} \end{pmatrix} \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} V_1$$

has orthonormal columns. Thus, $C^2 + \tilde{Q}_{21}^T \tilde{Q}_{21} = I_n$, which implies that $\tilde{Q}_{21}^T \tilde{Q}_{21} = I_n - C^2$ is diagonal and hence the matrix $\tilde{Q}_{21} = (\tilde{q}_1^{(2)}, \dots, \tilde{q}_n^{(2)})$ has orthogonal columns.

We assume that the singular values $c_i = \cos(\theta_i)$ of Q_{11} have been ordered according to (8.1.24) and that $c_r < c_{r+1} = 1$. Then the matrix $U_2 = (u_1^{(2)}, \dots, u_p^{(2)})$ is constructed as follows. Since $\|\tilde{q}_j^{(2)}\|_2^2 = 1 - c_j^2 \neq 0$, $j \leq r$ we take

$$u_j^{(2)} = \tilde{q}_j^{(2)} / \|\tilde{q}_j^{(2)}\|_2, \quad j = 1, \dots, r,$$

and fill the possibly remaining columns of U_2 with orthonormal vectors in the orthogonal complement of $\mathcal{R}(\tilde{Q}_{21})$. From the construction it follows that $U_2 \in$

$\mathbf{R}^{m_2 \times m_2}$ is orthogonal and that

$$U_2^T \tilde{Q}_{21} = U_2 Q_{21} V_1 = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}, \quad S = \text{diag}(s_1, \dots, s_q)$$

with $s_j = (1 - c_j^2)^{1/2} > 0$, if $j = 1 : r$, and $s_j = 0$, if $j = r + 1 : n$. \square

In the theorem above we assumed that $n \leq m/2$. The general case gives rise to four different forms corresponding to cases where Q_{11} and/or Q_{21} have too few rows to accommodate a full diagonal matrix of order n .

The proof of the CS decomposition is constructive. In particular, U_1 , V_1 , and C can be computed by a standard SVD algorithm. However, the above algorithm for computing U_2 is unstable when some singular values c_i are close to 1. and needs to be modified.

Using the same technique the following CS decomposition of a square partitioned orthogonal matrix can be shown.

Theorem 8.1.25 (Full CS Decomposition).

Let

$$Q = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \in \mathbf{R}^{m \times m}. \quad (8.1.67)$$

be an arbitrary partitioning of the orthogonal matrix Q . Then there are orthogonal matrices

$$\begin{pmatrix} U_1 & 0 \\ 0 & U_2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} V_1 & 0 \\ 0 & V_2 \end{pmatrix}$$

such that

$$U^T Q V = \left(\begin{array}{c|c} U_1^T Q_{11} V_1 & U_1^T Q_{12} V_2 \\ \hline U_2^T Q_{21} V_1 & U_2^T Q_{22} V_2 \end{array} \right) = \left(\begin{array}{ccc|ccc} I & 0 & 0 & 0 & 0 & 0 \\ 0 & C & 0 & 0 & S & 0 \\ 0 & 0 & 0 & 0 & 0 & I \\ \hline 0 & 0 & 0 & I & 0 & 0 \\ 0 & S & 0 & 0 & -C & 0 \\ 0 & 0 & I & 0 & 0 & 0 \end{array} \right) \quad (8.1.68)$$

where $C = \text{diag}(c, \dots, c_n)$ and $S = \text{diag}(s, \dots, s_n)$ are square diagonal matrices satisfying $C^2 + S^2 = I_n$, and $0 < c_i, s_i < 1$, $i = 1 : n$.

Proof. For a proof, see Paige and Saunders [468]. \square

The history of the CS decomposition and its many applications are surveyed in Paige and Wei [473].

Review Questions

1.1 State the Gauss–Markov theorem.

- 1.2** Assume that A has full column rank. Show that the matrix $P = A(A^T A)^{-1} A^T$ is symmetric and satisfies the condition $P^2 = P$.
- 1.3** (a) Give conditions for a matrix P to be the orthogonal projector onto a subspace $S \in \mathbf{R}^n$.
 (b) Define the orthogonal complement of S in \mathbf{R}^n .
- 1.4** (a) Which are the four fundamental subspaces of a matrix? Which relations hold between them? Express the orthogonal projections onto the fundamental subspaces in terms of the SVD.
 (b) Give two geometric conditions which are necessary and sufficient conditions for x to be the pseudo-inverse solution of $Ax = b$.
- 1.5** Which of the following relations are universally correct?
 (a) $\mathcal{N}(B) \subseteq \mathcal{N}(AB)$. (b) $\mathcal{N}(A) \subseteq \mathcal{N}(AB)$. (c) $\mathcal{N}(AB) \subseteq \mathcal{N}(A)$.
 (d) $\mathcal{R}(AB) \subseteq \mathcal{R}(B)$. (e) $\mathcal{R}(AB) \subseteq \mathcal{R}(A)$. (f) $\mathcal{R}(B) \subseteq \mathcal{R}(AB)$.
- 1.6** (a) What are the four Penrose conditions for X to be the pseudo-inverse of A ?
 (b) A matrix X is said to be a **left-inverse** if $XA = I$. Show that a left-inverse is an $\{1, 2, 3\}$ -inverse, i.e. satisfies the Penrose conditions (1), (2), and (3). Similarly, show that a **right-inverse** is an $\{1, 2, 4\}$ -inverse.
- 1.7** Let the singular values of $A \in \mathbf{R}^{m \times n}$ be $\sigma_1 \geq \dots \geq \sigma_n$. What relations are satisfied between these and the singular values of
- $$\tilde{A} = (A, u), \quad \hat{A} = \begin{pmatrix} A \\ v^T \end{pmatrix}?$$
- 1.8** (a) Show that $A^\dagger = A^{-1}$ when A is a nonsingular matrix.
 (b) Construct an example where $G \neq A^\dagger$ despite the fact that $GA = I$.

Problems

- 1.1** (a) Compute the pseudo-inverse x^\dagger of a column vector x .
 (b) Take $A = \begin{pmatrix} 1 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 1 \end{pmatrix}^T$, and show that $1 = (AB)^\dagger \neq B^\dagger A^\dagger = 1/2$.
- 1.2** (a) Verify that the Penrose conditions uniquely defines the matrix X . Do it first for $A = \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$, and then transform the result to a general matrix A .
- 1.3** (a) Show that if $w \in \mathbf{R}^n$ and $w^T w = 1$, then the matrix $P(w) = I - 2ww^T$ is both symmetric and orthogonal.
 (b) Given two vectors $x, y \in \mathbf{R}^n$, $x \neq y$, $\|x\|_2 = \|y\|_2$, then

$$(w)x = y, \quad w = (y - x)/\|y - x\|_2.$$
- 1.4** Let $S \subseteq \mathbf{R}^n$ be a subspace, P_1 and P_2 be orthogonal projections onto $S = \mathcal{R}(P_1) = \mathcal{R}(P_2)$. Show that $P_1 = P_2$, i.e., the orthogonal projection onto S is

unique.

Hint: Show that for any $z \in \mathbf{R}^n$

$$\|(P_1 - P_2)z\|_2^2 = (P_1 z)^T(I - P_2)z + (P_2 z)^T(I - P_1)z = 0.$$

- 1.5** (R. E. Cline) Let A and B be any matrices for which the product AB is defined, and set

$$B_1 = A^\dagger AB, \quad A_1 = AB_1 B_1^\dagger.$$

Show that $AB = AB_1 = A_1 B_1$ and that $(AB)^\dagger = B_1^\dagger A_1^\dagger$.

Hint: Use the Penrose conditions.

- 1.6** (a) Show that the matrix $A \in \mathbf{R}^{m \times n}$ has a **left inverse** $A^L \in \mathbf{R}^{n \times m}$, 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.
(c) show the relation $\text{rank}(A) = \text{trace}(A^\dagger A)$.

- 1.7** Show that A^\dagger minimizes $\|AX - I\|_F$.

- 1.8** Prove *Bjerhammar's characterization*: Let A have full column rank and let B be any matrix such that $A^T B = 0$ and $(A \ B)$ is nonsingular. Then $A^\dagger = X^T$ where

$$\begin{pmatrix} X^T \\ Y^T \end{pmatrix} = (A \ B)^{-1}.$$

8.2 The Method of Normal Equations

8.2.1 Forming and Solving the Normal Equations

Consider the linear Gauss–Markov model

$$Ax = b + \epsilon, \quad A \in \mathbf{R}^{m \times n}, \quad (8.2.1)$$

where ϵ has zero mean and variance-covariance matrix equal to $\sigma^2 I$. By the Gauss–Markov theorem x is a least squares estimate if and only if it satisfies the normal equations $A^T Ax = A^T b$.

Theorem 8.2.1.

The matrix $A^T A$ is positive definite if and only if the columns of A are linearly independent, i.e., when $\text{rank}(A) = n$. In this case the least squares solution is unique and given by

$$x = (A^T A)^{-1} A^T b, \quad r = (I - A(A^T A)^{-1} A^T)b. \quad (8.2.2)$$

Proof. If the columns of A are linearly independent, then $x \neq 0 \Rightarrow Ax \neq 0$. Therefore, $x \neq 0 \Rightarrow x^T A^T Ax = \|Ax\|_2^2 > 0$, and hence $A^T A$ is positive definite. On the other hand, if the columns are linearly dependent, then for some $x_0 \neq 0$ we have

$Ax_0 = 0$. Then $x_0^T A^T Ax_0 = 0$, and therefore $A^T A$ is not positive definite. When $A^T A$ is positive definite it is also nonsingular and (8.2.2) follows. \square

After forming $A^T A$ and $A^T b$ the normal equations can be solved by symmetric Gaussian elimination (which Gauss did), or by computing the Cholesky factorization (due to [45])

$$A^T A = R^T R,$$

where R is upper triangular.

We now discuss some details in the numerical implementation of the method of normal equations. We defer treatment of rank deficient problems to later and assume throughout this section that the numerical rank of A equals n . The first step is to compute the elements of the symmetric matrix $C = A^T A$ and the vector $d = A^T b$. If $A = (a_1, a_2, \dots, a_n)$ has been partitioned by columns, we can use the inner product formulation

$$c_{jk} = (A^T A)_{jk} = a_j^T a_k, \quad d_j = (A^T b)_j = a_j^T b, \quad 1 \leq j \leq k \leq n. \quad (8.2.3)$$

Since C is symmetric it is only necessary to compute and store its lower (or upper) triangular which requires $\frac{1}{2}mn(n+1)$ multiplications. Note that if $m \gg n$, then the number of elements $\frac{1}{2}n(n+1)$ in the upper triangular part of $A^T A$ is much smaller than the number mn of elements in A . Hence, in this case the formation of $A^T A$ and $A^T b$ can be viewed as a *data compression*!

In the inner product formulation (8.2.3) the data A and b are accessed columnwise. This may not always be suitable since each column needs to be accessed many times. For example, if A is so large that it is held in secondary storage, this will be expensive. In an alternative row oriented algorithm can be used, where outer products of the rows are accumulated. Denoting the i th row of A by \tilde{a}_i^T , $i = 1 : m$, we have

$$C = A^T A = \sum_{i=1}^m \tilde{a}_i \tilde{a}_i^T, \quad d = A^T b = \sum_{i=1}^m b_i \tilde{a}_i. \quad (8.2.4)$$

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 *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 $A^T A$ and $A^T b$. This outer product form is also preferable if the matrix A is sparse; see the hint to Problem 7.7.1. Note that both formulas can be combined if we adjoin b as the $(n+1)$ st column to A and form

$$(A, b)^T (A, b) = \begin{pmatrix} A^T A & A^T b \\ b^T A & b^T b \end{pmatrix}.$$

The matrix $C = A^T A$ is symmetric, and if $\text{rank}(A) = n$ it is also positive definite. Gauss solved the normal equations by symmetric Gaussian elimination. Computing the Cholesky factorization

$$C = A^T A = R^T R, \quad R \in \mathbf{R}^{n \times n}, \quad (8.2.5)$$

is now the standard approach. The Cholesky factor R is upper triangular and nonsingular and can be computed by one of the algorithms given in Section 7.4.2. The least squares solution is then obtained by solving the two triangular systems

$$R^T z = d, \quad Rx = z. \quad (8.2.6)$$

Forming the matrix $A^T A$ and computing its Cholesky factorization requires (neglecting lower order terms) $mn^2 + \frac{1}{3}n^3$ flops. If we have several right hand sides b_i , $i = 1 : p$, then the Cholesky factorization need only be computed once. Forming $A^T b_i$ and solving the two triangular systems requires $mn + n^2$ additional flops for each right hand side.

Example 8.2.1.

In statistics, the **multiple linear regression** problem is the problem of fitting a given linear model to a set of data points (x_i, y_i) , $i = 1 : m$. Assume that the model to be fitted is $y = \alpha + \beta x$. Substituting the data gives m linear equations for the unknown parameters α and β

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}.$$

The least squares solution is obtained by forming the normal equations

$$\begin{pmatrix} m & m\bar{x} \\ m\bar{x} & x^T x \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} m\bar{y} \\ x^T y \end{pmatrix}. \quad (8.2.7)$$

where

$$\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i, \quad \bar{y} = \frac{1}{m} \sum_{i=1}^m y_i,$$

are the mean values. Eliminating α we obtain the “classical” formulas

$$\beta = (x^T y - m\bar{y}\bar{x}) / (x^T x - m\bar{x}^2), \quad (8.2.8)$$

The first equation in (8.2.7) gives $\bar{y} = \alpha + \beta\bar{x}$. This shows that (\bar{y}, \bar{x}) lies on the fitted line and we have

$$\alpha = \bar{y} - \beta\bar{x}. \quad (8.2.9)$$

A more accurate formula for β is obtained by first subtracting out the mean values from the data and writing the model as $y - \bar{y} = \beta(x - \bar{x})$. In the new variables the matrix of normal equation is *diagonal*, and we find

$$\beta = \sum_{i=1}^m (y_i - \bar{y})(x_i - \bar{x})_i / \sum_{i=1}^m (x_i - \bar{x})^2. \quad (8.2.10)$$

A drawback of this formula is that it requires two passes through the data.

This is easily generalized to fitting a general linear model with $n < m$ parameters which leads to a linear least squares problem $\min_x \|y - Ax\|_2$. In statistics the vector y is called the response variable and the parameters are the explanatory variables.

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. For such a matrix we define:

Definition 8.2.2.

Let f_i and l_i be the column subscripts of the first and last nonzero in the i th row of the matrix $A \in \mathbf{R}^{m \times n}$, i.e.,

$$f_i = \min\{j \mid a_{ij} \neq 0\}, \quad l_i = \max\{j \mid a_{ij} \neq 0\}. \quad (8.2.11)$$

Then A is said to have row bandwidth w , where

$$w = \max_{1 \leq i \leq m} w_i, \quad w_i = (l_i - f_i + 1). \quad (8.2.12)$$

For this structure to have practical significance we need to have $w \ll n$. Matrices of small row bandwidth often occur naturally, since they correspond to a situation where only variables "close" to each other are coupled by observations. We now prove a relation between the row bandwidth of the matrix A and the bandwidth of the corresponding matrix of normal equations $A^T A$.

Theorem 8.2.3.

Assume that the matrix $A \in \mathbf{R}^{m \times n}$ has row bandwidth w . Then the symmetric matrix $A^T A$ has bandwidth $r \leq w - 1$.

Proof. From the Definition 8.2.2 it follows that $a_{ij}a_{ik} \neq 0$ implies that $|j - k| < w$ and thus

$$|j - k| \geq w \Rightarrow a_{ij}a_{ik} = 0 \quad \forall i = 1 : m. \quad (8.2.13)$$

and hence

$$(A^T A)_{jk} = \sum_{i=1}^m a_{ij}a_{ik} = 0.$$

□

If the matrix A also has full column rank it follows that we can use the band Cholesky Algorithm 7.7 to solve the normal equations.

8.2.2 Computing the Covariance Matrix.

Consider a general univariate linear model with error covariance matrix equal to a symmetric positive definite matrix $\sigma^2 \mathcal{V}$. By Theorem 8.1.7 the least squares

estimate is the solution to

$$\min_x (Ax - b)^T V^{-1} (Ax - b).$$

and the covariance matrix of the solution x is $V_x = \sigma^2 C_x$, where

$$C_x = (A^T V^{-1} A)^{-1} = (R^T R)^{-1} = R^{-1} R^{-T}. \quad (8.2.14)$$

Here R is the Cholesky factor of $A^T V^{-1} A$. In the special case of a weighted least squares problem $V^{-1} = D^2$ is a diagonal matrix with positive elements.

An unbiased estimate of σ^2 is given by

$$s^2 = \frac{1}{m-n} (b - Ax)^T V^{-1} (b - Ax),. \quad (8.2.15)$$

The least squares residual vector $r = b - A\hat{x}$ has variance-covariance matrix equal to $\sigma^2 V_r$, where

$$V_r = V^{-1/2} A (A^T V^{-1} A)^{-1} A^T V^{-1/2} = BB^T, \quad B = V^{-1/2} AR^{-1}. \quad (8.2.16)$$

The **normalized residuals**

$$\tilde{r} = \frac{1}{s} (\text{diag } V_r)^{-1/2} \hat{r}$$

are often used to detect and identify bad data, which is assumed to correspond to large components in \tilde{r} . If the error covariance matrix is correct, then the components of \tilde{r} should be uniformly distributed random quantities. In particular, the histogram of the entries of the residual should look like a bell curve.

In order to assess the accuracy of the computed estimate of x it is often required to compute the matrix C_x or part of it. If, for simplicity, we assume that $V = I$, then C_x is obtained from

$$C_x = SS^T, \quad S = R^{-1}.$$

The upper triangular matrix S is computed by solving the matrix equation $RS = I$. Using the algorithm given in (7.2.41) this requires $n^3/3$ flops. The upper triangular part of the symmetric matrix SS^T can then be formed. The computation of C_x can be sequenced so that the elements of C_x overwrite those of R and requires a total of $2n^3/3$ flops.

In many situations the variance of a linear functional $\varphi = f^T \hat{x}$ of the least squares solution is of interest. This equals

$$\mathcal{V}(\varphi) = f^T V_x f = \sigma^2 f^T R^{-1} R^{-T} f = \sigma^2 z^T z, \quad (8.2.17)$$

where $z = R^{-T} f$. Here the matrix C_x occurs only as an intermediate quantity and the variance can be computed by solving the lower triangular system $R^T z = f$ by forward substitution and then forming $\sigma^2 z^T z = \sigma^2 \|z\|_2^2$. This is a more stable and efficient approach than using the expression involving V_x . In particular, the variance of a single component $x_i = e_i^T x$ of the solution is obtained from

$$\mathcal{V}(x_i) = \sigma^2 z^T z, \quad R^T z = e_i, \quad i = 1 : n. \quad (8.2.18)$$

Note that since R^T is lower triangular, z will have $i - 1$ leading zeros. For $i = n$ only the last component is nonzero and equals r_{nn}^{-1} .

Example 8.2.2.

In an early application of least squares Laplace [394] 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 Alexis Bouvard (1767–1843), French astronomer and director of the Paris Observatory. The final normal equations are $A^T A x = A^T b$, where

$$A^T A = \begin{pmatrix} 795938 & -12729398 & 6788.2 & -1959.0 & 696.13 & 2602 \\ -12729398 & 424865729 & -153106.5 & -39749.1 & -5459 & 5722 \\ 6788.2 & -153106.5 & 71.8720 & -3.2252 & 1.2484 & 1.3371 \\ -1959.0 & -153106.5 & -3.2252 & 57.1911 & 3.6213 & 1.1128 \\ 696.13 & -5459 & 1.2484 & 3.6213 & 21.543 & 46.310 \\ 2602 & 5722 & 1.3371 & 1.1128 & 46.310 & 129 \end{pmatrix},$$

$$A^T b = \begin{pmatrix} 7212.600 \\ -738297.800 \\ 237.782 \\ -40.335 \\ -343.455 \\ -1002.900 \end{pmatrix}.$$

In these equations the mass of Uranus is $(1 + x_1)/19504$, the mass of Jupiter is $(1 + x_2)/1067.09$ if the mass of the sun is taken as unity. Bouvard also gave the square residual norm as

$$\|b - A\hat{x}\|_2^2 = 31096.$$

Working from these normal equations Laplace computed the least squares solution and obtained

$$x_1 = 0.0895435, \quad x_2 = -0.0030431.$$

From this we can calculate the mass of Jupiter and Uranus as a fraction of the mass of the sun. We obtain 1070.3 for Saturn and 17918 for Uranus. The covariance matrix of the solution is $V_x = s^2(R^T R)^{-1}$, where $s^2 = 31096/(129 - 6)$ is an unbiased estimate of σ^2 . We get

$$v_{11} = 0.5245452 \cdot 10^{-2}, \quad v_{22} = 4.383233 \cdot 10^{-6}.$$

This shows that the computed mass of Jupiter is very reliable, while the computed mass of Uranus is not. Laplace concludes that with a probability of $1 - 10^{-6}$ the error in the computed mass of Jupiter is less than one per cent.

There is an alternative way of computing C_x without inverting R . We have from (8.2.17), multiplying by R from the left,

$$RC_x = R^{-T}. \tag{8.2.19}$$

The diagonal elements of R^{-T} are simply r_{kk}^{-1} , $k = n, \dots, 1$, and since R^{-T} is lower triangular it has $\frac{1}{2}n(n - 1)$ zero elements. Hence, $\frac{1}{2}n(n + 1)$ elements of R^{-T} are

known and the corresponding equations in (8.2.19) suffice to determine the elements in the upper triangular part of the symmetric matrix C_x .

To compute the elements in the last column c_n of C_x we solve the system

$$Rc_n = r_{nn}^{-1}e_n, \quad e_n = (0, \dots, 0, 1)^T$$

by back-substitution. This gives

$$c_{nn} = r_{nn}^{-2}, \quad c_{in} = -r_{ii}^{-1} \sum_{j=i+1}^n r_{ij}c_{jn}, \quad i = n-1, \dots, 1. \quad (8.2.20)$$

By symmetry $c_{ni} = c_{in}$, $i = n-1, \dots, 1$, so we know also the last row of C_x . Now assume that we have computed the elements $c_{ij} = c_{ji}$, $j = n, \dots, k+1$, $i \leq j$. We next determine the elements c_{ik} , $i \leq k$. We have

$$c_{kk}r_{kk} + \sum_{j=k+1}^n r_{kj}c_{jk} = r_{kk}^{-1},$$

and since the elements $c_{kj} = c_{jk}$, $j = k+1 : n$, have already been computed,

$$c_{kk} = r_{kk}^{-1} \left(r_{kk}^{-1} - \sum_{j=k+1}^n r_{kj}c_{kj} \right). \quad (8.2.21)$$

Similarly, for $i = k-1 : (-1) : 1$,

$$c_{ik} = -r_{ii}^{-1} \left(\sum_{j=i+1}^k r_{ij}c_{jk} + \sum_{j=k+1}^n r_{ij}c_{kj} \right). \quad (8.2.22)$$

Using the formulas (8.2.20)–(8.2.22) all the elements of C_x can be computed in about $2n^3/3$ flops.

When the matrix R is sparse, Golub and Plemmons [260] have shown that the same algorithm can be used very efficiently to compute *all elements in C_x , associated with nonzero elements in R* . Since R has a nonzero diagonal this includes the diagonal elements of C_x giving the variances of the components x_i , $i = 1 : n$. If R has bandwidth w , then the corresponding elements in C_x can be computed in only $2nw^2$ flops; see Björck [61, Sec 6.7.4].

8.2.3 Perturbation Bounds for Least Squares Problems

We now consider the effect of perturbations of A and b on the least squares solution x . In this analysis the condition number of the matrix $A \in \mathbf{R}^{m \times n}$ will play a significant role. The following definition generalizes the condition number (6.6.3) of a square nonsingular matrix.

Definition 8.2.4.

Let $A \in \mathbf{R}^{m \times n}$ have rank $r > 0$ and singular values equal to $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$. Then the condition number of A is

$$\kappa(A) = \|A\|_2 \|A^\dagger\|_2 = \sigma_1/\sigma_r,$$

where the last equality follows from the relations $\|A\|_2 = \sigma_1$, $\|A^\dagger\|_2 = \sigma_r^{-1}$.

Using the SVD $A = U\Sigma V^T$ we obtain

$$A^T A = V \Sigma^T (U^T U) \Sigma V^T = V \begin{pmatrix} \Sigma_r^2 & 0 \\ 0 & 0 \end{pmatrix} V^T. \quad (8.2.23)$$

Hence, $\sigma_i(A^T A) = \sigma_i^2(A)$, and it follows that $\kappa(A^T A) = \kappa^2(A)$. This shows that *the matrix of the normal equations has a condition number which is the square of the condition number of A* .

We now give a first order perturbation analysis for the least squares problem when $\text{rank}(A) = n$. Denote the perturbed data $A + \delta A$ and $b + \delta b$ and assume that δA sufficiently small so that $\text{rank}(A + \delta A) = n$. Let the perturbed solution be $x + \delta x$ and $r + \delta r$, where $r = b - Ax$ is the residual vector. The perturbed solution satisfies the augmented system

$$\begin{pmatrix} I & A + \delta A \\ (A + \delta A)^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{s} \\ \tilde{x} \end{pmatrix} = \begin{pmatrix} b + \delta b \\ 0 \end{pmatrix}. \quad (8.2.24)$$

Subtracting the unperturbed equations and neglecting second order quantities the perturbations $\delta s = \delta r$ and δx satisfy the linear system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \delta s \\ \delta x \end{pmatrix} = \begin{pmatrix} \delta b - \delta Ax \\ -\delta A^T s \end{pmatrix}. \quad (8.2.25)$$

From the Schur–Banachiewicz formula (see Section 7.1.5) it follows that the inverse of the matrix in this system equals

$$\begin{aligned} \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}^{-1} &= \begin{pmatrix} (I - A(A^T A)^{-1} A^T) & A(A^T A)^{-1} \\ (A^T A)^{-1} A^T & -(A^T A)^{-1} \end{pmatrix} \\ &= \begin{pmatrix} P_{N(A^T)} & (A^\dagger)^T \\ A^\dagger & -(A^T A)^{-1} \end{pmatrix}. \end{aligned} \quad (8.2.26)$$

We find that the perturbed solution satisfies

$$\delta x = A^\dagger(\delta b - \delta Ax) + (A^T A)^{-1} \delta A^T r, \quad (8.2.27)$$

$$\delta r = P_{N(A^T)}(\delta b - \delta Ax) - (A^\dagger)^T \delta A^T r. \quad (8.2.28)$$

Assuming that the perturbations δA and δb satisfy

$$|\delta A| \leq \omega E, \quad |\delta b| \leq \omega f, \quad (8.2.29)$$

and substituting in (8.2.27)–(8.2.28) yields the componentwise bounds

$$|\delta x| \lesssim \omega (|A^\dagger|(f + E|x|) + |(A^T A)^{-1}|E^T|r|), \quad (8.2.30)$$

$$|\delta r| \lesssim \omega (|I - AA^\dagger|(f + E|x|) + |(A^\dagger)^T|E^T|r|). \quad (8.2.31)$$

where terms of order $O(\omega^2)$ have been neglected. Note that if the system $Ax = b$ is consistent, then $r = 0$ and the bound for $|\delta x|$ is identical to that obtained for a square nonsingular linear system.

Taking norms in (8.2.27) and (8.2.28) and using

$$\|A^\dagger\|_2 = \|(A^\dagger)^T\|_2 = 1/\sigma_n, \quad \|(A^T A)^{-1}\|_2 = 1/\sigma_n^2, \quad \|P_{N(A^T)}\|_2 = 1.$$

it follows that

$$\|\delta x\|_2 \lesssim \frac{1}{\sigma_n} \|\delta b\|_2 + \frac{1}{\sigma_n} \|\delta A\|_2 \left(\|x\|_2 + \frac{1}{\sigma_n} \|r\|_2 \right), \quad (8.2.32)$$

$$\|\delta r\|_2 \lesssim \|\delta b\|_2 + \|\delta A\|_2 \left(\|x\|_2 + \frac{1}{\sigma_n} \|r\|_2 \right), \quad (8.2.33)$$

If $r \neq 0$ there is a term proportional to σ_n^{-2} present in the bound for $\|\delta x\|_2$. A more refined perturbation analysis (see Wedin [606]) shows that if

$$\eta = \|A^\dagger\|_2 \|\delta A\|_2 \ll 1.$$

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 an upper bound for the normwise relative perturbation

$$\frac{\|\delta x\|_2}{\|x\|_2} \leq \kappa_{LS} \frac{\|\delta A\|_2}{\|A\|_2}, \quad \kappa_{LS} = \kappa(A) \left(1 + \frac{\|r\|_2}{\sigma_n \|x\|_2} \right) \quad (8.2.34)$$

Hence, κ_{LS} is the condition number for the least squares problem. The following two important facts should be noted:

- κ_{LS} depends not only on A but also on $r = P_{N(A^T)}b$.
- If $\|r\|_2 \ll \sigma_n \|x\|_2$, then $\kappa_{LS} \approx \kappa(A)$, but if $\|r\|_2 > \sigma_n \|x\|_2$ the second term in (8.2.34) will dominate,

Example 8.2.3.

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

$$x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \delta x = \frac{2}{5\delta} \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad r = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \delta r = -\frac{1}{5} \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}.$$

For this right hand side $\|x\|_2 = \|r\|_2$ and $\kappa_{LS} = 1/\delta + 1/\delta^2 \approx \kappa^2(A)$. This is reflected in the size of δx .

If instead we take $\alpha = \delta$, then a short calculation shows that $\|r\|_2/\|x\|_2 = \delta$ and $\kappa_{LS} = 2/\delta$. The same perturbation δA now gives

$$\delta x = \frac{2}{5} \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \delta r = -\frac{\delta}{5} \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}.$$

It should be stressed that in order for the perturbation analysis above to be useful, the matrix A and vector 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 $A = (a_1, a_2, \dots, a_n)$ have widely differing norms. Then a much better estimate may often be obtained by applying (8.2.34) to the scaled problem $\min_{\tilde{x}} \|\tilde{A}\tilde{x} - b\|_2$, chosen so that \tilde{A} has columns of unit length, i.e.,

$$\tilde{A} = AD^{-1}, \quad \tilde{x} = Dx, \quad D = \text{diag}(\|a_1\|_2, \dots, \|a_n\|_2).$$

By Theorem 8.2.5 this column scaling approximately minimizes $\kappa(AD^{-1})$ over $D > 0$. Note that scaling the columns changes also the norm in which the error in the original variables x is measured.

If the *rows* in A differ widely in norm, then (8.2.34) may also considerably overestimate the perturbation in x . As remarked above, we cannot scale the rows in A without changing the least squares solution.

Perturbation bounds with better scaling properties can be obtained by considering componentwise perturbations; see Section 8.2.4.

8.2.4 Stability and Accuracy with Normal Equations

We now turn to a discussion of the accuracy of the method of normal equations for least squares problems. First we consider rounding errors in the formation of the system of normal equations. Using the standard model for floating point computation the computed matrix $\bar{C} = fl(A^T A)$ satisfies

$$\bar{C} = fl(A^T A) = C + E, \quad |E| < \gamma_m |A|^T |A|. \quad (8.2.35)$$

where (see Lemma 7.1.21) $|\gamma_m| < mu/(1 - mu)u$ and u is the unit roundoff. A similar estimate holds for the rounding errors in the computed vector $A^T b$.

It should be emphasized that it is *not* in general possible to show that $\bar{C} = (A + E)^T (A + E)$ for some small error matrix E . That is, the rounding errors performed in forming the matrix $A^T A$ are not in general equivalent to small perturbations of

the initial data matrix A . This means that it is not in general true that the solution computed from the normal equations equals the exact solution to a problem where the data A and b have been perturbed by small amounts. In other words, *the method of normal equations is not backwards stable*. In spite of this it often gives satisfactory results. However, as the following example illustrates, when $A^T A$ is ill-conditioned, *it might be necessary to use extra precision in forming and solving the normal equations in order to avoid loss of significant information in the data.*

Example 8.2.4.

Läuchli [398]: 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.

Least squares problems of this form are called **stiff**. They occur when the error in some equations (here $x_1 + x_2 + x_3 = 1$) have a much smaller variance than in the others; see Section 8.6.1.

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. Using the backward error bound given in Theorem 7.5.16 a perturbation analysis shows that provided $2n^{3/2}u\kappa(A^T A) < 0.1$, an upper bound for the error in the computed solution \bar{x} is

$$\|\bar{x} - x\|_2 \leq 2.5n^{3/2}u\kappa(A^T A)\|x\|_2. \quad (8.2.36)$$

As seen in Section 8.2.4, for “small” residual least squares problem the true condition number is approximately $\kappa(A) = \kappa^{1/2}(A^T A)$. In this case, *the system of normal equations can be much worse conditioned than the least squares problem from which it originated.*

Sometimes ill-conditioning is caused by an unsuitable formulation of the problem. Then a different choice of parameterization can significantly reduce the condition number. In approximation problems one should try to use orthogonal, or

nearly orthogonal, base functions. In case the elements in A and b are the original data the ill-conditioning cannot be avoided in this way. In Secs. 8.3 and 8.4 we consider methods for solving least squares problems based on orthogonalization. These methods work directly with A and b and are backwards stable.

In Section 7.7.7 we discussed 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 $\min_x \|Ax - b\|_2$ a row scaling of (A, b) is not allowed since such a scaling would change the exact solution. However, we can scale the columns of A . If we take $x = Dx'$, the normal equations will change into

$$(AD)^T(AD)x' = D(A^TA)Dx' = DA^Tb.$$

Hence, this corresponds to a *symmetric scaling* of rows and columns in A^TA . It is important to note that if the Cholesky algorithm is carried out without pivoting the computed solution is *not* affected by such a scaling, cf. Theorem 7.5.6. This means that even if no explicit scaling is carried out, the rounding error estimate (8.2.36) for the computed solution \bar{x} holds for *all* D ,

$$\|D(\bar{x} - x)\|_2 \leq 2.5n^{3/2}u\kappa(DA^TAD)\|Dx\|_2.$$

(Note, however, 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

$$\kappa'(A^TA) = \min_{D>0} \kappa(DA^TAD). \quad (8.2.37)$$

The following result by van der Sluis [1969] shows the scaling where D is chosen so that in AD all column norms are equal, i.e. $D = \text{diag}(\|a_1\|_2, \dots, \|a_n\|_2)^{-1}$, comes within a factor of n of the minimum value.

Theorem 8.2.5. *Let $C \in \mathbf{R}^{n \times n}$ be a symmetric and positive definite matrix, and denote by \mathcal{D} the set of $n \times n$ nonsingular diagonal matrices. Then if in C all diagonal elements are equal, and C has at most q nonzero elements in any row, it holds that*

$$\kappa(C) \leq q \min_{D \in \mathcal{D}} \kappa(DCD).$$

As the following example shows, this 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!

Example 8.2.5. The matrix $A \in R^{21 \times 6}$ 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 $p(t) = x_0 + x_1t + x_2t^2 + \dots + x_5t^5$ to observations at points $x_i = 0, 1, \dots, 20$. 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 Theorem 8.2.5. Thus, the condition number of the matrix of normal equations is reduced by about seven orders of magnitude by this scaling!

8.2.5 Backward Error Analysis

An algorithm for solving the linear least squares problem is said to numerically stable if for any data A and b , there exist small perturbation matrices and vectors δA and δb such that *the computed solution \bar{x} is the exact solution to*

$$\min_x \|(A + \delta A)x - (b + \delta b)\|_2, \quad (8.2.38)$$

where $\|\delta A\| \leq \tau \|A\|$, $\|\delta b\| \leq \tau \|b\|$, with τ being a small multiple of the unit round-off u . We shall see that methods in which the normal equations are explicitly formed cannot be backward stable. On the other hand, many methods based on orthogonal factorizations have been proved to be backward stable.

Any computed solution \bar{x} is called a stable solution if it satisfies (8.2.38). This does not mean that \bar{x} is close to the exact solution x . If the least squares problem is ill-conditioned then a stable solution can be very different from x . For a stable solution the error $\|x - \bar{x}\|$ can be estimated using the perturbation results given in Section 8.2.4.

Many special fast methods exist for solving structured least squares problems, e.g., where A is a Toeplitz matrix. These methods cannot be proved to be backward stable, which is one reason why a solution to the following problem is of interest:

For a consistent linear system we derived in Section 7.5.2 a simple posteriori bounds for the the smallest backward error of a computed solution \bar{x} . The situation is more difficult for the least squares problem.

Given an alleged solution \tilde{x} , we want to find a perturbation δA of smallest norm such that \tilde{x} is the exact solution to the perturbed problem

$$\min_x \|(b + \delta b) - (A + \delta A)x\|_2. \quad (8.2.39)$$

If we could find the backward error of smallest norm, this could be used to verify numerically the stability properties of an algorithm. There is not much loss in assuming that $\delta b = 0$ in (8.2.40). Then the optimal backward error in the Frobenius norm is

$$\eta_F(\tilde{x}) = \min\{\|\delta A\|_F \mid \tilde{x} \text{ solves } \min_x \|b - (A + \delta A)x\|_2\}. \quad (8.2.40)$$

Thus, the optimal backward error can be found by characterizing the set of all backward perturbations and then finding an optimal bound, which minimizes the Frobenius norm.

Theorem 8.2.6. Let \tilde{x} be an alleged solution and $\tilde{r} = b - A\tilde{x} \neq 0$. The optimal backward error in the Frobenius norm is

$$\eta_F(\tilde{x}) = \begin{cases} \|A^T \tilde{r}\|_2 / \|\tilde{r}\|_2, & \text{if } \tilde{x} = 0, \\ \min \{\eta, \sigma_{\min}([A \quad C])\} & \text{otherwise.} \end{cases} \quad (8.2.41)$$

where

$$\eta = \|\tilde{r}\|_2 / \|\tilde{x}\|_2, \quad C = I - (\tilde{r}\tilde{r}^T) / \|\tilde{r}\|_2^2$$

and $\sigma_{\min}([A \quad C])$ denotes the smallest (nonzero) singular value of the matrix $[A \quad C] \in \mathbb{R}^{m \times (n+m)}$.

The task of computing $\eta_F(\tilde{x})$ is thus reduced to that of computing $\sigma_{\min}([A \quad C])$. Since this is expensive, approximations that are accurate and less costly have been derived. If a QR factorization of A is available lower and upper bounds for $\eta_F(\tilde{x})$ can be computed in only $\mathcal{O}(mn)$ operations. Let $r_1 = P_{\mathcal{R}(A)}\tilde{r}$ be the orthogonal projection of \tilde{r} onto the range of A . If $\|r_1\|_2 \leq \alpha\|\tilde{r}\|_2$ it holds that

$$\frac{\sqrt{5}-1}{2} \tilde{\sigma}_1 \leq \eta_F(\tilde{x}) \leq \sqrt{1 + \alpha^2} \tilde{\sigma}_1, \quad (8.2.42)$$

where

$$\tilde{\sigma}_1 = \|(A^T A + \eta I)^{-1/2} A^T \tilde{r}\|_2 / \|\tilde{x}\|_2. \quad (8.2.43)$$

Since $\alpha \rightarrow 0$ for small perturbations $\tilde{\sigma}_1$ is an asymptotic upper bound.

A simple way to improve the accuracy of a solution \bar{x} computed by the method of normal equations is by fixed precision iterative refinement; see Section 7.7.8. This requires that the data matrix A is saved and used to compute the residual vector $b - A\bar{x}$. In this way information lost when $A^T A$ was formed can be recovered. If also the corrections are computed from the normal equations, we obtain the following algorithm:

Iterative Refinement with Normal Equations:

Set $x_1 = \bar{x}$, and for $s = 1, 2, \dots$ until convergence do

$$\begin{aligned} r_s &:= b - Ax_s, & R^T R \delta x_s &= A^T r_s, \\ x_{s+1} &:= x_s + \delta x_s. \end{aligned}$$

Here R is computed by Cholesky factorization of the matrix of normal equation $A^T A$. This algorithm only requires one matrix-vector multiplication each with A and A^T and the solution of two triangular systems. Note that the first step, i.e., for $i = 0$, is identical to solving the normal equations. It can be shown that initially the errors will be reduced with rate of convergence equal to

$$\bar{\rho} = c u \kappa'(A^T A), \quad (8.2.44)$$

where c is a constant depending on the dimensions m, n . Several steps of the refinement may be needed to get good accuracy. (Note that $\bar{\rho}$ is proportional to $\kappa'(A^T A)$ even when no scaling of the normal equations has been performed!)

Example 8.2.6.

If $\kappa'(A^T A) = \kappa(A^T A)$ and $c \approx 1$ the error will be reduced to a backward stable level in p steps if $\kappa^{1/2}(A^T A) \leq u^{-p/(2p+1)}$. (As remarked before $\kappa^{1/2}(A^T A)$ is the condition number for a small residual problem.) For example, with $u = 10^{-16}$, the maximum value of $\kappa^{1/2}(A^T A)$ 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 QR factorization described in Sections 8.3 and 8.4 are usually to be preferred.

8.2.6 The Peters–Wilkinson method

Standard algorithms for solving nonsymmetric linear systems $Ax = b$ are usually based on LU factorization with partial pivoting. Therefore it seems natural to consider such factorizations in particular, for least squares problems which are only mildly over- or under-determined, i.e. where $|m - n| \ll n$.

A rectangular matrix $A \in \mathbf{R}^{m \times n}$, $m \geq n$, can be reduced by Gaussian elimination to an upper trapezoidal form U . In general, column interchanges are needed to ensure numerical stability. Usually it will be sufficient to use partial pivoting with a linear independence check. Let $\tilde{a}_{q,p+1}$ be the element of largest magnitude in column $p+1$. If $|\tilde{a}_{q,p+1}| < tol$, 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.

In the full column rank case, $\text{rank}(A) = n$, the resulting LDU factorization becomes

$$\Pi_1 A \Pi_2 = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = LDU = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} DU, \quad (8.2.45)$$

where $L_1 \in \mathbf{R}^{n \times n}$ is unit lower triangular, D diagonal, and $U \in \mathbf{R}^{n \times n}$ is unit upper triangular and nonsingular. Thus, the matrix L has the same dimensions as A and a lower trapezoidal structure. Computing this factorization requires $\frac{1}{2}n^2(m - \frac{1}{3}n)$ flops.

Using the LU factorization (8.2.45) and setting $\tilde{x} = \Pi_2^T x$, $\tilde{b} = \Pi_1 b$, the least squares problem $\min_x \|Ax - b\|_2$ is reduced to

$$\min_y \|Ly - \tilde{b}\|_2, \quad DU\tilde{x} = y. \quad (8.2.46)$$

If partial pivoting by rows is used in the factorization (8.2.45), then L is usually a well-conditioned matrix. In this case the solution to the least squares problem (8.2.46) can be computed from the normal equations

$$L^T Ly = L^T \tilde{b},$$

without substantial loss of accuracy. This is the approach taken by Peters and Wilkinson [484, 1970].

Forming the symmetric matrix $L^T L$ requires $\frac{1}{2}n^2(m - \frac{2}{3}n)$ flops, and computing its Cholesky factorization takes $n^3/6$ flops. Hence, neglecting terms of order n^2 , the total number of flops to compute the least squares solution by the Peters–Wilkinson method is $n^2(m - \frac{1}{3}n)$. Although this is always more expensive than the standard method of normal equations, it is a more stable method as the following example shows. The method is particularly suitable for weighted least squares problems; see Section 8.6.1.

Example 8.2.7. (Noble [451, 1976])

Consider the matrix A and its pseudo-inverse

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 + \epsilon^{-1} \\ 1 & 1 - \epsilon^{-1} \end{pmatrix}, \quad A^\dagger = \frac{1}{6} \begin{pmatrix} 2 & 2 - 3\epsilon^{-1} & 2 + 3\epsilon^{-1} \\ 0 & 3\epsilon^{-1} & -3\epsilon^{-1} \end{pmatrix}.$$

The (exact) matrix of normal equations is

$$A^T A = \begin{pmatrix} 3 & 3 \\ 3 & 3 + 2\epsilon^2 \end{pmatrix}.$$

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. The LU 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. The correct pseudo-inverse is now obtained from

$$A^\dagger = U^{-1} D^{-1} (L^T L)^{-1} L^T = \begin{pmatrix} 1 & -\epsilon \\ 0 & \epsilon \end{pmatrix} \begin{pmatrix} 1/3 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \end{pmatrix}.$$

and there is no cancellation.

As seen in Example 8.2.4 weighted least squares problems of the form

$$\min \left\| \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} x - \begin{pmatrix} \gamma b_1 \\ b_2 \end{pmatrix} \right\|, \quad (8.2.47)$$

where $\gamma \gg 1$, are not suited to a direct application of the method of normal equations. If $p = \text{rank}(A_1)$ steps of Gaussian elimination with pivoting are applied to the resulting factorization can be written

$$\Pi_1 \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} \Pi_2 = LDU, \quad (8.2.48)$$

where Π_1 and Π_2 are permutation matrices, and

$$L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \in \mathbf{R}^{m \times n}, \quad U = \begin{pmatrix} U_{11} & U_{12} \\ & I \end{pmatrix} \in \mathbf{R}^{n \times n}.$$

Here $L_{11} \in \mathbf{R}^{p \times p}$ is unit lower triangular, and $U_{11} \in \mathbf{R}^{p \times p}$ is unit upper triangular. Assuming that A has full rank, D is nonsingular. Then (4.4.1) is equivalent to

$$\min_y \|Ly - \Pi_1 b\|_2, \quad D U \Pi_2^T x = y.$$

The least squares problem in y is usually well-conditioned, since any ill-conditioning from the weights is usually reflected in D . Therefore, it can be solved by forming the normal equations.

Review Questions

- 2.1** Give a necessary and sufficient condition for x to be a solution to $\min_x \|Ax - b\|_2$, and interpret this geometrically. When is the least squares solution x unique? When is $r = b - Ax$ unique?
- 2.2** What are the advantages and drawbacks with the method of normal equations for computing the least squares solution of $Ax = b$? Give a simple example, which shows that loss of information can occur in forming the normal equations.
- 2.3** Discuss how the accuracy of the method of normal equations can be improved by (a) scaling the columns of A , (b) iterative refinement.
- 2.4** Show that the more accurate formula in Example 8.2.1 can be interpreted as a special case of the method (8.5.7)–(8.5.8) for partitioned least squares problems.
- 2.5** (a) Let $A \in \mathbf{R}^{m \times n}$ with $m < n$. Show that $A^T A$ is singular.
(b) Show, using the SVD, that $\text{rank}(A^T A) = \text{rank}(AA^T) = \text{rank}(A)$.
- 2.6** Define the condition number $\kappa(A)$ of a rectangular matrix A . What terms in the perturbation of a least squares solution depend on κ and κ^2 , respectively?

Problems

- 2.1** In order to estimate the height above sea level for three points, A,B, and C, the difference in altitude was measured between these points and points D,E, and F at sea level. The measurements obtained form a linear system in the heights x_A , x_B , and x_C of A,B, and C,

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ x_B \\ x_C \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 1 \end{pmatrix}.$$

Show that the least squares solution and residual vector are

$$x = \frac{1}{4}(5, 7, 12)^T, \quad r = \frac{1}{4}(-1, 1, 0, 2, 3, -3)^T.$$

and verify that the residual vector is orthogonal to all columns in A .

- 2.2** (a) Consider the linear regression problem of fitting $y(t) = \alpha + \beta(t - c)$ by the method of least squares to the data

t	1	3	4	6	7
$f(t)$	-2.1	-0.9	-0.6	0.6	0.9

With the (unsuitable) choice $c = 1,000$ the normal equations

$$\begin{pmatrix} 5 & 4979 \\ 4979 & 4958111 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} -2.1 \\ -2097.3 \end{pmatrix}$$

become very ill-conditioned. Show that if the element 4958111 is rounded to $4958 \cdot 10^3$ then β is perturbed from its correct value 0.5053 to -0.1306 !

(b) As shown in Example 8.2.1, a much better choice of base functions is shifting with the mean value of t , i.e., taking $c = 4.2$. It is not necessary to shift with the *exact* mean. Show that shifting with 4, the midpoint of the interval $(1, 7)$, leads to a very well-conditioned system of normal equations.

- 2.3** Denote by x_V the solution to the weighted least squares problem with covariance matrix V . Let x be the solution to the corresponding unweighted problem ($V = I$). Using the normal equations, show that

$$x_V - x = (A^T V^{-1} A)^{-1} A^T (V^{-1} - I)(b - Ax). \quad (8.2.49)$$

Conclude that weighting the rows affects the solution if $b \notin \mathcal{R}(A)$.

- 2.4** Assume that $\text{rank}(A) = n$, and put $\bar{A} = (A, b) \in \mathbf{R}^{m \times (n+1)}$. Let the corresponding cross product matrix, and its Cholesky factor be

$$\bar{C} = \bar{A}^T \bar{A} = \begin{pmatrix} C & d \\ d^T & b^T b \end{pmatrix}, \quad \bar{R} = \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}.$$

Show that the solution x and the residual norm ρ to the linear least squares problem $\min_x \|b - Ax\|_2$ is given by

$$Rx = z, \quad \|b - Ax\|_2 = \rho.$$

- 2.5** Let $A \in \mathbf{R}^{m \times n}$ 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 $y = Ax$.

- 2.6** (a) Let $A = (A_1 \ A_2) \in \mathbf{R}^{m \times n}$ 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

$$\min_{x_2} \|(I - P_1)(A_2 x_2 - b)\|_2^2,$$

for x_2 and then computing x_1 as the solution to the problem

$$\min_{x_1} \|A_1 x_1 - (b - A_2 x_2)\|_2^2.$$

- 2.7** (S. M. Stiegler [555].) 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 subsections of an arc from Dunkirk to Barcelona. For each subsection the length of the arc S (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 Carcassonne	84424.55	2.96336°	44° 41' 48"
Carcassonne to Barcelona	52749.48	1.85266°	42° 17' 20"

If the earth is ellipsoidal, then to a good approximation it holds

$$z + y \sin^2(L) = S/d,$$

where z and y are unknown parameters. The meridian quadrant then equals $M = 90(z + y/2)$ and the eccentricity is e is found from $1/e = 3(z/y + 1/2)$. Use least squares to determine z and y and then M and $1/e$.

2.8 Show that if $A, B \in \mathbf{R}^{m \times n}$ 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 $\|B - A\|_2 = \epsilon$, $\|B^\dagger - A^\dagger\|_2 > 1/\epsilon$.

2.9 Show that for any matrix A it holds

$$A^\dagger = \lim_{\mu \rightarrow 0} (A^T A + \mu^2 I)^{-1} A^T = \lim_{\mu \rightarrow 0} A^T (A A^T + \mu^2 I)^{-1}. \quad (8.2.50)$$

2.10 (a) Let $A = (a_1, a_2)$, where $a_1^T a_2 = \cos \gamma$, $\|a_1\|_2 = \|a_2\|_2 = 1$. Hence, γ 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

$$A^T A = \begin{pmatrix} 1 & \cos \gamma \\ \cos \gamma & 1 \end{pmatrix}.$$

Then determine the left singular vectors u_1, u_2 from (7.1.33).

(b) Show that if $\gamma \ll 1$, then $\sigma_1 \approx \sqrt{2}$ and $\sigma_2 \approx \gamma/\sqrt{2}$ and

$$u_1 \approx (a_1 + a_2)/2, \quad u_2 \approx (a_1 - a_2)/\gamma.$$

2.11 The least squares problem $\min_x \|Ax - b\|_2$, where

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \epsilon & \epsilon & \epsilon \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

is of the form (8.2.47). Compute the factorization $A = LDU$ that is obtained after one step of Gaussian elimination and show that L and U are well-conditioned. Compute the solution from

$$L^T Ly = L^T b, \quad Ux = D^{-1}y.$$

by solving the system of normal equations for y and solving for x by back-substitution.

8.3 Orthogonal Factorizations

8.3.1 Elementary Orthogonal Matrices

When solving linear equations we made use of elementary elimination matrices of the form $L_j = I + l_j e_j^T$, see (7.2.17). These were used to describe the elementary steps in Gaussian elimination and LU factorization. We now introduce **elementary orthogonal matrices**, which are equal to the unit matrix modified by a matrix of

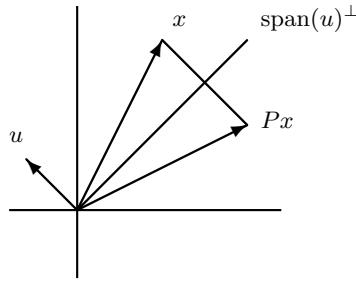


Figure 8.3.1. A Householder reflection of the vector x .

rank one. Such matrices are a flexible and very useful tool for constructing matrix algorithms for a variety of problems in linear algebra. This stems from the fact that because they preserve the Euclidean norm, their use leads to numerically stable algorithms.

Elementary orthogonal matrices of the form

$$P = I - 2 \frac{uu^T}{u^Tu}, \quad u \neq 0, \quad (8.3.1)$$

are called **Householder reflections**. It is easily seen that P is symmetric and setting $\beta = 2/(u^Tu)$, we have

$$P^T P = P^2 = I - 2\beta uu^T + \beta^2 u(u^Tu)u^T = I.$$

It follows that P is orthogonal and $P^{-1} = P$. For any vector x ,

$$Px = (I - \beta uu^T)x = x - \beta(u^Tx)u \in \text{span}[x, u].$$

In particular, $Pu = -u$, that is P reverses u and if $x \perp u$ then $Px = x$.

The effect of the transformation Px for a general vector x is to reflect x in the $(m-1)$ dimensional hyper plane with normal vector u ; see Figure 8.1.3. This is equivalent to subtracting *twice* the orthogonal projection of x onto u . Note that the normal u is parallel to the vector $(x - Px)$.

The use of elementary reflectors in numerical linear algebra was made popular in matrix computation by Householder [341]. Matrices of the form (8.3.1) are therefore often called **Householder reflectors** and the vector u is called a **Householder vector**.

In applications of Householder reflectors the following construction is central. Given a nonzero vector $x \in \mathbf{R}^m$ we want to construct a plane reflection such that *multiplication by P zeros all components except the first in x* , i.e.,

$$Px = \pm \sigma e_1, \quad \sigma = \|x\|_2. \quad (8.3.2)$$

Here e_1 is the first unit vector and the second equation is a consequence of the fact that P is orthogonal. Multiplying (8.3.2) from the left by P and using $P^2 = I$, we find that

$$Pe_1 = \pm x/\|x\|_2,$$

i.e., this task is equivalent to finding a square orthogonal matrix P with its first column proportional to x . It is easily seen that (8.3.2) is satisfied if we take

$$u = \begin{pmatrix} u_1 \\ x_2 \end{pmatrix} \cdot x \mp \sigma e_1 = \begin{pmatrix} \xi_1 \mp \sigma \\ x_2 \end{pmatrix}, \quad x = \begin{pmatrix} \xi_1 \\ x_2 \end{pmatrix}. \quad (8.3.3)$$

Note that the Householder vector u differs from x only in its first component. A short calculation shows that

$$\begin{aligned} \frac{1}{\beta} &= \frac{1}{2} u^T u = \frac{1}{2} (x \mp \sigma e_1)^T (x \mp \sigma e_1) \\ &= \frac{1}{2} (\sigma^2 \mp 2\sigma\xi_1 + \sigma^2) = \sigma(\sigma \mp \xi_1). \end{aligned}$$

If x is close to a multiple of e_1 , then $\sigma \approx |\xi_1|$ and cancellation may lead to a large relative error in β .²⁹ To avoid this we take

$$u_1 = \text{sign}(\xi_1)(|\xi_1| + \sigma), \quad \beta = 1/(\sigma(\sigma + |\xi_1|)), \quad (8.3.4)$$

which gives

$$Px = -\text{sign}(\xi_1)\sigma e_1 = \hat{\sigma}e_1.$$

Note that with this choice of sign the vector $x = e_1$ will be mapped onto $-e_1$.

Algorithm 8.1. *Construct Householder reflection.*

```
function [u,beta,sigma] = house(x)
    % HOUSE computes a Householder reflection
    % P = I - beta*u*u' where u = [1; u2]
    % such that P*x = sigma*e_1;
    u = x;
    sigma == norm(x);
    if sigma = 0
        x(1) = -1; beta = 2;
    else
        u1 = abs(x(1)) + sigma;
        beta = 1/(sigma*u1);
        s1 = sign(x(1));
        u(1) = s1*u1;
        sigma = -s1*sigma;
    end
```

The Householder reflection in (8.3.1) does not depend on the scaling of u . It is often more convenient to scale u so that its first component equals 1. Then $P = I - \beta uu^T$, where

$$u = \begin{pmatrix} 1 \\ u_2 \end{pmatrix}, \quad u_2 = \frac{\text{sign}(\xi_1)}{\sigma + |\xi_1|} x_2, \quad \beta = 1 + \frac{|\xi_1|}{\sigma}. \quad (8.3.5)$$

²⁹It is possible to rewrite the formula in (8.3.4) for β so that the other choice of sign does not give rise to numerical cancellation; see Parlett [478, pp. 91].

(Check this!) This scaling has the advantage that we can stably reconstruct β from u_2 using

$$\beta = 2/(u^T u) = 2/(1 + u_2^T u_2).$$

The following algorithm constructs a Householder reflection $P = I - \beta uu^T$, where $u^T e_1 = 1$, such that

$$Px = -\text{sign}(\xi_1)\sigma e_1, \quad \sigma = \|x\|_2.$$

where $\|x\|_2 = \sigma \neq 0$ and $x^T e_1 = \xi_1$. Note that if $x = 0$, then we can just take $P = I$, i.e. skip the transformation.

If a matrix $A = (a_1, \dots, a_n) \in \mathbf{R}^{m \times n}$ is *premultiplied* by P the product is $PA = (Pa_1, \dots, Pa_n)$, where

$$Pa_j = a_j - \beta(u^T a_j)u. \quad (8.3.6)$$

An analogous formula exists for *postmultiplying* A with P , where P now acts on the *rows* of A . Hence, such products can be computed without explicitly forming P itself. The products

$$PA = A - \beta u(u^T A), \quad AP = A - \beta(Au)u^T,$$

can be computed in $4mn$ flops using one matrix–vector product followed by a rank one update.

Householder reflector can be generalized to the complex case A *unitary Householder matrix* has the form

$$P = I - \beta uu^H, \quad \beta = \frac{2}{u^H u}, \quad u \in \mathbf{C}^n. \quad (8.3.7)$$

It is easy to check that P is Hermitian and unitary

$$P = P^H = P^{-1}.$$

Let $z \in \mathbf{C}^n$ and u be such that

$$Pz = \sigma e_1.$$

Then $|\sigma| = \|z\|_2$, but it is in general not possible to have σ real. Since P is Hermitian $z^H P z = \sigma z^H e_1$ must be real. If we denote the first component of z by $z_1 = e^{i\theta_1}|z_1|$, we can take

$$\sigma = \|z\|_2 e^{i\theta_1}, \quad (8.3.8)$$

Then in (8.3.7) $u = z + \sigma e_1$ and

$$\frac{1}{\beta} = \frac{1}{2}(\|z\|_2^2 + 2|\sigma||z_1| + |\sigma|^2) = \|z\|_2(\|z\|_2 + |z_1|). \quad (8.3.9)$$

Note that with unitary matrices of the form $e^{i\theta_1}$ we can reduce a arbitrary vector to the form ke_1P with k real.

Another useful class of elementary orthogonal transformations are **plane rotations** also called **Givens rotations**. A rotation clockwise through an angle θ in \mathbf{R}^2 is represented by the matrix

$$G = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \quad (8.3.10)$$

Note that $G^{-1}(\theta) = G(-\theta)$, and $\det G(\theta) = +1$. Such transformations are also known as **Givens rotations**.³⁰

In \mathbf{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

$$G_{ij}(\theta) = \begin{pmatrix} 1 & & & & & i & & j \\ & \ddots & & & & & & \\ & & c & & s & & & \\ & & & \ddots & & & & \\ & & & & -s & & c & \\ & & & & & \ddots & & \\ & & & & & & & 1 \end{pmatrix}. \quad (8.3.11)$$

Premultiplying a vector $a = (\alpha_1, \dots, \alpha_n)^T$ by $G_{ij}(\theta)$, $j > i$, will affect only the components $\alpha : i$ and α_j

$$G_{ij} \begin{pmatrix} \alpha_i \\ \alpha_j \end{pmatrix} = \begin{pmatrix} c\alpha_i + s\alpha_j \\ -s\alpha_i + c\alpha_j \end{pmatrix}. \quad (8.3.12)$$

A plane rotation may be multiplied into a vector at a cost of two additions and four multiplications. We can determine the rotation $G_{ij}(\theta)$ so that j th component becomes zero by taking

$$c = \alpha_i / \sigma, \quad s = \alpha_j / \sigma. \quad (8.3.13)$$

Then

$$G_{ij} \begin{pmatrix} \alpha_i \\ \alpha_j \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix}, \quad \sigma = (\alpha_i^2 + \alpha_j^2)^{1/2} \geq 0.$$

The following procedure computes the Givens 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 can just use $-G$ if needed.

Algorithm 8.2.

```
function [c,s,r] = givens(a,b)
% GIVENS computes c and s in a Givens rotation
% Given scalars a and b computes c and s in
% a Givens rotation such that
```

³⁰Named after Wallace Givens, who used them in [257] to reduce matrices to simpler.

```
% 0 = -s*a + c*b, and sigma = c*a + s*b
if b == 0
    c = 1.0; s = 0.0; sigma = a;
else if abs(b) > abs(a)
    t = a/b; tt = sqrt(1+t*t);
    s = 1/tt; c = t*s; sigma = tt*b;
else
    t = b/a; tt = sqrt(1+t*t);
    c = 1/tt; s = t*c; sigma = tt*a;
end
```

This algorithm requires 5 flops and one square root. No trigonometric functions are involved.

Example 8.3.1.

The polar representation of a complex number $z = x + iy$ can be computed by a function call $[c, s, r] = \text{givens}(x, y)$. This gives $z = |r|e^{i\theta}$, where $e^{i\theta} = z/|r|$, and

$$z = \begin{cases} r(c + i s) & \text{if } \sigma \geq 0, \\ |r|(-c + i(-s)) & \text{if } \sigma < 0. \end{cases}$$

Premultiplication of a matrix $A \in R^{m \times n}$ with a Givens rotation G_{ij} will only affect the two rows i and j in A , which are transformed according to

$$a_{ik} := ca_{ik} + sa_{jk}, \quad (8.3.14)$$

$$a_{jk} := -sa_{ik} + ca_{jk}, \quad k = 1 : n. \quad (8.3.15)$$

The product requires $4n$ multiplications and $2n$ additions or $6n$ flops. An analogous algorithm, which only affects columns i and j , exists for postmultiplying A with G_{ij} .

An arbitrary vector $x \in \mathbf{R}^n$ can be transformed into σe_1 with $\sigma = \|x\|_2^2 \geq 0$, using a sequence of Givens rotations. Let G_{1k} , $k = 2 : m$ be a sequence of Givens rotations, where G_{1k} is determined to zero the k th component in the vector a . Then

$$G_{1n} \dots G_{13} G_{12} a = \sigma e_1.$$

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 (8.3.10) has determinant equal to $+1$. We could equally well work with Givens reflectors which have the form

$$G = \begin{pmatrix} -\cos \theta & \sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \quad (8.3.16)$$

Using trigonometric identities

$$G = I - (I - G) = I - 2uu^T, \quad u = \begin{pmatrix} \sin(\theta/2) \\ \cos(\theta/2) \end{pmatrix}.$$

which shows the relation to 2×2 Householder matrices.

For complex transformations we can use **unitary** Givens rotations of the form

$$G = \begin{pmatrix} \bar{c} & \bar{s} \\ -s & c \end{pmatrix}, \quad c = e^{i\gamma} \cos \theta, \quad s = e^{i\delta} \sin \theta. \quad (8.3.17)$$

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 $(x_1 \ x_2)^T \in \mathbf{C}^2$ we have

$$G \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \bar{c}z_1 + \bar{s}z_2 \\ -sz_1 + cz_2 \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix}, \quad \sigma^2 = |z_1|^2 + |z_2|^2 > 0, \quad (8.3.18)$$

provided that

$$c = z_1/\sigma, \quad s = z_2/\sigma.$$

A vector $x \in \mathbf{C}^n$ can be transformed into σe_1 , by successive premultiplication with $(n-1)$ unitary plane rotations in planes $(1, 2), (1, 3), \dots, (1, n)$. The rotations may be chosen so that

$$\sigma = x^H x = \|x\|_2^2$$

is real and nonnegative.

Example 8.3.2.

Often one needs to represent an orthogonal rotation $Q \in \mathbf{R}^{3 \times 3}$, $\det(Q) = 1$ as three successive plane rotations or by the angles of these rotations. The classical choice corresponds to a product of three Givens rotations

$$G_{23}(\phi)G_{12}(\theta)G_{23}(\psi)Q = I.$$

The three angles ϕ, θ , and ψ are called the **Euler angles**.

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_3 & s_3 \\ 0 & -s_3 & c_3 \end{pmatrix} \begin{pmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_1 & s_1 \\ 0 & -s_1 & c_1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Here the first Givens rotation $G_{23}(\psi)$ is used to zero the element a_{31} . Next $G_{12}(\theta)$ is used to zero the element a_{21} . Finally, $G_{23}(\phi)$ is used to zero the element a_{32} . The final product is orthogonal and lower triangular and thus must be equal the unit matrix.

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 a_{21}, a_{31}, a_{32} is to be preferred. This corresponds to a product of three Givens rotations

$$G_{23}(\phi)G_{13}(\theta)G_{12}(\psi)Q = I.$$

This yields a continuous representation of Q .

Given's rotation are ubiquitous in matrix algorithms and used to transform a matrix to a more compact form. To illustrate the rotation pattern it is convenient to use a Wilkinson diagram. The diagram below shows the zeroing of the (4,2) element in A by a rotation of rows 2 and 4.

$$\rightarrow \begin{pmatrix} \times & \times & \times & \times \\ \otimes & \times & \times & \times \\ \otimes & \times & \times & \times \\ \otimes & \otimes & \times & \times \\ \otimes & \times & \times & \times \\ \otimes & \times & \times & \times \end{pmatrix}.$$

In a Wilkinson diagram \times stands for a (potential) nonzero element and \otimes for a nonzero element that has been zeroed out. The arrows points to the rows that took part in the last rotation.

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 [544] 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

$$\rho = \begin{cases} s, & \text{if } |s| < |c|; \\ 1/c, & \text{if } |c| \leq |s| \end{cases}.$$

In case $c = 0$ we put $\rho = 1$, a value that cannot appear otherwise. To reconstruct the Givens rotation, if $\rho = 1$, we take $s = 1$, $c = 0$, and

$$\rho = \begin{cases} sF = \rho, & c = \sqrt{1 - s^2}, \quad \text{if } |\rho| < 1; \\ c = 1/\rho, & s = \sqrt{1 - c^2}, \quad \text{if } |\rho| > 1; \end{cases}$$

It is possible to rearrange the Givens rotations so that it uses only two instead of four multiplications per element and no square root. These modified transformations called “fast” Givens transformations, and are described in Golub and Van Loan [277, 1996, Sec. 5.1.13].

8.3.2 Householder QR Factorization

Orthogonality plays a key role in least squares problems; see Theorem 8.2.1. By using methods directly based on orthogonality the squaring of the condition number that results from forming the normal equations can be avoided.

We first show that any matrix $A \in \mathbf{R}^{m \times n}$ ($m \geq n$) can be factored into the product of a *square* orthogonal matrix $Q \in \mathbf{R}^{m \times m}$ and an upper triangular matrix $R \in \mathbf{R}^{m \times n}$ with positive diagonal elements.

Theorem 8.3.1. The Full QR Factorization

Let $A \in \mathbf{R}^{m \times n}$ with $\text{rank}(A) = n$. Then there is a square orthogonal matrix $Q \in \mathbf{R}^{m \times m}$ and an upper triangular matrix R with positive diagonal elements such that

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = (Q_1 \quad Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}. \quad (8.3.19)$$

Proof. The proof is by induction on n . Let A be partitioned in the form $A = (a_1, A_2)$, $a_1 \in \mathbf{R}^m$, where $\rho = \|a_1\|_2 > 0$. Put $y = a_1/\rho$, and let $U = (y, U_1)$ be an orthogonal matrix. Then since $U_1^T a_1 = 0$ the matrix $U^T A$ must have the form

$$U^T A = \begin{pmatrix} \rho & y^T A_2 \\ 0 & U_1^T A_2 \end{pmatrix} = \begin{pmatrix} \rho & r^T \\ 0 & B \end{pmatrix},$$

where $B \in \mathbf{R}^{(m-1) \times (n-1)}$. For $n = 1$, A_2 is empty and the theorem holds with $Q = U$ and $R = \rho$, a scalar. If $n > 1$ then $\text{rank}(B) = n - 1 > 0$, and by the induction hypothesis there is an orthogonal matrix \tilde{Q} such that $\tilde{Q}^T B = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$. (8.3.19) will hold if we define

$$Q = U \begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q} \end{pmatrix}, \quad R = \begin{pmatrix} \rho & r^T \\ 0 & \tilde{R} \end{pmatrix}.$$

□

The proof of this theorem gives a way to compute Q and R , provided we can construct an orthogonal matrix $U = (y, U_1)$ given its first column. Several ways to perform this construction using elementary orthogonal transformations were given in Section .

Note that from the form of the decomposition (8.3.19) it follows that R has the same singular values and right singular vectors as A . A relationship between the Cholesky factorization of $A^T A$ and the QR factorization of A is given next.

Lemma 8.3.2.

Let $A \in \mathbf{R}^{m \times n}$ have rank n . Then if the R factor in the QR factorization of A has positive diagonal elements it equals the Cholesky factor of $A^T A$.

Proof. If $\text{rank}(A) = n$ then the matrix $A^T A$ is nonsingular. Then its Cholesky factor R_C is uniquely determined, provided that R_C is normalized to have a positive diagonal. From (8.3.54) we have $A^T A = R^T Q_1^T Q_1 R = R^T R$, and hence $R = R_C$. Since $Q_1 = AR^{-1}$ the matrix Q_1 is also uniquely determined. □

The QR factorization can be written

$$A = (Q_1 \quad Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1 R. \quad (8.3.20)$$

where the square orthogonal matrix $Q \in \mathbf{R}^{m \times m}$ has been partitioned as

$$Q = (Q_1, Q_2), \quad Q_1 \in \mathbf{R}^{m \times n}, \quad Q_2 \in \mathbf{R}^{m \times (m-n)}.$$

Here $A = Q_1 R$ is called the **thin QR factorization**. From (8.3.20) it follows that the columns of Q_1 and Q_2 form orthonormal bases for the range space of A and its orthogonal complement,

$$\mathcal{R}(A) = \mathcal{R}(Q_1), \quad \mathcal{N}(A^T) = \mathcal{R}(Q_2), \quad (8.3.21)$$

and the corresponding orthogonal projections are

$$P_{\mathcal{R}(A)} = Q_1 Q_1^T, \quad P_{\mathcal{N}(A^T)} = Q_2 Q_2^T. \quad (8.3.22)$$

Note that although the matrix Q_1 in (8.3.20) is uniquely determined, Q_2 can be any orthogonal matrix with range $\mathcal{N}(A^T)$. The matrix Q is *implicitly* defined as a product of Householder or Givens matrices.

The QR factorization of a matrix $A \in \mathbf{R}^{m \times n}$ of rank n can be computed using a sequence of n Householder reflectors. Let $A = (a_1, a_2, \dots, a_n)$, $\sigma_1 = \|a_1\|_2$, and choose $P_1 = I - \beta_1 u_1 u_1^T$, so that

$$P_1 a_1 = P_1 \begin{pmatrix} \alpha_1 \\ \hat{a}_1 \end{pmatrix} = \begin{pmatrix} r_{11} \\ 0 \end{pmatrix}, \quad r_{11} = -\text{sign}(\alpha_1)\sigma_1.$$

By (8.3.4) we achieve this by choosing $\beta_1 = 1 + |\alpha_1|/\sigma_1$,

$$u_1 = \begin{pmatrix} 1 \\ \hat{u}_1 \end{pmatrix}, \quad \hat{u}_1 = \text{sign}(\alpha_1)\hat{a}_1/\rho_1, \quad \rho_1 = \sigma_1\beta_1.$$

P_1 is then applied to the remaining columns a_2, \dots, a_n , giving

$$A^{(2)} = P_1 A = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{n2} & \cdots & \tilde{a}_{nn} \end{pmatrix}.$$

Here the first column has the desired form and, as indicated by the notation, the first row is the final first row in R . In the next step the $(m-1) \times (n-1)$ block in the lower right corner is transformed. All remaining steps, $k = 2 : n$ are similar to the first. Before the k th step we have computed a matrix of the form

$$A^{(k)} = \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & \hat{A}^{(k)} \end{pmatrix}, \quad (8.3.23)$$

where the first $k-1$ rows of $A^{(k)}$ are rows in the final matrix R , and $R_{11}^{(k)}$ is upper triangular. In step k the matrix $a^{(k)}$ is transformed,

$$A^{(k+1)} = P_k A^{(k)}, \quad P_k = \begin{pmatrix} I_k & 0 \\ 0 & \tilde{P}_k \end{pmatrix}. \quad (8.3.24)$$

Here $\tilde{P}_k = I - \beta_k u_k u_k^T$ is chosen to zero the elements below the main diagonal in the first column of the submatrix

$$\hat{A}^{(k)} = (a_k^{(k)}, \dots, a_n^{(k)}) \in \mathbf{R}^{(m-k+1) \times (n-k+1)},$$

i.e. $\tilde{P}_k a_k^{(k)} = r_{kk} e_1$. With $\sigma_k = \|a_k^{(k)}\|_2$, using (8.3.3), we get $r_{kk} = -\text{sign}(a_{kk}^{(k)})\sigma_k$, and

$$\hat{u}_k = \text{sign}(a_k^{(k)})\hat{a}_k^{(k)}/\rho_k, \quad \beta_k = 1 + |a_{kk}|/\sigma_k. \quad (8.3.25)$$

where $\rho_k = \sigma_k/\beta_k$. After n steps we have obtained the QR factorization of A , where

$$R = R_{11}^{(n+1)}, \quad Q = P_1 P_2 \cdots P_n. \quad (8.3.26)$$

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 8.3. Householder QR Factorization.

The algorithm computes the full QR factorization of a matrix $A \in \mathbf{R}^{m \times n}$ of rank n such that $Q = P_1 P_2 \cdots P_n$ where

$$P_k = \text{diag}(I_{k-1}, \tilde{P}_k), \quad \tilde{P}_k = I - \beta_k u_k u_k^T, \quad k = 1 : n, \quad (8.3.27)$$

are Householder matrices.

```

for k = 1 : n
    [u_k, beta_k, r_kk] = house(a_k^{(k)});
    for j = k + 1, ..., n
        gamma_jk = beta_k * u_k^T * a_j^{(k)};
        r_kj = a_{kj}^{(k)} - gamma_jk;
        a_j^{(k+1)} = a_j^{(k)} - gamma_jk * u_k;
    end
end

```

If $m = n$ the last step can be skipped. The vectors \hat{u}_k can overwrite the elements in the strictly lower trapezoidal part of A . Thus, all information associated with the factors Q and R can be overwritten A . The vector $(\beta_1, \dots, \beta_n)$ of length n can be recomputed from

$$\beta_k = \frac{1}{2}(1 + \|\hat{u}_k\|_2^2)^{1/2},$$

and therefore need not be saved. In step k the application of the Householder transformation to the active part of the matrix requires $4(m - k + 1)(n - k)$ flops. Hence, the total flop count is

$$4 \sum_{k=1}^{n-1} (m - k + 1)(n - k) = 4 \sum_{p=1}^{n-1} ((m - n)p + p(p + 1)) = 2(mn^2 - n^3/3).$$

If $m = n$ this is $4n^3/3$ flops.

Theorem 8.3.3.

Let \bar{R} denote the upper triangular matrix R computed by the Householder QR algorithm. Then there exists an exactly orthogonal matrix $\hat{Q} \in \mathbf{R}^{m \times m}$ (not the matrix corresponding to exact computation throughout) such that

$$A + E = \hat{Q} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \quad \|e_j\|_2 \leq \bar{\gamma}_n \|a_j\|_2, \quad j = 1 : n. \quad (8.3.28)$$

where $\bar{\gamma}_n$ is defined in (7.1.79).

As have been stressed before it is usually not advisable to compute the matrix Q in the QR factorization explicitly, even when it is to be used in later computing matrix-vector products. In case that

$$Q = Q^{(0)} = P_1 P_2 \cdots P_n$$

from the Householder algorithm is explicitly required it can be accumulated using the backward recurrence

$$Q^{(n)} = I_m, \quad Q^{(k-1)} = P_k Q^{(k)}, \quad k = n : -1 : 1. \quad (8.3.29)$$

which requires $4(mn(m-n) + n^3/3)$ flops. (Note that this is more efficient than the corresponding forward recurrence. By setting

$$Q^{(n)} = \begin{pmatrix} I_n \\ 0 \end{pmatrix}, \quad \text{or} \quad Q^{(n)} = \begin{pmatrix} 0 \\ I_{m-n} \end{pmatrix},$$

For $m = n$ this becomes $4n^3/3$ flops. The matrices Q_1 and Q_2 , whose columns span the range space and nullspace of A , respectively, can be similarly computed in $2(mn^2 - n^3/3)$ and $2m^2n - 3mn^2 + n^3$ flops, respectively; see Problem 8.3.7 (b).

Example 8.3.3.

In structured problems the greater flexibility of Givens rotations is an advantage. An important example is the QR factorization of a Hessenberg matrix

$$H_n = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,n-1} & h_{1,n} \\ h_{21} & h_{22} & \cdots & h_{2,n-1} & h_{2,n} \\ h_{32} & \cdots & \vdots & & \vdots \\ \ddots & h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & & h_{n,n} \end{pmatrix} \in \mathbf{R}^{n \times n},$$

which can be computed efficiently using Givens rotations. We illustrate below the first two steps of the algorithm for $n = 5$ in a Wilkinson diagram. In the first step 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.

$$\rightarrow \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} \rightarrow \begin{pmatrix} \times & \times & \times & \times & \times \\ \otimes & \times & \times & \times & \times \\ \otimes & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix}.$$

The arrows points to the rows that took part in the last rotation. After $n - 1$ steps all subdiagonal elements have been zeroed out and we have obtained the QR factorization

$$Q^T H = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad Q^T = G_{n-1,n} \cdots G_{23} G_{12}. \quad (8.3.30)$$

The first step in the QR factorization takes $6n$ flops and the total work of this QR factorization is only about $3n$ flops. \square

It is often advisable to use **column pivoting** in Householder QR factorization. The standard procedure is choose a pivot column that maximizes the diagonal element r_{kk} in the k th step. Assume that after $k - 1$ steps we have computed the partial QR factorization

$$A^{(k)} = (P_{k-1} \cdots P_1) A (\Pi_1 \cdots \Pi_{k-1}) = \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & \tilde{A}^{(k)} \end{pmatrix}, \quad (8.3.31)$$

Then the pivot column in the next step is chosen as a column of largest norm in the submatrix

$$\tilde{A}^{(k)} = (\tilde{a}_k^{(k)}, \dots, \tilde{a}_n^{(k)}),$$

If $\tilde{A}^{(k)} = 0$ the algorithm terminates. Otherwise, let

$$s_j^{(k)} = \|\tilde{a}_j^{(k)}\|_2^2, \quad j = k : n. \quad (8.3.32)$$

and interchange columns p and k , where p is the smallest index such that $s_p^{(k)} = \max_{j=k}^n s_j^{(k)}$. This pivoting strategy ensures that the computed triangular factor has the property stated in Theorem 8.3.4. Since the column lengths are invariant under orthogonal transformations the quantities $s_j^{(k)}$ can be updated

$$s_j^{(k+1)} = s_j^{(k)} - r_{jk}^2, \quad j = k + 1 : n. \quad (8.3.33)$$

We remark that the pivoting rule (8.3.56) is equivalent to maximizing the diagonal element r_{kk} , $k = 1 : r$. Therefore, (in exact arithmetic it computes the Cholesky factor that corresponds to using the pivoting (7.3.11) in the Cholesky factorization.

If the column norms in $\tilde{a}^{(k)}$ were recomputed at each stage, then column pivoting would increase the operation count by 50%. Instead the norms of the columns of A can be computed initially, and recursively updated as the factorization proceeds. This reduces the overhead of column pivoting to $O(mn)$ operations. This pivoting strategy can also be implemented in the Cholesky and modified Gram–Schmidt algorithms.

Since column norms are preserved by orthogonal transformations the factor R has the following important property:

Theorem 8.3.4.

Suppose that R is computed by QR factorization with column pivoting. Then the elements in R satisfy the inequalities

$$r_{kk}^2 \geq \sum_{i=k}^j r_{ij}^2, \quad j = k+1, \dots, n. \quad (8.3.34)$$

In particular, $|r_{kk}| \geq |r_{kj}|$, $j > k$, and the diagonal elements form a non-increasing sequence

$$|r_{11}| \geq |r_{22}| \geq \dots \geq |r_{nn}|. \quad (8.3.35)$$

For any QR factorization it holds that

$$\sigma_1 = \max_{\|x\|=1} \|Rx\|_2 \geq \|Re_1\|_2 = |r_{11}|,$$

and thus $|r_{11}|$ is a lower bound for the largest singular value σ_1 of A . and singular values $1/\sigma_k(A)$. Since R and R^T have the same singular values, we also have

$$\sigma_n = \min_{\|x\|=1} \|R^Tx\|_2 \leq \|R^Te_n\|_2 = |r_{nn}|,$$

which gives an upper bound for σ_n . For a triangular matrix satisfying (8.3.34) we also have the upper bound

$$\sigma_1(R) = \|R\|_2 \leq \|R\|_F = \left(\sum_{i \leq j} r_{ij}^2 \right)^{1/2} \leq \sqrt{n}|r_{11}|.$$

$\sigma_1 \leq n^{1/2}|r_{11}|$. Using the interlacing property of singular values (Theorem 8.1.17), a similar argument gives the upper bounds

$$\sigma_k(R) \leq (n-k+1)^{1/2}|r_{k,k}|, \quad 1 \leq k \leq n. \quad (8.3.36)$$

If after k steps in the pivoted QR factorization it holds that

$$|r_{k,k}| \leq (n-k+1)^{-1/2}\delta,$$

then $\sigma_k(A) = \sigma_k(R) \leq \delta$, and A has numerical rank at most equal to $k-1$, and we should terminate the algorithm. Unfortunately, the converse is not true, i.e., the

rank is not always revealed by a small element $|r_{kk}|$, $k \leq n$. Let R be an upper triangular matrix whose elements satisfy (8.3.34). The best known *lower* bound for the smallest singular value is

$$\sigma_n \geq 3|r_{nn}|/\sqrt{4^n + 6n - 1} \geq 2^{1-n}|r_{nn}|. \quad (8.3.37)$$

(For a proof of this see Lawson and Hanson [399, Ch. 6].)

The lower bound in (8.3.37) can almost be attained as shown in the example below due to Kahan. Then the pivoted QR factorization may not reveal the rank of A .

Example 8.3.4. Consider the upper triangular matrix

$$R_n = \text{diag}(1, s, s^2, \dots, s^{n-1}) \begin{pmatrix} 1 & -c & -c & \dots & -c \\ & 1 & -c & \dots & -c \\ & & 1 & & \vdots \\ & & & \ddots & -c \\ & & & & 1 \end{pmatrix}, \quad s^2 + c^2 = 1.$$

It can be verified that the elements in R_n satisfies the inequalities in (8.3.37), and that R_n is invariant under QR factorization with column pivoting. For $n = 100$, $c = 0.2$ the last diagonal element of R is $r_{nn} = s^{n-1} = 0.820$. This can be compared with the smallest singular value which is $\sigma_n = 0.368 \cdot 10^{-8}$. If the columns are reordered as $(n, 1, 2, \dots, n-1)$ and the rank is revealed from the pivoted QR factorization!

The above example has inspired research into alternative column permutation strategies. We return to this problem in Section 8.4.3, where algorithms for the subset selection problem are given.

8.3.3 Least Squares Problems by QR Factorization

We now show how to use the QR factorization to solve the linear least squares problem (8.1.1).

Theorem 8.3.5.

Let $A \in \mathbf{R}^{m \times n}$, $\text{rank}(A) = n$, have the full QR factorization

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad Q = (Q_1 \quad Q_2),$$

Then the unique solution x to $\min_x \|Ax - b\|_2$ and the corresponding residual vector $r = b - Ax$ are given by

$$d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = Q^T b, \quad x = R^{-1} d_1, \quad r = Q \begin{pmatrix} 0 \\ d_2 \end{pmatrix}, \quad (8.3.38)$$

and hence $\|r\|_2 = \|c_2\|_2$.

Proof. Since Q is orthogonal we have

$$\|Ax - b\|_2^2 = \|Q^T(Ax - b)\|_2^2 = \left\| \begin{pmatrix} Rx \\ 0 \end{pmatrix} - \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \right\|_2^2 = \|Rx - c_1\|_2^2 + \|d_2\|_2^2.$$

Obviously the right hand side is minimized for $x = R^{-1}c_1$. With c defined by (8.3.38) we have using the orthogonality of Q that

$$b = QQ^Tb = Q_1d_1 + Q_2d_2 = Ax + r.$$

Since $Q_1c_1 = Q_1Rz = Ax$ it follows that $r = Q_2c_2$. \square

By Theorem 8.3.5, when R and the Householder reflections P_1, P_2, \dots, P_n have been computed by Algorithm 8.3.2 the least squares solution x and residual r can be computed as follows:

$$d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = P_n \cdots P_2 P_1 b, \quad (8.3.39)$$

$$Rx = d_1, \quad r = P_1 \cdots P_{n-1} P_n \begin{pmatrix} 0 \\ d_2 \end{pmatrix}, \quad (8.3.40)$$

and $\|r\|_2 = \|d_2\|_2$. Note that the matrix Q should not be explicitly formed.

The operation count for the Householder QR factorization is $2n^2(m - n/3)$ flops. For $m = n$ both methods require the same work but for $m \gg n$ the Householder QR method is twice as expensive. To compute Q^Tb and solve $Rx = d_1$ requires $4n(m - n/4)$ flops. In some cases only $\|r\|_2$ is required. If r is needed this requires another $4n(m - n/2)$ flops. 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.

The Householder algorithm is backward stable for computing both the solution x and the residual $r = b - Ax$. This means that the computed residual \bar{r} satisfies

$$(A + E)^T \bar{r} = 0, \quad \|E\|_2 \leq cu\|A\|_2, \quad (8.3.41)$$

for some constant $c = c(m, n)$. Hence, $A^T \bar{r} = -E^T \bar{r}$, and

$$\|A^T \bar{r}\|_2 \leq cu\|\bar{r}\|_2\|A\|_2. \quad (8.3.42)$$

Note that this is much better result than if the residual is computed as

$$\tilde{r} = fl(b - fl(Ax)) = fl \left(\begin{pmatrix} b & A \end{pmatrix} \begin{pmatrix} 1 \\ -x \end{pmatrix} \right),$$

even when x is the *exact* least squares solution. Using (2.3.13) and $A^T r = 0$

$$|A^T \tilde{r}| < \gamma_{n+1} |A^T|(|b| + |A||x|).$$

we obtain the normwise bound

$$\|A^T \bar{r}\|_2 \leq n^{1/2} \gamma_{n+1} \|A\|_2 (\|b\|_2 + n^{1/2} \|A\|_2 \|x\|_2),$$

This is much weaker than (8.3.42), in particular when $\|\bar{r}\|_2 \ll \|b\|_2$.

The method in Theorem 8.3.5 generalizes easily to a method for solving the augmented system (8.1.10). Recall that the solution to the augmented system gives the solution to the two optimizations problems (8.1.11)–(8.1.12).

Theorem 8.3.6.

Let the full QR factorization of $A \in \mathbf{R}^{m \times n}$ with $\text{rank}(A) = n$, be given by $A = (Q_1 \ Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}$. Then the solution to the augmented system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix},$$

can be computed as

$$z = R^{-T}c, \quad \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = Q^Tb, \quad (8.3.43)$$

$$x = R^{-1}(d_1 - z) \quad y = Q \begin{pmatrix} z \\ d_2 \end{pmatrix}, \quad (8.3.44)$$

Further, we have

$$\min \|b - y\|_2 = \min \|Q^Tb - Q^Ty\|_2 = \|d_1 - z\|_2.$$

Proof. Using the QR factorization the subsystems $y + Ax = b$ and $A^T y = c$ of the augmented system can be written

$$y + Q \begin{pmatrix} R \\ 0 \end{pmatrix} x = b, \quad (R^T \ 0) Q^T y = c.$$

Multiplying the first system by Q^T and the second by R^{-T} gives

$$Q^T y + \begin{pmatrix} R \\ 0 \end{pmatrix} x = Q^T b, \quad (I_n \ 0) Q^T y = R^{-T} c, \quad \min \|y\|_2 = \|z\|_2.$$

Using the second equation to eliminate the first n components of $Q^T y$ in the first equation, we can then solve for x . The last $m - n$ components of $Q^T y$ are obtained from the last $m - n$ equations in the first block. \square

Note that either x or y can be computed without the other. Thus the algorithm (8.3.43)–(8.3.44) can be used to solve either the linear least squares problem (8.1.11) or the conditional least squares problem (8.1.12).

In the special case that $b = 0$ the algorithm simplifies to

$$z = R^{-T}c, \quad y = Q \begin{pmatrix} z \\ 0 \end{pmatrix}, \quad (8.3.45)$$

and solves the minimum norm problem $\min \|y\|_2$ subject to $A^T y = c$.

The Householder QR algorithm, and the resulting method for solving the least squares problem are backwards stable, both for x and r , and the following result holds.

Theorem 8.3.7.

Let \bar{R} denote the computed R . Then there exists an exactly orthogonal matrix $\tilde{Q} \in \mathbf{R}^{m \times m}$ (not the matrix corresponding to exact computation throughout) such that

$$A + \Delta A = \tilde{Q} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \quad \|\Delta A\|_F \leq c\gamma_{mn}\|A\|_F, \quad (8.3.46)$$

where c is a small constant. Further, the computed solution \bar{x} is the exact solution of a slightly perturbed least squares problem

$$\min_x \|(A + \delta A)x - (b + \delta b)\|_2,$$

where the perturbation can be bounded in norm by

$$\|\delta A\|_F \leq c\gamma_{mn}\|A\|_F, \quad \|\delta b\|_2 \leq c\gamma_{mn}\|b\|_2, \quad (8.3.47)$$

Proof. See Higham [328, Theorem 19.5]. \square

The column pivoting strategy suggested earlier may not be appropriate, when we are given one vector b , which is to be approximated by a linear combination of as few columns of the matrix A as possible. This occurs, e.g., in regression analysis, where each column in A corresponds to one factor. Even when the given b is parallel to one column in A it could happen that we had to complete the full QR factorization of A before this fact was discovered. Instead we would like at each stage to select the column that maximally reduces the sum of squares of the residuals.

8.3.4 Gram–Schmidt QR Factorization

We start by considering the case of orthogonalizing *two* given linearly independent vectors a_1 and a_2 in \mathbf{R}^n . We set

$$q_1 = a_1/r_{11}, \quad r_{11} = \|a_1\|_2.$$

and seek a unit vector $q_2 \in \text{span}[a_1, a_2]$ such that $q_1^T q_2 = 0$. By subtracting from a_2 its orthogonal projection onto q_1 , we get

$$\hat{q}_2 = (I - q_1 q_1^T) a_2 = a_2 - r_{12} q_1, \quad r_{12} = q_1^T a_2, \quad (8.3.48)$$

We have $\hat{q}_2 \neq 0$, since otherwise a_2 would not be linearly independent of a_1 . It is easily verified that $q_1^T \hat{q}_2 = q_1^T a_2 - r_{12} q_1^T q_1 = 0$. Thus, we can set

$$q_2 = \hat{q}_2/r_{22}, \quad r_{22} = \|\hat{q}_2\|_2.$$

Since q_1 and q_2 both are linear combinations of a_1 and a_2 , they span the same subspace of \mathbf{R}^n . Further, we have the relation

$$(a_1 \ a_2) = (q_1 \ q_2) \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}.$$

Clearly, the process works also for complex vectors $a_k \in \mathbf{C}^m$,

The above algorithm can be extended to orthogonalizing any sequence of linearly independent vectors a_1, \dots, a_n in \mathbf{R}^m ($m \geq n$). This process, known as **Gram–Schmidt orthogonalization**,³¹ ³² uses elementary orthogonal projections. After step k , $k = 1 : n$, orthonormal vectors q_1, \dots, q_k have been computed such that

$$\text{span}[q_1, \dots, q_k] = \text{span}[a_1, \dots, a_k] \quad (8.3.49)$$

We give two variants of the algorithm. Although these only differ in which order the operations are carried out, they have greatly different numerical stability.

Classical Gram–Schmidt (CGS):

For $k = 1 : n$ orthogonalize a_k against q_1, \dots, q_{k-1} :

$$\hat{q}_k = a_k - \sum_{i=1}^{k-1} r_{ik} q_i, \quad r_{ik} = q_i^T a_k, \quad i = 1 : k-1, \quad (8.3.50)$$

and normalize

$$r_{kk} = \|\hat{q}_k\|_2, \quad q_k = \hat{q}_k / r_{kk}. \quad (8.3.51)$$

The unnormalized vector \tilde{q}_k is just the orthogonal projection a_k onto the complement of $\text{span}[a_1, \dots, a_{k-1}]$.

Modified Gram–Schmidt (MGS)

Set $a_j^{(1)} = a_j$, $j = 1 : n$. For $k = 1 : n$, normalize

$$r_{kk} = \|a_k^{(k)}\|_2, \quad q_k = a_k^{(k)} / r_{kk}.$$

Orthogonalize $a_j^{(k)}$, $j = k+1 : n$, against q_k :

$$a_j^{(k+1)} = a_j^{(k)} - r_{kj} q_k, \quad r_{kj} = q_k^T a_j^{(k)}, \quad j = k+1 : n. \quad (8.3.52)$$

At the beginning of step k in the MGS algorithm we have computed

$$(q_1, \dots, q_{k-1}, a_k^{(k)}, \dots, a_n^{(k)}),$$

where $a_k^{(k)}, \dots, a_n^{(k)}$ are orthogonal to q_1, \dots, q_{k-1} .

In CGS the orthogonalization of a_k can be written

$$\hat{q}_k = (I - Q_{k-1} Q_{k-1}^T) a_k, \quad Q_{k-1} = (q_1, \dots, q_{k-1}).$$

What makes the difference in numerical stability is that in MGS *the projections $r_{ik} q_i$ are subtracted from a_k as soon as they are computed*, which corresponds to computing

$$\hat{q}_k = (I - q_{k-1} q_{k-1}^T) \cdots (I - q_1 q_1^T) a_k. \quad (8.3.53)$$

³¹Jørgen Pedersen Gram (1850–1916), Danish mathematician, Gram worked for Hafnia Insurance Company and made contributions to probability and numerical analysis.

³²Erhard Schmidt (1876–1959), German mathematician obtained his Ph.D. from the University of Göttingen in 1905 under Hilbert's supervision. In 1920 Schmidt was appointed to the post of Director of the Institute of Applied Mathematics at the University of Berlin in 1917.

Clearly, for $n = 2$ MGS and CGS are identical. For $k > 2$ these two expressions are identical only if the q_1, \dots, q_{k-1} are orthogonal. As described in CGS the matrix R is determined row by row, in MGS column by column. The rowwise order used in MGS is convenient when column pivoting is to be performed. However, a columnwise MGS version is also possible for applications where the columns are a_k are generated one at a time.

In matrix terms the Gram–Schmidt process uses elementary column operations to transform the matrix A into an orthogonal matrix Q to give R and Q (explicitly) in the thin QR factorization. This can be contrasted with the Householder QR factorization, where the matrix A is premultiplied by a sequence of elementary orthogonal transformations to produce R and Q (in product form) in the full QR factorization.

Theorem 8.3.8.

Let the matrix $A = (a_1, a_2, \dots, a_n) \in \mathbf{C}^{m \times n}$ have linearly independent columns. Then the Gram–Schmidt algorithm computes a matrix $Q_1 \in \mathbf{R}^{m \times n}$ with orthonormal columns $Q_1^H Q_1 = I_n$ and an upper triangular matrix $R \in \mathbf{C}^{n \times n}$ with real positive diagonal elements such that

$$A = (q_1, q_2, \dots, q_n) \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{pmatrix} \equiv Q_1 R. \quad (8.3.54)$$

Proof. Note that $\hat{q}_k \neq 0$, since otherwise a_k is a linear combination of the vectors a_1, \dots, a_{k-1} , which contradicts the assumption. Combining (8.3.50) and (8.3.51) 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 (8.3.54). Since the vectors q_k are mutually orthogonal by construction, the theorem follows. \square

Although mathematically equivalent, MGS has greatly superior numerical properties compared to CGS, and is therefore usually to be preferred. (We say that two formulas or algorithms are mathematically equivalent if they produce the same result in exact arithmetic.) In fact, because it is more convenient, MGS had almost always been used in practice long before its superior numerical stability was appreciated.

Algorithm 8.4. Modified Gram–Schmidt.

Given $A \in \mathbf{R}^{m \times n}$ with $\text{rank}(A) = n$ the following algorithm computes the factorization $A = Q_1 R$:

```

function [Q,R] = mgs(A);
[m,n] = size(A);
Q = A;
for k = 1:n
    R(k,k) = norm(Q(:,k));
    Q(:,k) = Q(:,k)/R(k,k);
    for j = k+1:n
        R(k,j) = Q(:,k)'*Q(:,j);
        Q(:,j) = Q(:,j) - Q(:,k)*R(k,j);
    end
end

```

The operations in Algorithm 8.3.4 can be sequenced so that the elements in R are computed in a columnwise fashion. However, the rowwise version given above is more suitable if column pivoting is to be performed. Gram–Schmidt orthogonalization requires approximately $2mn^2$ flops, which is slightly more than the $2(mn^2 - n^3/3)$ needed for Householder QR factorization.

It can be shown by an elementary error analysis that if \bar{Q}_1 and \bar{R} are computed by MGS, then it holds that

$$A + E = \bar{Q}_1 \bar{R}, \quad \|E\|_2 \leq c_0 u \|A\|_2. \quad (8.3.55)$$

Thus, the product $\bar{Q}_1 \bar{R}$ accurately represents A .

Assume that the columns a_1, \dots, a_{k-1} are linearly independent but a_k is a linear combination of a_1, \dots, a_{k-1} . Then $a_k^{(k-1)} = 0$ and the Gram–Schmidt process breaks down. But if $\text{rank}(A) > k$ there must be a vector a_j , $j > k$, which is linearly independent of a_1, \dots, a_{k-1} . We can then interchange columns k and j and proceed until all remaining columns are linearly dependent on the computed q vectors.

This suggest that we augment the MGS method with **column pivoting** as follows. Compute

$$\|a_j^{(k)}\|_2, \quad j = k : n, \quad (8.3.56)$$

and let the maximum be $|a_j^{(p)}\|_2$. (If there are more than one maximum, choose the first.) If this maximum is zero, then all columns a_k, \dots, a_n are linearly dependent and the reduction is complete. Otherwise interchange columns p and k and proceed. After the k th step we can cheaply update the column norms using

$$\|a_j^{(k+1)}\|_2 = \|a_j^{(k)}\|_2 - r_{kj}^2, \quad j = k+1 : n.$$

With column pivoting MGS can be used also when the matrix A has linearly dependent columns. If $\text{rank}(A) = r$ it will compute a factorization of the form

$$A\Pi = QR, \quad Q \in \mathbf{R}^{m \times r}, \quad R = (R_{11} \quad R_{12}) \in \mathbf{R}^{r \times n}, \quad (8.3.57)$$

where Π is a permutation matrix, $Q^T Q = I$, and R_{11} is upper triangular and nonsingular. Indeed, MGS with column pivoting is a good algorithm for determining the rank of a given matrix A .

8.3.5 Loss of Orthogonality in GS and MGS

In some applications it may be essential that the computed columns of Q are orthogonal to working accuracy. For example, this is the case when MGS is used in algorithms for computing eigenvectors to symmetric matrices.

We will now use the standard model for floating point computation, and the basic results in Section 2.3.2 to analyze the rounding errors when the Gram–Schmidt process is applied to orthogonalize two linearly independent vectors a_1 and a_2 in \mathbf{R}^n . For the *computed* scalar product $\bar{r}_{12} = fl(q_1^T a_2)$ we get

$$|\bar{r}_{12} - r_{12}| < \gamma_m \|a_2\|_2, \quad \gamma_m = \frac{mu}{1 - mu/2},$$

where u is the unit roundoff. Using $|r_{12}| \leq \|a_2\|_2$ we obtain for $\bar{q}_2 = fl(a_2 - fl(\bar{r}_{12} q_1))$

$$\|\bar{q}_2 - \hat{q}_2\|_2 < \gamma_{m+2} \|a_2\|_2.$$

Since $q_1^T \hat{q}_2 = 0$, it follows that $|q_1^T \bar{q}_2| < \gamma_{m+2} \|a_2\|_2$ and the loss of orthogonality

$$\frac{|q_1^T \bar{q}_2|}{\|\bar{q}_2\|_2} \approx \frac{|q_1^T \bar{q}_2|}{\|\hat{q}_2\|_2} < \gamma_{m+2} \frac{\|a_2\|_2}{\|\tilde{q}_2\|_2} = \frac{\gamma_{m+2}}{\sin \phi(q_1, a_2)}, \quad (8.3.58)$$

is proportional to $\phi(q_1, a_2)$, the angle between q_1 and a_2 .

We conclude that in the Gram–Schmidt process a severe loss of orthogonality may occur whenever cancellation takes place in subtracting the orthogonal projection on q_i from $a_k^{(i)}$, that is when

$$a_j^{(k+1)} = (I - q_k q_k^T) a_j^{(k)}, \quad \|a_k^{(i+1)}\|_2 \ll \alpha \|a_k^{(i)}\|_2. \quad (8.3.59)$$

Example 8.3.5.

For the extremely ill-conditioned matrix

$$A = (a_1, a_2) = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix},$$

in Example 7.5.4 the Gram–Schmidt algorithm in IEEE double precision gives

$$q_1 = \begin{pmatrix} 0.98640009002732 \\ 0.16436198585466 \end{pmatrix},$$

$r_{12} = q_1^T a_2 = 0.87672336001729$, Subtracting the orthogonal projection onto q_1 we get

$$\hat{q}_2 = a_2 - r_{12} q_1 = \begin{pmatrix} -0.12501091273265 \\ 0.75023914025696 \end{pmatrix} 10^{-8}.$$

Normalizing this vector gives

$$q_2 = \begin{pmatrix} -0.16436196071471 \\ 0.98640009421635 \end{pmatrix},$$

Table 8.3.1. Loss of orthogonality and 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

and

$$R = \begin{pmatrix} 1.31478090189963 & 0.87672336001729 \\ 0 & 0.00000000760583 \end{pmatrix}.$$

Severe cancellation has taken place when computing \hat{q}_2 . This leads to a serious loss of orthogonality between q_1 and q_2 :

$$q_1^T q_2 = 2.5486557 \cdot 10^{-8},$$

The loss of orthogonality is roughly equal to a factor $\kappa(A) \approx 10^{-8}$.

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 8.3.9.

Let \bar{Q} and \bar{R} denote the factors computed by the MGS algorithm. Then there are constants $c_i = c_i(m, n)$, $i = 1, 2$, such that if $c_1 \kappa(A) u < 1$,

$$\|I - \bar{Q}_1^T \bar{Q}_1\|_2 \leq \frac{c_1}{1 - c_1 u \kappa_2(A)} u \kappa_2(A). \quad (8.3.60)$$

Proof. See Björck [57]. \square

The computed vectors q_k from CGS depart from orthogonality much more rapidly. After a small modification of the algorithm an upper bound for the loss of orthogonality proportional to κ^2 has been proved. Even computing $Q_1 = AR^{-1}$, where R is determined by Cholesky factorization often gives better orthogonality than CGS; see the example below.

Example 8.3.6.

A matrix $A \in \mathbf{R}^{50 \times 10}$ was generated by computing

$$A = UDV^T, \quad D = \text{diag}(1, 10^{-1}, \dots, 10^{-9})$$

with U and V orthonormal matrices. Hence, A has singular values $\sigma_i = 10^{-i+1}$, $i = 1 : 10$, and $\kappa(A) = 10^9$. Table 8.3.1 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 $\|I_k - Q_k^T Q_k\|_2$. Note that for MGS the loss of orthogonality is more gradual than for CGS and proportional to $\kappa(A_k)$,

In several applications it is important that the computed \bar{Q}_1 is orthogonal to working accuracy. We shall call this the **orthogonal basis problem**. To achieve this we can **reorthogonalize** the computed vectors in the Gram–Schmidt algorithm. This must be carried out whenever substantial cancellation occurs when subtracting a projection.

Assume that we are given a matrix

$$Q_1 = (q_1, \dots, q_{k-1}), \quad \|q_j\|_2 = 1, \quad j = 1 : k-1.$$

Adding the new vector a_k , we want to compute a vector \hat{q}_k such that

$$\hat{q}_k \perp \text{span}(Q_1), \quad \hat{q}_k \in \text{span}(Q_1, a_k).$$

If $(Q_1 a_k)$ has full numerical rank and Q_1 is accurately orthogonal, then it has been proved that *one reorthogonalization will always suffice*. For $k = 2$ this result is due to Kahan and Parlett; see Parlett [478, Sec. 6.9]. As an example, reorthogonalizing the computed vector $a_2^{(2)}$ in Example 8.3.5 against q_1 gives

$$q_1^T q_2 = 2.5486557 \cdot 10^{-8}, \quad \tilde{q}_2 = \begin{pmatrix} -0.16436198585466 \\ 0.98640009002732 \end{pmatrix}.$$

The vector \tilde{q}_2 is exactly orthogonal to q_1 .

If A has full numerical rank and you reorthogonalize as you go along, then for both CGS and MGS one reorthogonalization suffices to produce vectors that are orthonormal to machine precision. Hence, reorthogonalization will double the cost. An MGS algorithm with reorthogonalization is given below. Here a columnwise version is used and the reorthogonalization against previously computed vectors is performed in backward order.

Algorithm 8.5. *MGS with reorthogonalization.*

```
function [Q,R] = mgs2(A);
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
    for i = [1:k-1,k-1:-1:1]
```

```

V = Q(:,i)'*Q(:,k);
Q(:,k) = Q(:,k) - V*Q(:,i);
R(i,k) = R(i,k) + V;
end
R(k,k) = norm(Q(:,k));
Q(:,k) = Q(:,k)/R(k,k);
end

```

In **selective reorthogonalization** a test is performed at each step to see whether or not it is necessary to reorthogonalize. Usually reorthogonalization is carried out whenever

$$\|\bar{q}_k\|_2 < \alpha \|a_k\|_2. \quad (8.3.61)$$

for some chosen tolerance α . 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. Typically α is chosen in the range $0.1 \leq \alpha \leq 1/\sqrt{2}$. In practice, the value $\alpha = 1/\sqrt{2}$ is often used. In selective reorthogonalization the columns of Q_1 may not be orthogonal to working accuracy. Then it might be necessary to reorthogonalize more than once.

8.3.6 MGS as a Householder Method

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 augmented with a square matrix of zero elements on top.³³ This is not true only in theory, but in the presence of rounding errors as well. We first look at the theoretical result.

Let $A \in \mathbf{R}^{m \times n}$ have rank n and consider the two QR factorizations

$$A = (Q_1 \quad Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad \tilde{A} = \begin{pmatrix} O \\ A \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \quad (8.3.62)$$

$Q \in \mathbf{R}^{m \times m}$ and $P \in \mathbf{R}^{(n+m) \times (n+m)}$ are orthogonal matrices. If the upper triangular matrices R and \tilde{R} are both chosen to have positive diagonal elements then by uniqueness $R = \tilde{R}$. 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 applied to the augmented matrix. To see this, recall that the Householder transformation $Px = e_1\rho$ uses

$$P = I - 2vv^T/\|v\|_2^2, \quad v = x - \rho e_1, \quad \rho = \pm\|x\|_2.$$

If the second factorization in (8.3.62) is obtained using Householder transformations, then

$$P^T = P_n \cdots P_2 P_1, \quad P_k = I - 2\hat{v}_k \hat{v}_k^T / \|\hat{v}_k\|_2^2, \quad k = 1 : n, \quad (8.3.63)$$

³³This observation was made by Charles Sheffield, apparently when comparing Fortran code for Householder and MGS QR factorization.

where the vectors \hat{v}_k are described below. Now, from MGS applied to $A^{(1)} = A$, $r_{11} = \|a_1^{(1)}\|_2$, and $a_1^{(1)} = q'_1 = q_1 r_{11}$. The first Householder transformation applied to the augmented matrix

$$\begin{aligned}\tilde{A}^{(1)} &\equiv \begin{pmatrix} O_n \\ A^{(1)} \end{pmatrix}, & \tilde{a}_1^{(1)} &= \begin{pmatrix} 0 \\ a_1^{(1)} \end{pmatrix}, \\ \hat{v}_1^{(1)} &\equiv \begin{pmatrix} -e_1 r_{11} \\ q'_1 \end{pmatrix} = r_{11} v_1, & v_1 &= \begin{pmatrix} -e_1 \\ q_1 \end{pmatrix}.\end{aligned}$$

But $\|v_1\|_2^2 = 2$, giving

$$P_1 = I - 2\hat{v}_1\hat{v}_1^T/\|\hat{v}_1\|_2^2 = I - 2v_1v_1^T/\|v_1\|_2^2 = I - v_1v_1^T,$$

and

$$P_1 \tilde{a}_j^{(1)} = \tilde{a}_j^{(1)} - v_1 v_1^T \tilde{a}_j^{(1)} = \begin{pmatrix} 0 \\ a_j^{(1)} \end{pmatrix} - \begin{pmatrix} -e_1 \\ q_1 \end{pmatrix} q_1^T a_j^{(1)} = \begin{pmatrix} e_1 r_{1j} \\ a_j^{(2)} \end{pmatrix},$$

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 $a_j^{(2)}$ are the same. We see that the next Householder transformation produces the second row of R and $a_j^{(3)}$, $j = 3 : n$, just as in MGS. Carrying on this way we can conclude that this Householder QR is numerically equivalent to MGS applied to A , and that every P_k is effectively defined by Q_1 , since

$$P_k = I - v_k v_k^T, \quad v_k = \begin{pmatrix} -e_k \\ q_k \end{pmatrix}, \quad k = 1 : n. \quad (8.3.64)$$

From the numerical equivalence it follows that the backward error analysis for the Householder QR factorization of the augmented matrix can also be applied to the modified Gram–Schmidt algorithm on A . Let $\bar{Q}_1 = (\bar{q}_1, \dots, \bar{q}_n)$ be the matrix of vectors computed by MGS, and for $k = 1, \dots, n$ define

$$\begin{aligned}\bar{v}_k &= \begin{pmatrix} -e_k \\ \bar{q}_k \end{pmatrix}, & \bar{P}_k &= I - \bar{v}_k \bar{v}_k^T, & \bar{P} &= \bar{P}_1 \bar{P}_2 \dots \bar{P}_n, \\ \tilde{q}_k &= \bar{q}_k / \|\bar{q}_k\|_2, & \tilde{v}_k &= \begin{pmatrix} -e_k \\ \tilde{q}_k \end{pmatrix}, & \tilde{P}_k &= I - \tilde{v}_k \tilde{v}_k^T, & \tilde{P} &= \tilde{P}_1 \tilde{P}_2 \dots \tilde{P}_n.\end{aligned} \quad (8.3.65)$$

Then \bar{P}_k is the computed version of the Householder matrix applied in the k th step of the Householder QR factorization of the augmented matrix and \tilde{P}_k is its orthonormal equivalent, so that $\tilde{P}_k^T \tilde{P}_k = I$. From the error analysis for Householder QR (see Theorem 8.3.7) it follows that for \bar{R} computed by MGS,

$$\begin{pmatrix} E_1 \\ A + E_2 \end{pmatrix} = \tilde{P} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \quad \bar{P} = \tilde{P} + E',$$

where

$$\|E_i\|_2 \leq c_i u \|A\|_2, \quad i = 1, 2; \quad \|E'\|_2 \leq c_3 u, \quad (8.3.66)$$

and 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 [65]) that there exist an *exactly orthonormal matrix* \hat{Q}_1 and E such that

$$A + E = \hat{Q}_1 \bar{R}, \quad \hat{Q}_1^T \hat{Q}_1 = I, \quad \|E\|_2 \leq c u \|A\|_2. \quad (8.3.67)$$

If $\bar{\sigma}_1 \geq \dots \geq \bar{\sigma}_n$ are the singular values of \bar{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 (8.3.67) shows that \bar{R} computed by MGS is comparable in accuracy to the upper triangular matrix from the Householder QR factorization applied to A .

Using the relationship algorithms for least squares problems using the MGS factorization can be developed that give results comparable in accuracy with those obtained by the corresponding Householder QR algorithm. We first derive the MGS algorithm for solving the least squares problems. Clearly, the two problems

$$\min_x \|Ax - b\|_2 \quad \text{and} \quad \min_x \left\| \begin{pmatrix} 0 \\ A \end{pmatrix} x - \begin{pmatrix} 0 \\ b \end{pmatrix} \right\|_2.$$

have the same solution. As above the q_k from MGS corresponds to the Householder transformation P_k , $1 : n$, in (8.3.64). To compute d and h in

$$\begin{pmatrix} d \\ h \end{pmatrix} = P_n \dots P_1 \begin{pmatrix} 0 \\ b \end{pmatrix},$$

define

$$\begin{pmatrix} d_1 \\ h_1 \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}, \quad \begin{pmatrix} d_{k+1} \\ h_{k+1} \end{pmatrix} = P_k \dots P_1 \begin{pmatrix} 0 \\ b \end{pmatrix} = P_k \begin{pmatrix} d_k \\ h_k \end{pmatrix}.$$

Now using induction we see d_k has all but its first $k - 1$ elements zero, and

$$\begin{pmatrix} d_{k+1} \\ h_{k+1} \end{pmatrix} = \begin{pmatrix} d_k \\ h_k \end{pmatrix} - \begin{pmatrix} -e_k \\ q_k \end{pmatrix} (-e_k^T \quad q_k^T) \begin{pmatrix} d_k \\ h_k \end{pmatrix} = \begin{pmatrix} d_k + e_k(q_k^T h_k) \\ h_k - q_k(q_k^T h_k) \end{pmatrix}.$$

Starting with $h_1 := b$, this gives the computation

$$\delta_k := q_k^T h_k; \quad h_{k+1} := h_k - q_k \delta_k, \quad k = 1 : n.$$

so $h = h^{(n+1)}$, $d = d^{(n+1)} = (\delta_1, \dots, \delta_n)^T$. The computation for d and h is exactly the same as the one that would arise if MGS had been applied to (A, b) instead of just A . This leads to a backward stable MGS algorithm for computing x .

Using the analogy with Householder QR (cf. (8.3.40)) a backward stable approximation of r is obtained setting

$$\begin{pmatrix} 0 \\ r \end{pmatrix} = P \begin{pmatrix} 0 \\ h_{n+1} \end{pmatrix}, \quad P = P_1 \dots P_{n-1} P_n.$$

where P_k is given by (8.3.64). More generally, to compute an expression $P \begin{pmatrix} z \\ h \end{pmatrix}$, define

$$\begin{pmatrix} w_n \\ y_n \end{pmatrix} = \begin{pmatrix} z \\ h \end{pmatrix}, \quad \begin{pmatrix} w_{k-1} \\ y_{k-1} \end{pmatrix} = P_k \dots P_n \begin{pmatrix} z \\ h \end{pmatrix} = P_k \begin{pmatrix} w_k \\ y_k \end{pmatrix}.$$

This gives the recursion

$$\begin{pmatrix} w_{k-1} \\ y_{k-1} \end{pmatrix} = \begin{pmatrix} w_k \\ y_k \end{pmatrix} - \begin{pmatrix} -e_k \\ q_k \end{pmatrix} (-e_k^T w^{(k)} + q_k^T y_k),$$

which shows in this step only the k th element of w_k is changed from $\zeta_k = e_k^T z$ to $\omega_k = q_k^T y_k$. This gives

$$y_{k-1} := y_k - q_k(\omega_k - \zeta_k), \quad \omega_k := q_k^T y_k \quad k = n : -1 : 1. \quad (8.3.68)$$

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 a reorthogonalize of h_{n+1} against the vectors q_k in backward order. The derivation is summarized in the following algorithm:

Algorithm 8.6. Linear Least Squares Solution by MGS.

Assume that MGS has been applied to $A \in R^{m \times n}$, $\text{rank}(A) = n$, to give Q and R . The following function computes the solution x to the linear least squares $\min_x \|Ax - b\|_2$, the residual r , and its Euclidean norm.

```
function [x,r,rho] = mgss(Q,R,b);
[m,n] = size(Q);
d = zeros(n,1);
for k = 1:n %MGS on b
    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);
```

The MGS algorithm is backward stable also for computing the residual $r = b - Ax$. This means that the computed residual \bar{r} satisfies

$$\|A^T \bar{r}\|_2 \leq cu\|\bar{r}\|_2 \|A\|_2. \quad (8.3.69)$$

for some constant $c = c(m, n)$. As remarked for the Householder algorithm this is much better than if the residual is computed from its definition $r = b - Ax$. If only x and the residual norm $\|r\|$ is needed, then the reorthogonalization of r can be skipped. In that case only $2n(m + n)$ flops are needed for each right hand side.

Remark 8.3.1.

The algorithm above is the same that is obtained if the MGS algorithm is applied to the right-hand side b , i.e., when the right hand side b is treated as an extra $(n + 1)$ st column appended to A , except that the normalization is skipped. This gives the factorization

$$(A \quad b) = (Q_1 \quad r) \begin{pmatrix} R & z \\ 0 & 1 \end{pmatrix}. \quad (8.3.70)$$

and hence $r = b - Q_1 z$. By (8.3.55) the product of the computed factors accurately reproduce the matrix (A, b) . It follows that

$$\|Ax - b\|_2 = \left\| (A \quad b) \begin{pmatrix} x \\ -1 \end{pmatrix} \right\|_2 = \|Q_1(Rx - z) - r\|_2.$$

If $Q_1^T r = 0$ the minimum of the last expression occurs when $Rx - z = 0$ and it is not necessary to require that Q_1 is accurately orthogonal for this conclusion to hold.

An 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 technique above. Using the Householder QR factorization the solution is obtained from

$$z = R^{-T} c, \quad y = Q \begin{pmatrix} z \\ 0 \end{pmatrix},$$

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 (8.3.68)

$$y_{k-1} = y_k - q_k(\omega_k - \zeta_k), \quad \omega_k = q_k^T b, \quad k = n : -1 : 1.$$

to compute $y = y_0$.

Algorithm 8.7. Minimum Norm Solution by MGS.

Assume that MGS has been applied to $A \in R^{m \times n}$, $\text{rank}(A) = n$, to give Q and R . The following function computes the solution y to the linear system $A^T y = c$ which minimizes $\|y\|_2$

```
function [y,rho] = mgsmn(Q,R,c)
```

```
[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);
```

The two MGS algorithm can be generalized to give Based on the Householder QR algorithm given in Theorem 8.3.6 a backward stable MGS algorithm can also be developed for solving the augmented system.

8.3.7 Error Estimation and Iterative Refinement

Using QR factorization with column pivoting a lower bound for $\kappa(A) = \kappa(R)$ can be obtained from the diagonal elements of R . We have $|r_{11}| \leq \sigma_1 = \|R\|_2$, and since the diagonal elements of R^{-1} equal r_{ii}^{-1} , $i = 1 : n$, it follows that $|r_{nn}^{-1}| \leq \sigma_n^{-1} = \|R^{-1}\|_2$, provided $r_{nn} \neq 0$. Combining these estimates we obtain the *lower bound*

$$\kappa(A) = \sigma_1/\sigma_n \geq |r_{11}/r_{nn}| \quad (8.3.71)$$

Although this may considerably underestimate $\kappa(A)$, it has proved to give a fairly reliable estimate in practice. Extensive numerical testing has shown that (8.3.71) usually underestimates $\kappa(A)$ only by a factor of 2–3, and seldom by more than 10.

When column pivoting has not been performed, the above estimate of $\kappa(A)$ is not reliable. Then a condition estimator similar to that described in Section 7.5.3 can be used. Let u be a given vector, and define v and w from

$$R^T v = u, \quad R w = v.$$

We have $w = R^{-1}(R^{-T}u) = (A^T A)^{-1}u$ so this is equivalent to one step of inverse iteration with $A^T A$, and requires about $O(n^2)$ multiplications. Provided that u is suitably chosen (cf. Section 7.5.3).

$$\sigma_n^{-1} \approx \|w\|_2/\|v\|_2$$

will usually be a good estimate of σ_n^{-1} . We can also take u as a random vector and perform 2–3 steps of inverse iteration. This condition estimator will usually detect near rank deficiency even in the case when this is not revealed by a small diagonal element in R .

More reliable estimates can be based on the componentwise error bounds (8.2.30)–(8.2.31). In particular, if $E = |A|$, $f = |b|$, we obtain taking norms the estimates

$$\|\delta x\| \lesssim \omega (\| |A^\dagger|(|b| + |A||x|) \| + \| |(A^T A)^{-1}| |A|^T |r| \|), \quad (8.3.72)$$

$$\|\delta r\| \lesssim \omega (\| |I - AA^\dagger|(|A||x| + |b|) \| + \| |(A^\dagger)^T| |A|^T |r| \|). \quad (8.3.73)$$

For maximum norm the estimate for $\|\delta x\|$ can be written

$$\|\delta x\|_\infty \leq \omega(\|B_1|g_1\|_\infty + \|B_2|g_2\|_\infty), \quad (8.3.74)$$

where

$$B_1 = A^\dagger, \quad g_1 = |b| + |A||x|, \quad B_2 = (A^T A)^{-1}, \quad g_2 = |A^T||r|. \quad (8.3.75)$$

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 have³⁴

$$\|B^{-1}|d\|_\infty = \|B^{-1}|De\|_\infty = \|B^{-1}D|e\|_\infty = \|B^{-1}D\|_\infty = \|B^{-1}D\|_\infty.$$

There are algorithms that produce reliable order-of-magnitude estimates of $\|C^T\|_1 = \|C\|_\infty$, where $C = B^{-1}D$, using only a few matrix-vector products of the form Cx and C^Ty for some carefully selected vectors x and y . If A has full rank and a QR factorization of A is known, then we have

$$A^\dagger = R^{-1}Q^T, \quad (A^\dagger)^T = QR^{-T}.$$

Hence, the required products can be computed inexpensively. For details we refer to Higham [328, Chapter 15].

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 similar to that outlined for linear systems in Section . However, it turns out that this is only satisfactory provided the residual vector is r is sufficiently small. Otherwise the solution and residual vector both have to refined simultaneously as we now describe.

Let x be the solution and r be the residual vector of the least squares problem $\min_x \|Ax - b\|_2$, $A \in \mathbf{R}^{m \times n}$. Then x and the scaled residuals $y = r/\alpha$, $\alpha > 0$, satisfy the symmetric indefinite system of equations

$$\begin{pmatrix} \alpha I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix} \quad (8.3.76)$$

with $c = 0$. Let λ be an eigenvalues of the system matrix M_α and $(x, y)^T$ the corresponding eigenvector. Then

$$\alpha x + Ay = \lambda x, \quad A^T x = \lambda y,$$

³⁴This clever observation is due to Arioli, Demmel, and Duff [12].

and it follows that

$$\alpha\lambda y + A^T A y = \lambda^2 y.$$

If $y \neq 0$ then y is an eigenvector and $(\lambda^2 - \lambda\alpha) = \sigma_i^2$, where $\sigma_i, i = 1 : n$ are the singular values of $A^T A$. On the other hand $y = 0$, implies that $A^T x = 0$, and $\lambda = \alpha$. Thus, the eigenvalues equal

$$\lambda_i = \frac{\alpha}{2} \pm \left(\frac{\alpha^2}{4} + \sigma_i^2 \right)^{1/2}, \quad i = 1 : n,$$

Further, there are $(m - n)$ eigenvalues equal to α .

If we use a solution algorithm for (8.3.76) which is numerically invariant under a scaling α of the $(1, 1)$ -block then the relevant condition number is the smallest condition number of $M(\alpha)$. It can be shown that the minimum occurs for $\alpha^2 = \frac{1}{2}\sigma_n$ and

$$\min_{\alpha} \kappa(M_{\alpha}) = \frac{1}{2} + \left(\frac{1}{4} + 2\kappa(A)^2 \right)^{1/2} \leq 2\kappa(A), \quad (8.3.77)$$

where $\kappa(A) = \sigma_1/\sigma_n$.

We now show how to use the QR factorization to solve the augmented system (8.3.76). Let

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad Q \in \mathbf{R}^{m \times m}, \quad R \in \mathbf{R}^{n \times n}.$$

Using this we can transform the system (8.3.76) into

$$\begin{pmatrix} I & \begin{pmatrix} R \\ 0 \\ 0 \end{pmatrix} \\ (R^T & 0) \end{pmatrix} \begin{pmatrix} Q^T y \\ x \end{pmatrix} = \begin{pmatrix} Q^T b \\ c \end{pmatrix}.$$

It is easily verified that this gives the solution method

$$z = R^{-T} c, \quad \begin{pmatrix} d \\ e \end{pmatrix} = Q^T b, \quad y = Q \begin{pmatrix} z \\ e \end{pmatrix}, \quad x = R^{-1}(d - z). \quad (8.3.78)$$

Here, if $c = 0$ then $z = 0$ and we retrieve the Householder QR algorithm for linear least squares problems. If $b = 0$, then $d = f = 0$, and (8.3.78) gives the QR algorithm for the minimum norm solution of $A^T y = c$. Clearly the algorithm in (8.3.78) is numerically invariant under a scaling of the

In Section 7.5.6 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 $A\delta = \bar{r}$ 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.

We would like to have a similar process to compute highly accurate solutions to the linear least squares problems $\min_x \|Ax - b\|_2$. In the naive approach, we

would then compute the residual $\bar{r} = b - A\bar{x}$ in high precision and then solve $\min_x \|A\delta x - \bar{r}\|_2$ for a correction δ . If a QR factorization in low precision of A is known we compute

$$\delta x = R^{-1}Q^T\bar{r},$$

and then iterate the process. The accuracy of this refinement scheme is not satisfactory unless the true residual $r = b - Ax$ of the least squares problem equals zero. The solution to an efficient algorithm for iterative refinement is to apply the refinement to the augmented system and refine both the solution x and the residual r in each step. In floating-point arithmetic with base β this process of iterative refinement can be described as follows:

```

 $s := 0; \quad x^{(0)} := 0; \quad r^{(0)} := b;$ 
repeat
     $f^{(s)} := b - r^{(s)} - Ax^{(s)};$ 
     $g^{(s)} := c - A^T r^{(s)}; \quad (\text{in precision } u_2 = \beta^{-t_2})$ 
        (solve augmented system in precision  $u_1 = \beta^{-t_1}$ )
     $x^{(s+1)} := x^{(s)} + \delta x^{(s)};$ 
     $r^{(s+1)} := r^{(s)} + \delta r^{(s)};$ 
     $s := s + 1;$ 
end

```

Using the Householder QR factorization with the computed factors \bar{Q} and \bar{R} the method in (8.3.78) can be used to solve for the corrections giving

$$z^{(s)} = \bar{R}^{-T} g^{(s)}, \quad \begin{pmatrix} d^{(s)} \\ e^{(s)} \end{pmatrix} = \bar{Q}^T f^{(s)}, \quad (8.3.79)$$

$$\delta r^{(s)} = \bar{Q} \begin{pmatrix} z^{(s)} \\ e^{(s)} \end{pmatrix}, \quad \delta x^{(s)} = \bar{R}^{-1}(d^{(s)} - z^{(s)}). \quad (8.3.80)$$

Recall that $\bar{Q} = P_n \cdots P_2 P_1$, where P_i are Householder reflections, then $\bar{Q}^T = P_1 P_2 \cdots P_n$. The computation of the residuals and corrections, takes $4mn$ flops in high precision. Computing the solution from (8.3.79)–(8.3.80) 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.

A portable and parallelizable implementation of this algorithm using the extended precision BLAS is available and described in [148].

Review Questions

- 3.1** Let $w \in \mathbf{R}^n$, $\|w\|_2 = 1$. Show that $I - 2ww^T$ is orthogonal. What is the geometrical significance of the matrix $I - 2ww^T$? Give the eigenvalues and eigenvectors of these matrices.

- 3.2** Define a Givens transformations $G_{ij}(\phi) \in \mathbf{R}^{n \times n}$. Give a geometrical interpretations for the case $n = 2$.
- 3.3** Describe the difference between the classical and modified Gram–Schmidt methods for computing the factorization $A = Q_1 R$. What can be said about the orthogonality of the computed matrix Q_1 for these two algorithms?
- 3.4** Define the QR factorization of a matrix $A \in \mathbf{R}^{m \times n}$, in the case that $\text{rank}(A) = n \leq m$. What is its relation to the Cholesky factorization of $A^T A$?
-

Problems

- 3.1** Show that if H is a reflector, that is H is Hermitian and $H^2 = I$, then $P = (I - H)/2$ is an orthogonal projector. Conversely, if P is an orthogonal projector show that $H = I - 2P$ is a projector.
- 3.2** Compute using Householder reflectors P_1, P_2 , the factorization

$$Q^T A = P_2 P_1 A = \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad A = (a_1, a_2) = \begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{pmatrix},$$

to four decimal places

- 3.3** (a) Specialize the formulas in (8.3.5) for a Householder reflection P to the case $n = 2$. What is the relation between this and the corresponding Givens rotations?
(b) How many flops are needed to apply the reflector P to a matrix of dimension $2 \times n$?
- 3.4** Solve the least squares problem $\min_x \|Ax - b\|_2$, where

$$A = \begin{pmatrix} \sqrt{2} & 0 \\ 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

using a QR factorization computed with Givens transformation;

- 3.5** (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

$$A = \tilde{Q}_1 \tilde{R},$$

where \tilde{R} is **unit** upper triangular.

- (b) Suppose the square root free version of modified Gram–Schmidt is used. Modify Algorithm 8.3.6 for computing the least squares solution and residual from this factorization.

Comment: There is no square root free version of Householder QR factorization!

3.6 For any c and s such that $c^2 + s^2 = 1$ we have

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} 0 & s \\ 0 & c \end{pmatrix} = QR.$$

Here $\text{rank}(A) = 1 < 2 = n$. Show that the columns of Q do not provide any information about an orthogonal basis for $\mathcal{R}(A)$.

3.7 (a) If the matrix Q in the QR factorization is explicitly required in the Householder algorithm it can be computed by setting $Q^{(n)} = I_m$, and computing $Q = Q^{(0)}$ by the backward recursion

$$Q^{(k-1)} = P_k Q^{(k)}, \quad k = n : -1 : 1.$$

Show that if advantage is taken of the property that $P_k = \text{diag}(I_{k-1}, \tilde{P}_k)$ this accumulation requires $4(mn(m-n) + n^3/3)$ flops. What is the corresponding operation count if forward recursion is used?

(b) Show how we can compute

$$Q_1 = Q \begin{pmatrix} I_n \\ 0 \end{pmatrix}, \quad Q_2 = Q \begin{pmatrix} 0 \\ I_{m-n} \end{pmatrix}$$

separately in $2(mn^2 - n^3/3)$ and $2(2m^2n - 3mn^2 + n^3)$ multiplications, respectively.

3.8 Let $Q = Q_1 = (q_1, q_2, \dots, q_n) \in \mathbf{R}^{n \times n}$ 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 $\tilde{Q}_2 = (\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_n) \in \mathbf{R}^{(n-1) \times (n-1)}$ 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

$$P_{n-1} \cdots P_2 P_1 Q = \text{diag}(1, \dots, 1, \pm 1).$$

3.9 In several applications one needs 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 transformations which 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.

- 3.10** Show how to compute the QR factorization of a lower triangular matrix $L \in R^{nn}$ in n^3 flops using Givens rotations.

Hint: In the first step elements in the first column are zeroed out by rotating rows $(1, 2), (1, 3), \dots, (1, n)$ in this order. Then the first row and column are finished. The reduction continues in a similar way on the remaining lower triangular matrix of order $(n - 1)$.

- 3.11** 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 $Q_2^T A_2 A_1 = R_2 R_1$.

- 3.12** 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 , as well as the matrix R^{-1} which is required for computing the covariance matrix $\sigma R^{-1} R^{-T}$.

8.4 Rank Deficient Problems

8.4.1 Rank Revealing QR Factorizations

Let $A \in \mathbf{R}^{m \times n}$ be a matrix with $\text{rank}(A) = r \leq \min(m, n)$, and Π a permutation matrix such that the first r columns in $A\Pi$ are linearly independent. Then the QR factorization of $A\Pi$ will have the form

$$A\Pi = (Q_1 \quad Q_2) \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \quad (8.4.1)$$

where $R_{11} \in \mathbf{R}^{r \times r}$ is upper triangular with positive diagonal elements and $R_{11} \in \mathbf{R}^{r \times (n-r)}$. Note that it is not required that $m \geq n$. 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 (8.4.1) is not unique since there are many ways to select r linearly independent columns of A .

To simplify notations we assume in the following that $\Pi = I$. (This is no restriction since the column permutation of A can always be assumed to have been carried out in advance.) Using (8.4.1) the least squares problem $\min_x \|Ax - b\|_2$ is reduced to

$$\min_x \left\| \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \right\|_2, \quad (8.4.2)$$

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 becomes

$$x_1 = R_{11}^{-1}(c_1 - R_{12}x_2) \quad (8.4.3)$$

where x_2 can be chosen arbitrarily. By setting $x_2 = 0$ a particular solution $x_1 = R_{11}^{-1}c_1$ is obtained for which at most $r = \text{rank}(A)$ components are nonzero. This solution is appropriate in applications where it is required to fit the vector b using *as few columns of A* as possible. It is not unique and depends on the initial column permutation. Any such solution x such that Ax only involves at most r columns of A , is called a **basic solution**.

Letting x_2 be an arbitrary vector we have

$$x_1 = d - Cx_2, \quad d = R_{11}^{-1}c_1, \quad C = R_{11}^{-1}R_{12}, \quad (8.4.4)$$

where the $C \in \mathbf{R}^{r \times (n-r)}$ can be computed by solving the triangular systems $R_{11}C = R_{12}$. In particular, the solution of minimum norm, i.e., the pseudo-inverse solution is obtained by solving

$$\min_x \left\| \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right\|_2 = \min_{x_2} \left\| \begin{pmatrix} d \\ 0 \end{pmatrix} - \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix} x_2 \right\|_2. \quad (8.4.5)$$

which is a the linear least squares problem for x_2 . Note that this problem always has full rank and hence has a unique solution. To compute x_2 we could form and solve the normal equations

$$(I + CC^T)x_2 = C^T d.$$

It is usually preferable to compute the Householder QR factorization

$$Q_C^T \begin{pmatrix} C \\ I_{n-r} \end{pmatrix} = \begin{pmatrix} R_C \\ 0 \end{pmatrix}, \quad Q_C^T \begin{pmatrix} d \\ 0 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix},$$

taking into account the special structure of the matrix. Noticing that the pseudo-inverse solution $x = A^\dagger b$ equals *the residual* of the problem (8.4.5), we get

$$x = A^\dagger b = Q_C \begin{pmatrix} 0 \\ d_2 \end{pmatrix}.$$

We further note that for any $z \in \mathbf{R}^{n-r}$ 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 a basis for the null space of A is given by the columns of the matrix W , where

$$\mathcal{N}(A) = \mathcal{R}(W), \quad W = \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix}. \quad (8.4.6)$$

8.4.2 Complete QR Factorizations

In signal processing problems it is often the case that one wants to determine the rank of A as well as the range (signal subspace) and null space of A . Since the data analyzed arrives in real time it is necessary to update an appropriate matrix decompositions at each time step. For such applications the SVD has the disadvantage that it cannot be updated in less than $\mathcal{O}(n^3)$ operations, when rows and/or columns are added or deleted to A . Although the RRQR factorization can be updated, it is less suitable in applications where a basis for the approximate null space of A is needed. The matrix W in (8.4.6) may be ill-conditioned and cannot easily be updated.

This motivates the introduction of the **complete QR factorization** of a matrix $A \in \mathbf{R}^{m \times n}$. This is a factorization of the form

$$A = U \begin{pmatrix} T & 0 \\ 0 & 0 \end{pmatrix} V^T = U_1 T V_1^T, \quad (8.4.7)$$

where $T \in \mathbf{R}^{r \times r}$ is an upper or lower triangular matrix with positive diagonal elements and

$$U = (U_1 \ U_2) \in \mathbf{R}^{m \times m}, \quad V = (V_1 \ V_2) \in \mathbf{R}^{n \times n},$$

are orthogonal matrices partitioned so that $U_1 \in \mathbf{R}^{m \times r}$ and $V_1 \in \mathbf{R}^{n \times r}$. An advantage of the complete QR factorization of A is that, like the SVD, it gives orthogonal bases for all four fundamental subspaces of A . In particular, V_2 gives an orthogonal basis for the null space $\mathcal{N}(A)$. This is often useful in signal processing applications, where one wants to determine the part of the signal that corresponds to noise.

From the orthogonal invariance of the 2-norm it follows that the minimum norm solution of the least squares problem $\min_x \|Ax - b\|_2$ is $x = V_1 T^{-1} U_1^T b$ and the pseudo-inverse of A equals

$$A^\dagger = V \begin{pmatrix} T^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T = V_1 T^{-1} U_1^T. \quad (8.4.8)$$

compute the factorization (8.4.7) we can start from a rank revealing QR factorization

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}.$$

and then construct a sequence of Householder reflections P_k , $k = r : (-1) : 1$, 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 (8.4.7) holds with T upper triangular and

$$V = \Pi P_1 \cdots P_r.$$

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 reflection P_2 to zero the elements in position (2,5) and (2,6). In general, in step k a full matrix of size $k \times (n-r+1)$ is transformed by a Householder reflection. The transformations require a total of $2r^2(n-r+1)$ flops.

Even for a rank deficient matrix A of rank $r < n$, a rank revealing QR factorization will yield a factorization

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}.$$

where R_{22} is nonzero but $\|R_{22}\| \leq \epsilon$, where ϵ is small. In a similar way the complete QR factorization can be generalized to the case when A is only numerically rank deficient. We will consider URV factorizations, which have of the form

$$A = U RV^T = (U_1 \quad U_2) \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}, \quad (8.4.9)$$

where U and V are orthogonal matrices, $R_{11} \in \mathbb{R}^{r \times r}$, and satisfy

$$\sigma_r(R_{11}) \geq \frac{1}{c} \sigma_r, \quad (\|R_{12}\|_F^2 + \|R_{22}\|_F^2)^{1/2} \leq c \sigma_{r+1}. \quad (8.4.10)$$

Note that here both submatrices R_{12} and R_{22} are required to have small elements. From (8.4.9) we have

$$\|AV_2\|_2 = \left\| \begin{pmatrix} R_{12} \\ R_{22} \end{pmatrix} \right\|_F \leq c \sigma_{r+1},$$

and hence the orthogonal matrix V_2 can be taken as an approximation to the numerical null space \mathcal{N}_r .

Algorithms for computing an URV decomposition start with an initial QR factorization with column pivoting. Then 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 $P^T R Q = \hat{R}$ where \hat{R} is upper triangular. Then

$$\|\hat{R}e_n\| = \|P^T R Q Q^T w\| = \|P^T R w\| = \sigma_n,$$

which shows that the entire last column in \hat{R} is small. Given w the matrices P and Q can be constructed as a sequence of Givens rotations. Efficient algorithms can be given for updating an URV decomposition when a new row is appended to A .

Like the RRQR factorizations the URV decomposition yield approximations to the singular values. In [428] the following bounds are derived

$$f\sigma_i \leq \sigma_i(R_{11}) \leq \sigma_i, \quad i = 1 : r,$$

and

$$\sigma_i \leq \sigma_{i-k}(R_{22}) \leq \sigma_i/f, \quad i = r+1 : n,$$

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 .

An alternative decomposition that is more satisfactory for applications where an accurate approximate null space is needed, is the rank-revealing **ULV decomposition**

$$A = U \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} V^T. \quad (8.4.11)$$

where the middle matrix has lower triangular form. For this decomposition

$$\|AV_2\|_2 = \|L_{22}\|_F, \quad V = (V_1, V_2),$$

and hence the size of $\|L_{21}\|$ does not adversely affect the null space approximation. On the other hand the URV decomposition usually gives a superior approximation for the numerical range space and the updating algorithm for URV is much simpler. For recent work see also citebaes:05,baer:08

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 [103]. Matlab templates for rank-revealing UTV decompositions are described in [203].

8.4.3 Column Subset Selection Problem

In the column **subset selection** problem we are given a matrix $A \in \mathbf{R}^{m \times n}$ and want to determine a subset A_1 of $k < n$ columns such that

$$\|A - (A_1 A_1^\dagger) A\|$$

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 are maximized.

The column subset selection problem is closely related to rank-revealing QR factorization. The following theorem, which we state without proof, shows that a column permutation Π always exists such that the numerical rank of A is revealed by the QR factorization of $A\Pi$.

Theorem 8.4.1 (*H. P. Hong and C. T. Pan [337]*).

Let $A \in \mathbf{R}^{m \times n}$, ($m \geq n$), and r be a given integer $0 < r < n$. Then there exists a permutation matrix Π_r , such that the QR factorization has the form

$$Q^T A \Pi_r = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \quad (8.4.12)$$

with $R_{11} \in \mathbf{R}^{r \times r}$ upper triangular, and

$$\sigma_{\min}(R_{11}) \geq \frac{1}{c} \sigma_r(A), \quad \sigma_{\max}(R_{22}) \leq c \sigma_{r+1}(A), \quad (8.4.13)$$

where $c = \sqrt{r(n-r) + \min(r, n-r)}$.

If A has a well defined numerical rank $r < n$, i.e.,

$$\sigma_1 \geq \dots \geq \sigma_r \gg \delta \geq \sigma_{r+1} \geq \dots \geq \sigma_n,$$

then the above theorem says that if the ratio σ_k/σ_{k+1} is sufficiently large then there is a permutation of the columns of A such that the rank of A is revealed by the QR factorization. Unfortunately, to find such a permutation is an NP hard problem.

There are heuristic algorithms which almost always succeeds in practice. The following SVD-based algorithm for finding a good approximation to the solution of the subset selection problem is due to Golub, Klema, and Stewart [266]; see also Golub and Van Loan [277, Sec. 12.2].

Let k be the numerical rank of A determined from the SVD $A = U\Sigma V^T$. If P is any permutation matrix, then $U^T(AP)(P^TV) = \Sigma$. Thus, permutation of the columns of A correspond to a permutation of the rows of the orthogonal matrix V of right singular vectors. The theoretical basis for this column selection strategy is the following result.

Theorem 8.4.2 (**Theorem 6.1**, [266]).

Let $A = U\Sigma V^T \in \mathbf{R}^{m \times n}$ be the SVD of A . Partition A and V conformally as $A = (A_1 \ A_2)$ and $V = (V_1 \ V_2)$, where

$$(V_1 \ V_2) = \frac{k}{n-k} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}, \quad (8.4.14)$$

and \hat{V}_{11} is nonsingular. Then

$$\frac{\sigma_k(A)}{\|V_{11}^{-1}\|_2} \leq \sigma_k(A_1) \leq \sigma_k(A). \quad (8.4.15)$$

Proof. The upper bound in (8.4.15) follows directly from the interlacing property of the singular values of A_1 and A ; see Theorem 8.1.17. If we set $A(V_1 \ V_2) = (S_1 \ S_2)$, then $S_1^T S_2 = 0$. Now since $A = SV^T$, we have

$$A_1 = S_1 V_{11}^T + S_2 V_{21}^T.$$

If we define $\inf(A) = \inf_{\|x\|_2=1} \|Ax\|_2$, then

$$\inf(A_1) \geq \inf(S_1 V_{11}) \geq \sigma_k(A) \inf(V_{11}).$$

Since $\inf(V_{11}) = \|V_{11}^{-1}\|_2$ the lower bound follows. \square

The above result suggests that we choose a permutation P such that the submatrix \hat{V}_{11} consisting of the first k columns of $\hat{V} = VP$ is as well-conditioned as possible. This can be achieved by using QR factorization with column pivoting to compute

$$Q^T (V_{11}^T \ V_{21}^T) P = (R_{11} \ R_{12}).$$

The least squares problem $\min_z \|A_1 z - b\|_2$ can then be solved by QR factorization.

From the interlacing properties of singular values (Theorem 8.1.17) it follows by induction that for any factorization of the form (8.4.12), we have the inequalities

$$\sigma_{\min}(R_{11}) \leq \sigma_r(A), \quad \sigma_{\max}(R_{22}) \geq \sigma_{r+1}(A). \quad (8.4.16)$$

Hence, to achieve (8.4.13) we want to choose the permutation Π to maximize $\sigma_{\min}(R_{11})$ and simultaneously minimize $\sigma_{\max}(R_{22})$. These two problems are in a certain sense dual; cf. Problem 8.4.3.

From Theorem 8.1.25 it follows that in (8.4.14) $\|V_{11}\|_2 = \|V_{22}\|_2$. When $k > n/2$ it is more economical to use the QR factorization with column pivoting applied to the smaller submatrix $V_2^T = (V_{12}^T, V_{22}^T)$. This will then indicate *what columns in A to drop*.

In case $k = n - 1$, the column permutation P corresponds to the largest component in the right singular vector $V_2 = v_n$ belonging to the smallest singular value σ_n . The index indicates what column to drop. In a related strategy due to T. F. Chan [99] repeated use is made of this, by dropping one column at a time. The right singular vectors needed are determined by inverse iteration (see Section 9.4.3).

A comparison between the RRQR and the above SVD-based algorithms is given by Chan and Hansen [102, 1992]. Although in general the methods will not necessarily compute equivalent solutions, the subspaces spanned by the two sets of selected columns are still almost identical whenever the ratio σ_{k+1}/σ_k is small. The best bounds are those given by Gu and Eisenstat [292]. Chandrasekaran and Ipsen [105].

8.4.4 Modified Least Squares Problems

Suppose that we have solved the least squares problem $\min_x \|Ax - b\|_2$, where $A \in \mathbf{R}^{m \times n}$, $m > n$. It may often be required to later solve a related least squares problem where a simple modification of the data (A , b) has been performed. For example, one may want to add new observations or discard observations with unacceptably large residuals. In various time-series problems data are arriving sequentially and a least squares solution has to be updated at each time step. Such modifications are usually referred to as **updating** when (new) data is added and **downdating** when (old) data is removed. In time-series problems a **sliding window method** is often used. At each time step a new row of data is added and the oldest row of data deleted. Applications in signal processing often require real-time solutions so efficiency is critical.

Other applications arise in optimization and statistics. Indeed, the first systematic use of such algorithms seems to have been in optimization. In linear regression efficient and stable procedure for adding and/or deleting observations is needed. In stepwise regression one wants to examine different models by adding and/or deleting variables in each step. Another important application occurs in active set methods for solving least squares problems with inequality constraints.

We will consider the following type of modifications:

1. A general rank-one change to $(A \ b)$.
2. Adding or deleting a row of $(A \ b)$.
3. Adding or deleting a column of A .

If a row is added to A , then by the interlacing property (Theorem 8.1.17) the smallest singular value of the modified matrix will not decrease. On the other hand, when a row is deleted the rank can decrease. Similarly, when a column is deleted the smallest singular value will not decrease. When a column is added the modified matrix may become singular. This indicates that deleting a column and adding a row are “easy” operations, whereas adding a column and deleting a row can be more delicate operations. In general, we can not expect modification methods to be stable when the unmodified problem is much worse conditioned than the modified problem,

Another aspect that has to be considered is what factorization is available of the original matrix A . The simplest case is when the full QR factorization with $Q \in \mathbf{R}^{m \times m}$ explicitly known. In cases when $n \ll m$ it may be more desirable to update the thin QR factorization. Finally, sometimes only the factor R may be known. This is the case, e.g., when the initial problem has been solved by the normal equations.

Recursive Least Squares.

We first describe a simple algorithm for adding and deleting rows based on the Cholesky factorization and normal equations. The solution to the least squares

problem satisfies the normal equations $A^T A x = A^T b$. If the equation $w^T x = \beta$ is added, then the updated solution \tilde{x} satisfies the modified normal equations

$$(A^T A + w w^T) \tilde{x} = A^T b + \beta w, \quad (8.4.17)$$

where we assume that $\text{rank}(A) = n$. Let R is the Cholesky factor of $A^T A$. We would like to avoid computing the Cholesky factorization of the modified problem from scratch. In the **covariance matrix method**, the covariance matrix

$$C = (A^T A)^{-1} = R^{-1} R^{-T},$$

is updated. Since $\tilde{C}^{-1} = C^{-1} + w w^T$, we have by the Sherman–Morrison formula (7.1.26)

$$\tilde{C} = C - \frac{1}{1 + w^T u} u u^T, \quad u = C w. \quad (8.4.18)$$

Adding the term $w w^T x = (w^T x) w$ to both sides of the unmodified normal equations and subtracting from (8.4.17) gives

$$(A^T A + w w^T)(\tilde{x} - x) = (\beta - w^T x) w.$$

Solving for the updated solution gives the following basic formula:

$$\tilde{x} = x + (\beta - w^T x) \tilde{u}, \quad \tilde{u} = \tilde{C} w. \quad (8.4.19)$$

Equations (8.4.18)–(8.4.19) define a **recursive least squares** algorithm associated with the Kalman filter. The vector $\tilde{u} = \tilde{C} w$ which weights the predicted residual $\beta - w^T x$ of the new observation is called the **Kalman gain vector**.

The equations (8.4.18)–(8.4.19) can, with slight modifications, be used also for *deleting* an observation $w^T x = \beta$. We have

$$\tilde{C} = C + \frac{1}{1 - w^T u} u u^T, \quad \tilde{x} = x - (\beta - w^T x) \tilde{u}, \quad (8.4.20)$$

provided that $1 - w^T u \neq 0$.

The simplicity and recursive nature of this updating algorithm has made it popular for many applications. The main disadvantage of the algorithm is its serious sensitivity to roundoff errors. The updating algorithms based on orthogonal transformations developed in Section 8.4.4 are therefore generally to be preferred.

The method can also be used together with updating schemes for the Cholesky factor factor R or its inverse R^{-1} , where $A^T A = R^T R$. The Kalman gain vector can then be computed from

$$z = \tilde{R}^{-T} w, \quad p = \tilde{R}^{-1} z.$$

Such methods are often referred to as “square root methods” in the signal processing literature. Schemes for updating R^{-1} are described in [61, Sec. 3.3]. Since no back-substitutions is needed in these schemes, they are easier to parallelize.

Updating the QR Factorization

Algorithms for updating the thin QR factorization when $Q \in \mathbf{R}^{m \times n}$ and $R \in \mathbf{R}^{n \times n}$ are explicitly known were first described by Daniel, Gragg, Kaufman, and Stewart [134]. These algorithms use Gram–Schmidt with reorthogonalization combined with Givens rotations. Fortran subroutines that were modifications of the Algol procedures given there appeared in Reichel and Gragg [497].

We first consider updating the QR factorization of a matrix $A \in \mathbf{R}^{m \times n}$ of full column rank. A first observation is not feasible to update a QR factorization where Q is stored implicitly as a product of Householder transformations. Therefore, we assume that

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad (8.4.21)$$

where the orthogonal factor $Q \in \mathbf{R}^{m \times m}$ is known explicitly. These updating algorithms require $O(m^2)$ multiplications, and are (almost) normwise backward stable. We want to compute the factorization when A is subject to a rank one change

$$\tilde{A} = A + uv^T = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \quad (8.4.22)$$

$u \in \mathbf{R}^m$ and $v \in \mathbf{R}^n$. We assume that $\text{rank}(A) = \text{rank}(\tilde{A}) = n$, so that R and \tilde{R} are uniquely determined. The algorithm proceeds as follows:

1. Compute the vector $w = Q^T u \in \mathbf{R}^m$, so that

$$A + uv^T = Q \left[\begin{pmatrix} R \\ 0 \end{pmatrix} + wv^T \right]. \quad (8.4.23)$$

2. Determine a sequence of Givens rotations $J_k = G_{k,k+1}(\theta_k)$, $k = m-1 : (-1) : 1$ such that

$$J_1^T \cdots J_{m-1}^T w = \alpha e_1, \quad \alpha = \pm \|w\|_2.$$

These transformations zero the last $m-1$ components of w from bottom up. (For details on how to compute J_k see Section 8.3.1.) Apply these transformations to R to obtain

$$\tilde{H} = J_1^T \cdots J_{m-1}^T \left[\begin{pmatrix} R \\ 0 \end{pmatrix} + wv^T \right] = H + \alpha e_1 v^T. \quad (8.4.24)$$

(Note that J_{m+1}, \dots, J_{m-1} have no effect on R .) Because of the structure of the Givens rotations the matrix H will be an upper Hessenberg matrix, i.e., H is triangular except for extra nonzero elements $h_{k+1,k}$, $k = 1, 2, \dots, n$ (e.g., $m = 6, n = 4$),

$$H = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since only the first row of H is modified by the term $\alpha e_1 v^T$, \tilde{H} is also upper Hessenberg.

3. Determine Givens rotations $\tilde{J}_k = G_{k,k+1}(\phi_k)$, $k = 1 : n$, to zero the element in position $(k+1, k)$, so that

$$\tilde{J}_n^T \cdots \tilde{J}_1^T \tilde{H} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} \quad (8.4.25)$$

is upper triangular.

4. Accumulate the transformations into Q to get

$$\tilde{Q} = QU, \quad U = J_{m-1} \cdots J_1 \tilde{J}_1 \cdots \tilde{J}_n.$$

This gives the desired factorization (8.4.22).

The work needed for this update is as follows: computing $w = Q^T u$ takes m^2 flops. Computing H and \tilde{R} takes $4n^2$ flops and accumulating the transformations J_k and \tilde{J}_k into Q takes $4(m^2 + mn)$ flops for a total of $5m^2 + 4mn + 4n^2$ flops. Hence, the work has been decreased from $O(mn^2)$ to $O(m^2)$. However, if n is small updating may still be more expensive than computing the factorization from scratch.

There is a simple relationship between the problem of updating matrix factorizations and that of updating the least squares solutions. Recall that if A has full column rank and the R -factor of the matrix (A, b) is

$$\begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}, \quad (8.4.26)$$

then the solution to the least squares problem $\min_x \|Ax - b\|_2$ is given by

$$Rx = z, \quad \|Ax - b\|_2 = \rho. \quad (8.4.27)$$

The upper triangular matrix (8.4.26) can be computed either from the QR factorization of (A, b) or as the Cholesky factor of $(A, b)^T (A, b)$. Hence, to get an updating algorithm for least squares solutions, we only have to apply an updating algorithm for matrix factorization to the extended matrix (A, b) .

To update a least squares solution under a rank one change one applies the updating procedure above to compute the QR factorization of

$$(A + uv^T \quad b) = (A \quad b) + u(v^T \quad 0).$$

where the right hand side has been appended. [164, 1979, Chap. 10].

Algorithms for modifying the full QR factorization of (A, b) when a column is deleted or added are special cases of the rank one change; see Björck [61, Section .3.2]. Adding a row is simple, since the QR factorization is easily organized to treat a row at a time. A more subtle problem is to modify the QR factorization when a row is *deleted*, which is called the **downdating** problem. This corresponds to the problem of deleting an observation in a least squares problem. In the sliding window method one row is added and simultaneously one row is deleted. Another

instance when a row has to be removed is when it has somehow been identified as faulty.

There is no loss of generality in assuming that it is the *first row* of A that is to be deleted. We wish to obtain the QR factorization of the matrix $\tilde{A} \in \mathbf{R}^{(m-1) \times n}$ when

$$A = \begin{pmatrix} a_1^T \\ \tilde{A} \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \quad (8.4.28)$$

is known. We now show that this is equivalent to finding the QR factorization of (e_1, A) , where a dummy column $e_1 = (1, 0, \dots, 0)^T$ has been added. We have

$$Q^T(e_1, A) = \begin{pmatrix} q_1 & R \\ q_2 & 0 \end{pmatrix},$$

where $q^T = (q_1^T, q_2^T) \in \mathbf{R}^m$ is the *first row* of Q . We now determine Givens rotations $J_k = G_{k,k+1}$, $k = m-1 : -1 : 1$, so that

$$J_1^T \cdots J_{m-1}^T q = \alpha e_1, \quad \alpha = \pm 1. \quad (8.4.29)$$

Then we have

$$J_1^T \cdots J_{m-1}^T \begin{pmatrix} q_1 & R \\ q_2 & 0 \end{pmatrix} = \begin{pmatrix} \alpha & v^T \\ 0 & \tilde{R} \\ 0 & 0 \end{pmatrix}, \quad (8.4.30)$$

where the matrix \tilde{R} is upper triangular. Note that the transformations J_{n+1}, \dots, J_{m-1} will not affect R . Further, if we compute

$$\bar{Q} = Q J_{m-1} \cdots J_1,$$

it follows from (8.4.29) that the first row of \bar{Q} equals αe_1^T . Since \bar{Q} is orthogonal it must have the form

$$\bar{Q} = \begin{pmatrix} \alpha & 0 \\ 0 & \tilde{Q} \end{pmatrix},$$

with $|\alpha| = 1$ and $\tilde{Q} \in \mathbf{R}^{(m-1) \times (m-1)}$ orthogonal. From (8.4.29),

$$\begin{pmatrix} a_1^T \\ \tilde{A} \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \tilde{Q} \end{pmatrix} \begin{pmatrix} v^T \\ \tilde{R} \\ 0 \end{pmatrix},$$

and hence the desired factorization is

$$\tilde{A} = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}.$$

This algorithm for downdating is a special case of the rank one change algorithm, and is obtained by taking $u = -e_1$, $v^T = a_1^T$ in (8.4.22).

8.4.5 Golub–Kahan Bidiagonalization

So far we have considered methods for solving least squares problems based on reducing the matrix $A \in \mathbf{R}^{m \times n}$ to upper triangular (or trapezoidal) form using unitary operations. It is possible to carry this reduction further and obtain a lower bidiagonal matrix ($m \geq n$)

$$U^T A V = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad B = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & \ddots & \ddots & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix} \in \mathbf{R}^{(n+1) \times n}, \quad (8.4.31)$$

where U and V are square orthogonal matrices. The bidiagonal form is the closest that can be achieved by a finite process to the diagonal form in SVD. The bidiagonal decomposition (8.4.31) therefore is usually the first step in computing the SVD of A ; see Section 9.7. It is also powerful tool for solving various least squares problems.

Note that from (8.4.31) it follows that $A^T = VB^TU^T$, where B^T is upper bidiagonal. Thus, if we apply the same process to A^T we obtain an *upper* bidiagonal form. A complex matrix $A \in \mathbf{C}^{m \times n}$ can be reduced by a similar process to *real* bidiagonal form using unitary transformations U and V . We consider here only the real case, since the generalization to the complex case is straightforward.

The simplest method for constructing the bidiagonal decomposition is the **Golub–Kahan algorithm** in which the reduction is achieved by applying a sequence of Householder reflections alternately from left and right. We set $A = A^{(1)}$ and in the first double step compute

$$A^{(2)} = Q_1(AP_1) = \begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ \beta_2 & \tilde{\alpha}_{22} & \cdots & \tilde{\alpha}_{2n} \\ 0 & \tilde{\alpha}_{32} & \cdots & \tilde{\alpha}_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{\alpha}_{m2} & \cdots & \tilde{\alpha}_{mn} \end{pmatrix}.$$

Here the Householder reflection P_1 is chosen to zero the last $n - 1$ elements in the first row of A . Next Q_1 is chosen to zero the last $m - 2$ elements in the the first column of AP_1 . This key thing to observe is that Q_1 will leave the first row in AP_1 unchanged and thus not affect the zeros introduced by P_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 Q_k and P_k are Householder reflections. Here P_k is chosen to zero the last $n - k$ elements in the k th row of $A^{(k)}$. Then Q_k is chosen to zero the last $m - (k + 1)$ elements in the k th column of $A^{(k)}P_k$.

The process is continued until either the rows or columns are exhausted. When $m > n$ the process ends with the factorization

$$U = Q_1 Q_2 \cdots Q_n, \quad V = P_1 P_2 \cdots P_{n-1}. \quad (8.4.32)$$

Note that since Q_k , only works on rows $k + 1 : m$, and P_k , only works on columns $k : m$. It follows that

$$u_1 = e_1, \quad u_k = Ue_k = Q_1 \cdots Q_k e_k, \quad k = 2 : n, \quad (8.4.33)$$

$$v_k = Ve_k = P_1 \cdots P_k e_k, \quad k = 1 : n - 1, \quad v_n = e_n. \quad (8.4.34)$$

If $m \leq n$ then we obtain

$$U^T AV = (B \ 0), \quad B = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \ddots & \alpha_{m-1} \\ & & & \beta_m & \alpha_m \end{pmatrix} \in \mathbf{R}^{m \times m}.$$

$$U = Q_1 Q_2 \cdots Q_{m-2}, \quad V = P_1 P_2 \cdots P_{m-1}.$$

The above process can always be carried through although some elements in B may vanish. Note that the singular values of B equal those of A ; in particular, $\text{rank}(A) = \text{rank}(B)$. Using complex Householder transformations (see Section sec8.3.2) a complex matrix A can be reduced to *real* bidiagonal form by the algorithm above. In the nondegenerate case when all elements in B are nonzero, the bidiagonal decomposition is uniquely determined first column $u_1 := Ue_1$, which can be chosen arbitrarily.

The Householder reduction to bidiagonal form 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

$$\|E\|_F \leq cn^2 u\|A\|_F, \quad (8.4.35)$$

and c is a constant of order unity. Moreover, if we use the information stored to generate the products $U = Q_1 \cdots Q_n$ and $V = P_1 \cdots P_{n-2}$, then the computed matrices are close to the exact matrices U and V which reduce $A + E$. This will guarantee that the singular values and transformed singular vectors of \bar{B} are accurate approximations to those of a matrix close to A .

The bidiagonal reduction algorithm as described above requires approximately

$$4(mn^2 - n^3/3) \text{ flops}$$

when $m \geq n$, which is twice the work for a Householder QR factorization. The Householder vectors associated with U can be stored in the lower triangular part of A and those associated with V in the upper triangular part of A . Normally U and V are not 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.

When $m \gg n$ it is more efficient to use a two-step procedure as originally suggested by Lawson and Hanson [399] and later analyzed by T. Chan. In the first step the QR factorization of A is computed (possibly using column pivoting)

$$AP = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R \in \mathbf{R}^{n \times n},$$

which requires $4mn^2 - \frac{2}{3}n^3$ 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 $\frac{4}{3}n^3$ flops. The complete reduction to bidiagonal form then takes a total of

$$2(mn^2 + n^3) \text{ flops.}$$

This is less than the original Golub–Kahan algorithm when $m/n > 5/3$.

8.4.6 Regularization by TSVD and Partial Least Squares

In Example 7.1.5 we considered an integral equation of the first kind.

$$\int K(s, t)f(t)dt = g(s), \quad (8.4.36)$$

where the operator K is compact. This is an ill-posed problem in the sense that the solution f does not depend continuously on the data g . This is because there are rapidly oscillating functions $f(t)$, which come arbitrarily close to being annihilated by the integral operator. As shown in Example 7.1.5, when the the integral equation is discretized, this gives rise to a linear system $Ax = b$, or, more generally, a linear least squares problem

$$\min_x \|Ax - b\|_2, \quad A \in \mathbf{R}^{m \times n}, \quad (8.4.37)$$

The singular values σ_i of A will cluster at zero leading to a huge condition number of A . However, the *exact* right hand side b will be such that in the expansion of the solution in terms of the SVD

$$x_i = \sum_{i=1}^n \frac{c_i}{\sigma_i} v_i, \quad c_i = u_i^T b, \quad (8.4.38)$$

the coefficients c_i decrease faster than σ_i . This is a consequence of the **Picard condition** for the continuous problem. In spite the large condition number of A , the problem is in some sense quite well-conditioned. However, in practice, the exact right hand side is contaminated by noise in a way that affects *all coefficients* c_i in (8.4.38) more or less equally. Therefore, unless some sort of regularization is introduced, the computed solution may be useless.

The solution to the linear least squares problem (8.4.37) can be expressed in terms of the SVD expansion

$$x = \sum_{i=1}^r \frac{u_i^T b}{\sigma_i} \cdot v_i.$$

If the matrix A is ill-conditioned and possibly rank deficient. with numerical rank $k < n$, then a more stable approximate solution is obtained by discarding terms

corresponding to small singular values in this expansion. This gives the **truncated SVD** (TSVD) solution

$$x = \sum_{i=1}^r \frac{u_i^T b}{\sigma_i} \cdot v_i. \quad (8.4.39)$$

The TSVD solution is a reliable way to determine an approximate pseudo-inverse solution of a numerically rank deficient least squares problems. However, it is also an expensive method since it requires the computation of the SVD. Often a truncated QR factorization works well, provided that some form of column pivoting is carried out. An alternative to truncated SVD is to constrain the solution by adding a quadratic constraint; see Section 8.6.4.

Example 8.4.1.

The linear system $Ax = b$, $A \in \mathbf{R}^{n \times n}$ in Example 7.1.5 was derived from a discretization of an integral equation of the first kind. For $n = 100$ the singular values σ_k of A are close to the roundoff level for $k > 30$. Hence, the linear system is highly ill-conditioned. Let $b = Ax + \eta$ be a perturbed right-hand side, where x is a smooth solution and η a random noise vector with normally distributed components of mean zero and variance 10^{-3} . Figure 8.4.1 shows that the smallest error occurs

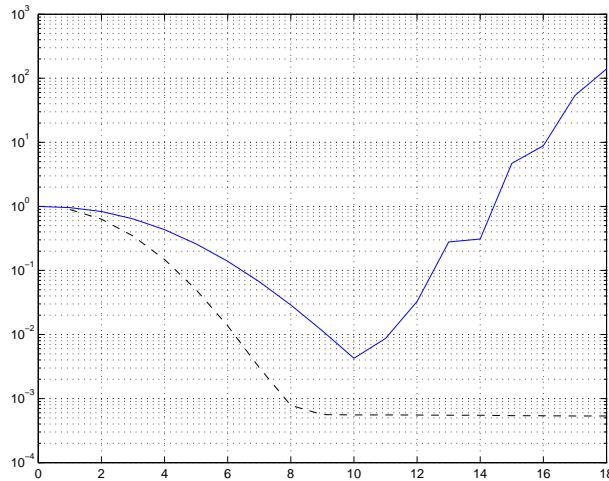


Figure 8.4.1. Relative error $\|x_k - x\|_2 / \|x\|_2$ and residual $\|Ax_k - b\|_2 / \|b\|_2$ for TSVD solutions truncated after k steps.

for $k = 10$. For larger values of k the error increases very rapidly. In practice, the error is unknown and we can only observe the residual. 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 is known as the **discrepancy principle**.

In scientific computing one often has one set of variables called the factors that is used to control, explain, or predict another variable. In multiple linear regression

the set of factors may be large and possibly highly collinear. Then one wants to express the solution by restricting it to lie in a lower dimensional subspace. One way to achieve this is to use a TSVD solution, in which a subspace spanned by the right singular vectors corresponding to the large singular values is used.

An alternative is to use a solution computed from a partial bidiagonalization of $(b \ A)$. This is equivalent to the **partial least squares** (PLS) method. It has been observed that compared to TSVD, PLS gives the same accuracy often with a lower dimensional subspace.

We first derive an algorithm for solving the linear least squares problem $\min \|Ax - b\|_2$, where $A \in \mathbf{R}^{m \times n}$, $m \geq n$, by bidiagonal decomposition. Let Q_1 be a Householder reflection such that

$$Q_1 b = \beta_1 e_1. \quad (8.4.40)$$

Using the Golub–Kahan algorithm to transform $P_1 A$ to lower triangular form, we obtain

$$U_n^T (b \ A) \begin{pmatrix} 1 & 0 \\ 0 & V_n \end{pmatrix} = U_n^T (b \ AV_n) = \begin{pmatrix} \beta_1 e_1 & B_n \\ 0 & 0 \end{pmatrix}, \quad (8.4.41)$$

where e_1 is the first unit vector, $B_n \in \mathbf{R}^{(n+1) \times n}$ is lower bidiagonal, and

$$U_n = (u_1, u_2, \dots, u_n) = Q_1 Q_2 \cdots Q_{n+1}, \quad (8.4.42)$$

$$V_n = (v_1, v_2, \dots, v_n) = P_1 P_2 \cdots P_{n-1}. \quad (8.4.43)$$

(Note the minor difference in notation in that Q_{k+1} now zeros elements in the k th column of A .)

Setting $x = V_n y$ and using the invariance of the l_2 -norm it follows that

$$\begin{aligned} \|b - Ax\|_2 &= \left\| (b \ A) \begin{pmatrix} -1 \\ x \end{pmatrix} \right\|_2 = \left\| U_n^T (b \ AV_n) \begin{pmatrix} -1 \\ y \end{pmatrix} \right\|_2 \\ &= \|\beta_1 e_1 - B_n y\|_2. \end{aligned}$$

Hence, if y is the solution to the bidiagonal least squares problem

$$\min_y \|B_n y - \beta_1 e_1\|_2, \quad (8.4.44)$$

then $x = V_n y$ solves the least squares problem $\|Ax - b\|_2$.

To solve (8.4.44) stably we need the QR factorization of $(B_n \ | \ \beta_1 e_1)$. This can be constructed by premultiplying with a sequence of Givens rotations, where $G_{k,k+1}$, $k = 1, 2, \dots$ is used to zero the element β_{k+1}

$$Q^T (B_n \ | \ \beta_1 e_1) = \left(\begin{array}{c|c} R_n & f_k \\ \hline & \phi_{n+1} \end{array} \right) = \left(\begin{array}{ccc|c} \rho_1 & \theta_2 & & \phi_1 \\ & \rho_2 & \ddots & \phi_2 \\ & & \ddots & \vdots \\ & & & \phi_n \\ \hline & \rho_n & & \phi_{n+1} \end{array} \right) \quad (8.4.45)$$

where Q is a product of n Givens rotations. The solution y is then obtained by back-substitution from $R_n y = d_n$. The norm of the corresponding residual vector equals $|\bar{\phi}_{n+1}|$.

To zero out the element β_2 we premultiply rows (1,2) with a rotation G_{12} , giving

$$\begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix} \left(\begin{array}{cc|c} \alpha_1 & 0 & \beta_1 \\ \beta_2 & \alpha_2 & 0 \end{array} \right) = \left(\begin{array}{cc|c} \rho_1 & \theta_2 & \phi_1 \\ 0 & \bar{\rho}_2 & \phi_2 \end{array} \right). \quad (8.4.46)$$

(Here and in the following only elements affected by the rotation are shown.) Here the elements ρ_1, θ_2 and ϕ_1 in the first row are final, but $\bar{\rho}_2$ and $\bar{\phi}_2$ will be transformed into ρ_2 and ϕ_2 in the next step.

Continuing in this way in step j the rotation $G_{j,j+1}$ is used to zero the element β_{j+1} . In steps, $j = 2 : n - 1$, the rows $(j, j + 1)$ are transformed

$$\begin{pmatrix} c_j & s_j \\ -s_j & c_j \end{pmatrix} \left(\begin{array}{cc|c} \bar{\rho}_j & 0 & \bar{\phi}_j \\ \beta_{j+1} & \alpha_{j+1} & 0 \end{array} \right) = \left(\begin{array}{cc|c} \rho_j & \theta_{j+1} & \phi_j \\ 0 & \bar{\rho}_{j+1} & \bar{\phi}_{j+1} \end{array} \right). \quad (8.4.47)$$

where

$$\begin{aligned} \phi_j &= c_j \bar{\phi}_j, & \bar{\phi}_{j+1} &= -s_j \bar{\phi}_j, & \rho_j &= \sqrt{\bar{\rho}_j^2 + \beta_{j+1}^2}, \\ \theta_{j+1} &= s_j \alpha_{j+1}, & \bar{\rho}_{n+1} &= c_j \alpha_{j+1}. \end{aligned}$$

Note that by construction $|\bar{\phi}_{j+1}| \leq |\bar{\phi}_j|$. Finally, in step n we obtain

$$\begin{pmatrix} c_n & s_n \\ -s_n & c_n \end{pmatrix} \left(\begin{array}{cc|c} \bar{\rho}_n & \bar{\phi}_n \\ \beta_{n+1} & 0 \end{array} \right) = \left(\begin{array}{c|c} \rho_n & \phi_n \\ 0 & \bar{\phi}_{n+1} \end{array} \right). \quad (8.4.48)$$

After n steps, we have obtained the factorization (8.4.45) with

$$G_n = G_{n,n+1} \cdots G_{23} G_{12}.$$

Now consider the result after $k < n$ steps of the above bidiagonalization process have been carried out. At this point we have computed $Q_1, Q_2, \dots, Q_{k+1}, P_1, P_2, \dots, P_k$ such that the first k columns of A are in lower bidiagonal form, i.e.

$$Q_{k+1} \cdots Q_2 Q_1 A P_1 P_2 \cdots P_k \begin{pmatrix} I_k \\ 0 \end{pmatrix} = \begin{pmatrix} B_k \\ 0 \end{pmatrix} = \begin{pmatrix} I_{k+1} \\ 0 \end{pmatrix} B_k,$$

where $B_k \in \mathbf{R}^{(k+1) \times k}$ is a leading submatrix of B_n . Multiplying both sides with $Q_1 Q_2 \cdots Q_{k+1}$ and using orthogonality we obtain the relation

$$AV_k = U_{k+1} B_k = \hat{B}_k + \beta_{k+1} v_{k+1} e_k^T, \quad k = 1 : n, \quad (8.4.49)$$

where

$$\begin{aligned} P_1 P_2 \cdots P_k \begin{pmatrix} I_k \\ 0 \end{pmatrix} &= V_k = (v_1, \dots, v_k), \\ Q_1 Q_2 \cdots Q_{k+1} \begin{pmatrix} I_{k+1} \\ 0 \end{pmatrix} &= U_{k+1} = (u_1, \dots, u_{k+1}). \end{aligned}$$

If we consider the intermediate result after applying also P_{k+1} the first $k+1$ rows have been transformed into bidiagonal form, i.e.

$$(I_{k+1} \ 0) Q_{k+1} \cdots Q_2 Q_1 A P_1 P_2 \cdots P_{k+1} = (B_k \ \alpha_{k+1} e_{k+1}) (I_{k+1} \ 0).$$

Transposing this gives a second relation

$$U_{k+1}^T A = B_k V_k^T + \alpha_{k+1} e_{k+1} v_{k+1}^T, \quad (8.4.50)$$

An nice feature of PLS is that if a zero element occurs in B , then the algorithm stops with an exact least squares solution. As an example, consider the case below, where the first two columns of B have been computed ($m=7, n=5$).

$$Q_3 Q_2 Q_1 (b \ A) P_1 P_2 = \left(\begin{array}{c|cc|ccc} \beta_1 & \alpha_1 & & & & & \\ & \beta_2 & \alpha_2 & & & & \\ & & \beta_3 & \times & \otimes & \otimes & \\ \hline & & & \times & \times & \times & \\ & & & \otimes & \times & \times & \\ & & & \otimes & \times & \times & \\ & & & \otimes & \times & \times & \end{array} \right),$$

In the next step elements in the third row are zeroed out and α_3 determined. If $\alpha_3 = 0$, then the problem decomposes and an overdetermined core system of dimension 3×2 can be extracted. If $\alpha_3 \neq 0$, then elements in the third column of the transformed matrix A are zeroed and β_3 determined. If $\beta_4 = 0$, then the problem decomposes and a square core system of dimension 3×3 can be extracted.

In general, assume that the first zero element to occur is $\alpha_{k+1} = 0$. Then we have obtained the decomposition

$$\tilde{U}_{k+1}^T A \tilde{V}_k = \begin{pmatrix} B_k & 0 \\ 0 & A_k \end{pmatrix},$$

where $A_k \in \mathbf{R}^{(m-k-1) \times (n-k)}$, and

$$\tilde{U}_{k+1} = Q_{k+1} \cdots Q_2 Q_1, \quad \tilde{V}_k = P_1 P_2 \cdots P_k,$$

are square orthogonal matrices. Then, setting $x = \tilde{V}_k y$, the transformed least squares problem takes the form

$$\min_y \left\| \begin{pmatrix} B_k & 0 \\ 0 & A_k \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} \beta_1 e_1 \\ 0 \end{pmatrix} \right\|_2, \quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad (8.4.51)$$

$y_1 \in \mathbf{R}^k, y_2 \in \mathbf{R}^{n-k}$. This problem is separable and decomposes into two independent subproblems

$$\min_{y_1} \|B_k y_1 - \beta_1 e_1\|_2, \quad \min_{y_2} \|A_k y_2\|_2. \quad (8.4.52)$$

By construction B_k has nonzero elements in its two diagonals. Thus, it has full column rank and the solution y_1 to the first subproblem is unique. Further, the

minimum norm solution of the initial problem is obtained simply by taking $y_2 = 0$. The first subproblem (8.4.52) is called a **core subproblem**. It can be solved by QR factorization exactly as outlined for the full system when $k = n$.

When $\beta_{k+1} = 0$ is the first zero element to occur, then the reduced problem has a separable form similar to (8.4.52). The core subproblem is

$$\hat{B}_k y_1 = \beta_1 e_1, \quad \hat{B}_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ \ddots & \ddots & \ddots & \\ & \beta_k & \alpha_k & \end{pmatrix} \in \mathbf{R}^{k \times k}. \quad (8.4.53)$$

In the k th step the k th column in L is computed and the k th columns in U and V determined

$$u_k = Q_1 \cdots Q_k e_k, \quad v_k = P_1 \cdots P_k e_k,$$

If $\alpha_k \beta_{k+1} = 0$, the problem decomposes and the reduction can be terminated

Paige and Strakoš [471] have shown the following important properties of the core subproblem obtained by the bidiagonalization algorithm:

Theorem 8.4.3.

Assume that the bidiagonalization of $(b \ A)$ terminates prematurely with $\alpha_k = 0$ or $\beta_{k+1} = 0$. Then the core corresponding subproblem (8.4.52) or (8.4.53) is minimally dimensioned. Further, the singular values of the core matrix B_k or \hat{B}_k , are simple and the right hand side βe_1 has nonzero components in along each left singular vector.

Proof. Sketch: The minimal dimension is a consequence of the uniqueness of the decomposition (8.4.41), as long as no zero element in B appears. That the matrix \hat{B}_k has simple singular values follows from the fact that all subdiagonal elements are nonzero. The same is true for the square bidiagonal matrix $(B_k \ 0)$ and therefore also for B_k . Finally, if βe_1 did not have nonzero components along a left singular vector, then the reduction must have terminated earlier. For a complete proof we refer to [471].) \square

We remark that the solution steps can be interleaved with the reduction to bidiagonal form. This makes it possible to compute a sequence of *approximate solutions* $x_k = P_1 P_2 \cdots P_k y_k$, where $y_k \in \mathbf{R}^k$ solves

$$\min_y \|\beta_1 e_1 - B_k y\|_2, \quad k = 1, 2, 3, \dots \quad (8.4.54)$$

After each (double) step in the bidiagonalization we advance the QR decomposition of B_k . The norm of the least squares residual corresponding to x_k is then given by

$$\|b - Ax_k\|_2 = |\bar{\phi}_{k+1}|.$$

The sequence of residual norms is nonincreasing. We stop and accept $x = V_k y_k$ as an approximate solution of the original least squares problem if this residual is sufficiently small.

The PLS algorithm often requires a lower dimensional subspace than TSVD to achieve similar accuracy. A heuristic explanation for this observation is that the TSVD subspaces do not depend on b and are equally suitable for any right-hand side. The Krylov subspaces, on the other hand, are tailored to the particular right-hand side b of the problem; see Hanke [306].

The PLS method can also be implemented using a Lanczos-type algorithm for bidiagonalization. This algorithm, called LSQR, Paige and Saunders [470, 469] is suitable for solving *sparse* linear least squares. A number of important properties of the successive approximations x_k in PLS are best discussed in connection with LSQR, see Algorithm 11.3.4.

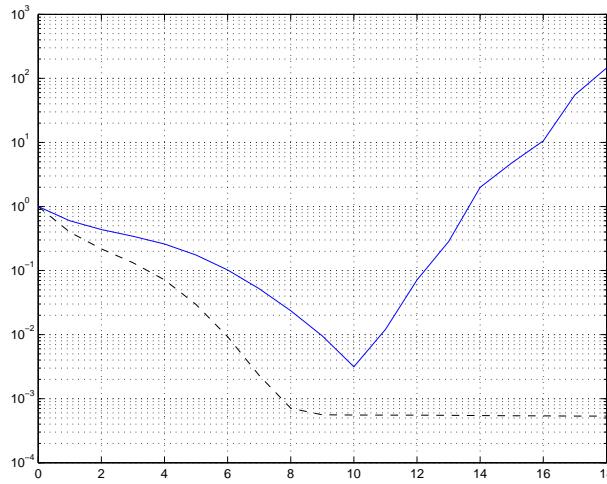


Figure 8.4.2. Relative error $\|x_k - x\|_2 / \|x\|_2$ and residual $\|Ax_k - b\|_2 / \|b\|_2$ for PLS solutions after k steps.

Example 8.4.2.

Consider the ill-posed linear system $Ax = b$, $A \in \mathbf{R}^{n \times n}$ in Example 8.4.1. Figure 8.4.2 shows the relative error $\|x_k - x\|_2 / \|x\|_2$ and residual $\|Ax_k - b\|_2 / \|b\|_2$ after k steps of PLS. For this problem the results are almost identical for PLS and TSVD. Note that in PLS only a partial bidiagonalization of A and generation of the matrix V_k is needed.

The PLS method can also be implemented using a Lanczos-type algorithm for bidiagonalization. This algorithm, called LSQR, is suitable for solving *sparse* linear least squares. A number of important properties of the successive approximations x_k in PLS are best discussed in connection with LSQR; see Section 10.7.4. The Householder algorithm described above is to be preferred for dense problems, because of its superior numerical stability.

Review Questions

- 4.1** When and why should column pivoting be used in computing the QR factorization of a matrix? What inequalities will be satisfied by the elements of R if the standard column pivoting strategy is used?
- 4.2** Show that the singular values and condition number of R equal those of A . Give a simple lower bound for the condition number of A in terms of its diagonal elements. Is it advisable to use this bound when no column pivoting has been performed?
- 4.3** Give a simple lower bound for the condition number of A in terms of the diagonal elements of R . Is it advisable to use this bound when no column pivoting has been performed?
- 4.4** What is meant by a Rank-revealing QR factorization? Does such a factorization always exist?
- 4.5** How is the *numerical* rank of a matrix A defined? Give an example where the numerical rank is not well determined.

Problems

- 4.1** (a) Describe how the QR factorizations of a matrix of the form

$$\begin{pmatrix} A \\ \mu D \end{pmatrix}, \quad A \in \mathbf{R}^{m \times n},$$

where $D \in \mathbf{R}^{n \times n}$ is diagonal, can be computed using Householder transformations in mn^2 flops.

(b) Estimate the number of flops that are needed for the reduction using Householder transformations in the special case that $A = R$ is upper triangular? Devise a method using Givens rotations for this special case!

Hint: In the Givens method zero one diagonal at a time in R working from the main diagonal inwards.

- 4.2** Let the vector v , $\|v\|_2 = 1$, satisfy $\|Av\|_2 = \epsilon$, and let Π be a permutation such that

$$|w_n| = \|w\|_\infty, \quad \Pi^T v = w.$$

- (a) Show that if R is the R factor of $A\Pi$, then $|r_{nn}| \leq n^{1/2}\epsilon$.

Hint: Show that $\epsilon = \|Rw\|_2 \geq |r_{nn}w_n|$ and then use the inequality $|w_n| = \|w\|_\infty \geq n^{-1/2}\|w\|_2$.

- (b) Show using (a) that if $v = v_n$, the right singular vector corresponding to the smallest singular value $\sigma_n(A)$, then

$$\sigma_n(A) \geq n^{-1/2}|r_{nn}|.$$

4.3 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 Π to *maximize* the $(1,1)$ element in the QR factorization of $R\Pi$. Show that this is achieved by taking $\Pi = I$ if $|a| \geq \sqrt{b^2 + c^2}$, else $\Pi = \Pi_{12}$, where Π_{12} 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 $\Pi = I$ if $|c^{-1}| \geq \sqrt{a^{-2} + b^2(ac)^{-2}}$ else $\Pi = \Pi_{12}$. Hence, the test compares *row* norms in R^{-1} instead of *column* norms in R .
- 4.6** To minimize $\|x\|_2$ is not always a good way to resolve rank deficiency, and therefore the following generalization of problem (8.4.5) is often useful: For a given matrix $B \in \mathbf{R}^{p \times n}$ consider the problem

$$\min_{x \in S} \|Bx\|_2, \quad S = \{x \in \mathbf{R}^n \mid \|Ax - b\|_2 = \min\}.$$

(a) Show that this problem is equivalent to

$$\min_{x_2} \|(BC)x_2 - (Bd)\|_2,$$

where C and d are defined by (8.4.4).

(b) Often one wants to choose B so that $\|Bx\|_2$ is a measure of the smoothness of the solution x . For example one can take B to be a discrete approximation to the second derivative operator,

$$B = \begin{pmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{pmatrix} \in \mathbf{R}^{(n-2) \times n}.$$

Show that provided that $\mathcal{N}(A) \cap \mathcal{N}(B) = \emptyset$ this problem has a unique solution, and give a basis for $\mathcal{N}(B)$.

4.5 Let $A \in \mathbf{R}^{m \times n}$ with $\text{rank}(A) = r$. A rank revealing LU factorizations of 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 permutation matrices and $L_{11}, U_{11} \in \mathbf{R}^{r \times r}$ are triangular and nonsingular can also be used to compute pseudo-inverse solutions $x = A^\dagger b$. Show, using Theorem 8.1.12 that

$$A^\dagger = \Pi_2 \begin{pmatrix} I_r & S \end{pmatrix}^\dagger U_{11}^{-1} L_{11}^{-1} \begin{pmatrix} I_r \\ T \end{pmatrix}^\dagger \Pi_1,$$

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 $r = m$.)

4.6 Consider the block upper-bidiagonal matrix

$$A = \begin{pmatrix} B_1 & C_1 & & \\ & B_2 & C_2 & \\ & & B_3 & \end{pmatrix}$$

Outline an algorithm for computing the QR factorization of A , which treats one block row at a time. (It can be assumed that A has full column rank.) Generalize the algorithm to an arbitrary number of block rows!

4.7 (a) Suppose that we have computed the pivoted QR factorization of A ,

$$Q^T A \Pi I = \begin{pmatrix} R \\ 0 \end{pmatrix} \in \mathbf{R}^{m \times n},$$

of a matrix $A \in \mathbf{R}^{m \times n}$. Show that by postmultiplying the *upper* triangular matrix R by a sequence of Householder transformations we can transform R into a *lower* triangular matrix $L = RP \in \mathbf{R}^{n \times n}$ and that by combining these two factorizations we obtain

$$Q^T A \Pi I P = \begin{pmatrix} L \\ 0 \end{pmatrix}. \quad (8.4.55)$$

Comment: This factorization, introduced by G. W. Stewart, is equivalent to one step of the basic unshifted QR–SVD algorithm; see Section 10.4.1.

(b) Show that the total cost for computing the QLP decomposition is roughly $2mn^2 + 2n^3/3$ flops. How does that compare with the cost for computing the bidiagonal decomposition of A ?

(c) Show that the two factorizations can be interleaved. What is the cost for performing the first k steps?

4.8 Work out the details of an algorithm for transforming a matrix $A \in \mathbf{R}^{m \times n}$ to *lower* bidiagonal form. Consider both cases when $m > n$ and $m \leq n$.

Hint: It can be derived by applying the algorithm for transformation to upper bidiagonal form to A^T .

4.9 Trefethen and Bau [575, pp. 237–238] have suggested a blend of the Golub–Kahan and Chan methods for bidiagonal reduction 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 $(m - k)/(n - k) = 2$. What is the resulting operation count?

8.5 Structured and Sparse Least Squares Problems

8.5.1 Banded Least Squares Problems

We now consider orthogonalization methods for the special case when A is a banded matrix of row bandwidth w , see Definition 8.2.2. From Theorem 8.2.3 we know that

$A^T A$ will be a symmetric band matrix with only the first $r = w - 1$ super-diagonals nonzero. Since the factor R in the QR factorization equals the unique Cholesky factor of $A^T A$ it will have only w nonzeros in each row.

Even though the *final* factor R is independent of the row ordering in A , the intermediate fill will vary. For banded rectangular matrices the QR factorization can be obtained efficiently by sorting the rows of A and suitably subdividing the Householder transformations. The rows of A should be sorted by leading entry order (i.e., increasing minimum column subscript order) That is, if $f_i, i = 1 : m$ denotes the column indices of the first nonzero element in row i we should have,

$$i \leq k \Rightarrow f_i \leq f_k.$$

Such a band matrix can then be written as

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_q \end{pmatrix}, \quad q \leq n,$$

is said to be in **standard form**. where in block A_i the first nonzero element of each row is in column i . The Householder QR process is then applied to the matrix in q major steps. In the first step a QR factorization of the first block A_1 is computed, yielding R_1 . Next at step k , $k = 2 : q$, R_{k-1} will be merged with A_k yielding

$$Q_k^T \begin{pmatrix} R_{k-1} \\ A_k \end{pmatrix} = R_k.$$

Since the rows of block A_k has their first nonzero elements in column k , the first $k - 1$ rows of R_{k-1} will not be affected. The matrix Q can be implicitly represented in terms of the Householder vectors of the factorization of the subblocks. This sequential Householder algorithm, which is also described in [399, Ch. 27], requires $(m + 3n/2)w(w + 1)$ multiplications or about twice the work of the less stable Cholesky approach. For a detailed description of this algorithm, see Lawson and Hanson [399, Ch. 11].

In Volume I, Section 4.4.4 we considered the interpolation of a function f with a linear combination of $m + k$ cubic B-splines k on the grid $\Delta = \{x_0 < x_1 < \dots < x_m\}$. If function values $f_j = f(\tau_j)$, are given at distinct points $\tau_1 < \tau_2 < \dots < \tau_n$, where $n \geq m + k$. This leads to a least squares approximation problem

$$\min \sum_{j=1}^n e_j^2, \quad e_j = w_j \left(f_j - \sum_{i=-k}^{m-1} c_i B_{i,k+1}(\tau_j) \right). \quad (8.5.1)$$

where w_j are positive weights. This is an overdetermined linear system for c_i , $i = -k, \dots, m-1$. The elements of its coefficient matrix $B_{i,k+1}(\tau_j)$ can be evaluated by the recurrence (4.6.19). The coefficient matrix has a band structure since in the j th row the i th element will be zero if $\tau_j \notin [x_i, x_{i+k+1}]$. It can be shown, see de Boor [1978, p. 200], that the coefficient matrix will have full rank equal to $m + k$ if and only if there is a subset of points τ_j satisfying

$$x_{j-k-1} < \tau_j < x_j, \quad \forall j = 1 : m + k. \quad (8.5.2)$$

Example 8.5.1.

The least squares approximation of a discrete set of data by a linear combination of cubic B-splines gives rise to a banded linear least squares problem. Let

$$s(t) = \sum_{j=1}^n x_j B_j(t),$$

where $B_j(t)$, $j = 1 : n$ are the normalized cubic B-splines, and let (y_i, t_i) , $i = 1 : m$ be given data points. If we determine x to minimize

$$\sum_{i=1}^m (s(t_i) - y_i)^2 = \|Ax - y\|_2^2,$$

then A will be a banded matrix with $w = 4$. In particular, if $m = 13$, $n = 8$ the matrix may have the form shown in Figure 8.5.1. Here A consists of blocks A_k^T , $k = 1 : 7$. In the Figure 8.5.1 we also show the matrix after the first three blocks have been reduced by Householder transformations P_1, \dots, P_9 . Elements which have been zeroed by P_j are denoted by \circ and fill elements by $+$. In step $k = 4$ only the indicated part of the matrix is involved.

$$\left(\begin{array}{ccccccccc} \times & \times & \times & \times & & & & & \\ 1 & \times & \times & \times & + & & & & \\ 1 & 2 & \times & \times & + & + & & & \\ 3 & 4 & \times & \times & + & & & & \\ 3 & 4 & 5 & \times & + & & & & \\ 6 & 7 & 8 & \times & & & & & \\ 6 & 7 & 8 & 9 & & & & & \\ 6 & 7 & 8 & 9 & & & & & \\ & \times & \times & \times & \times & & & & \\ & \times & \times & \times & \times & & & & \\ & & \times & \times & \times & \times & & & \\ & & \times & \times & \times & \times & & & \\ & & \times & \times & \times & \times & & & \\ & & & & & & & & \end{array} \right)$$

Figure 8.5.1. The QR factorization of a banded rectangular matrix A .

In the algorithm the Householder transformations can also be applied to one or several right hand sides b to produce

$$c = Q^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad c_1 \in \mathbf{R}^n.$$

The least squares solution is then obtained from $Rx = c_1$ by back-substitution. The vector c_2 is not stored but used to accumulate the residual sum of squares $\|r\|_2^2 = \|c_2\|_2^2$.

It is also possible to perform the QR factorization by treating one row at a time using Givens' rotations. Each step then is equivalent to updating a full

triangular matrix formed by columns $f_i(A)$ to $l_i(A)$. Further, if the matrix A is in standard form the first $f_i(A)$ rows of R are already finished at this stage. The reader is encouraged to work through Example 8.5.1 below in order to understand how the algorithm proceeds!

A special form of banded system that occurs in many applications is the **block-bidiagonal form**,

$$\begin{pmatrix} A_1 & B_1 & & \\ & A_2 & B_2 & \\ & & \ddots & \\ & & & B_{n-1} \\ & & \ddots & \\ & & & A_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}, \quad (8.5.3)$$

Assume that by orthogonal transformations the k first block rows have been reduced to the upper triangular form

$$\begin{pmatrix} R_1 & S_1 & & \\ & R_2 & S_2 & \\ & & \ddots & \\ & & & R_{k-1} & S_{k-1} \\ & & & & R_k \end{pmatrix}. \quad (8.5.4)$$

The last diagonal block here is the only one to be altered in the next step. In step $k+1$ the blocks A_{k+1} and B_{k+1} are added to the matrix together with observations b_{k+1} . We can now choose an orthogonal transformations that performs the reduction to upper triangular form

$$Q_k^T \begin{pmatrix} \bar{R}_k & 0 \\ A_{k+1} & B_{k+1} \end{pmatrix} = \begin{pmatrix} R_k & S_k \\ 0 & \bar{R}_{k+1} \end{pmatrix}.$$

The above structure arises in a least squares formulation of the Kalman filter, which is a popular method used to estimate the behavior of certain linear dynamic systems; see Paige and Saunders [466] for details.

8.5.2 Blocked Form of QR Factorization

In many least squares problems the unknowns x can be naturally partitioned into two groups with n_1 and n_2 components, respectively. Then the problem has the form

$$\min_{x_1, x_2} \| (A_1 \ A_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - b \|_2, \quad (8.5.5)$$

where $A = (A_1 \ A_2) \in \mathbf{R}^{m \times n}$ and $n = n_1 + n_2$. For example, in separable nonlinear least squares problems subproblems of the form (8.5.5) arise, see Section 9.2.4.

Assume that the matrix A has full column rank and let $P_{\mathcal{R}(A_1)}$ be the orthogonal projection onto $\mathcal{R}(A_1)$. For any x_2 we can split the vector $b - A_2 x_2 = r_1 + r_2$

into two orthogonal components

$$r_1 = P_{\mathcal{R}(A_1)}(b - A_2x_2), \quad r_2 = (I - P_{\mathcal{R}(A_1)})(b - A_2x_2).$$

Then the problem (8.5.5) takes the form

$$\min_{x_1, x_2} \left\| (A_1x_1 - r_1) - P_{\mathcal{N}(A_1^T)}(b - A_2x_2) \right\|_2. \quad (8.5.6)$$

Here, since $r_1 \in \mathcal{R}(A_1)$ the variables x_1 can always be chosen so that $A_1x_1 - r_1 = 0$. It follows that in the least squares solution to (8.5.5) x_2 is the solution to the reduced least squares problem

$$\min_{x_2} \|P_{\mathcal{N}(A_1^T)}(A_2x_2 - b)\|_2. \quad (8.5.7)$$

where the variables x_1 have been eliminated. When this reduced problem has been solved for x_2 then x_1 can be computed from the least squares problem

$$\min_{x_1} \|A_1x_1 - (b - A_2x_2)\|_2. \quad (8.5.8)$$

Sometimes it may be advantageous to carry out a *partial* QR factorization, where only the $k = n_1 < n$ columns of A are reduced. Using Householder QR

$$Q_1^T(A b) = \begin{pmatrix} R_{11} & \tilde{A}_{12} & \tilde{b}_1 \\ 0 & \tilde{A}_{22} & \tilde{b}_2 \end{pmatrix},$$

with R_{11} nonsingular. Then x_2 is the solution to the reduced least squares problem

$$\min_{x_2} \|\tilde{b}_2 - \tilde{A}_{22}x_2\|_2,$$

If this is again solved by Householder QR nothing new comes out—we have only emphasized a new aspect of this algorithm. However, it may be advantageous to use another method for the reduced problem.

In some applications, e.g., when A has block angular structure (see Section 8.5.3), it may be preferable instead not to save R_{11} and R_{12} and instead to refactorize A_1 and solve (8.5.8) for x_1 .

If we use perform n_1 steps of MGS on the full problem, this yields the partial factorization

$$(A b) = (Q_1 \quad \tilde{A}_2 \quad \tilde{b}) \begin{pmatrix} R_{11} & R_{12} & z_1 \\ 0 & I & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

where R_{11} is nonsingular. The residual is decomposed as $r = r_1 + r_2$, $r_1 \perp r_2$, where

$$r_1 = Q_1(z_1 - R_{12}x_2 - R_{11}x_1), \quad r_2 = \tilde{b}_1 - \tilde{A}_2x_2.$$

Then x_2 is the solution to the reduced least squares problem $\min_{x_2} \|\tilde{b}_1 - \tilde{A}_2x_2\|_2$. With x_2 known x_1 can be computed by back-substitution from $R_{11}x_1 = z_1 - R_{12}x_2$.

To obtain near-peak performance for large dense matrix computations on current computing architectures requires code that is dominated by matrix-matrix

operations since these involve less data movement per floating point computation. The QR factorization should therefore be organized in partitioned or blocked form, where the operations have been reordered and grouped into matrix operations.

For the QR factorization $A \in \mathbf{R}^{m \times n}$ ($m \geq n$) is partitioned as

$$A = (A_1, A_2), \quad A_1 \in \mathbf{R}^{m \times nb}, \quad (8.5.9)$$

where nb is a suitable block size and the QR factorization

$$Q_1^T A_1 = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \quad Q_1 = P_1 P_2 \cdots P_{nb}, \quad (8.5.10)$$

is computed, where $P_i = I - u_i u_i^T$ are Householder reflections. Then the remaining columns A_2 are updated

$$Q_1^T A_2 = Q_1^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} = \begin{pmatrix} R_{12} \\ \tilde{A}_{22} \end{pmatrix}. \quad (8.5.11)$$

In the next step we partition $\tilde{A}_{22} = (B_1, B_2)$, and compute the QR factorization of $B_1 \in \mathbf{R}^{(m-r) \times r}$. Then B_2 is updated as above, and we continue in this way until the columns in A are exhausted.

A major part of the computation is spent in the updating step (8.5.11). As written this step cannot use BLAS-3, which slows down the execution. To achieve better performance it is essential that this part is sped up. The solution is to aggregate the Householder transformations so that their application can be expressed as matrix operations. For use in the next subsection, we give a slightly more general result.

Lemma 8.5.1.

Let P_1, P_2, \dots, P_r be a sequence of Householder transformations. 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 $T_1, T_2 \in \mathbf{R}^{r \times r}$ are upper triangular matrices. Then for the product $Q_1 Q_2$ we have

$$Q = Q_1 Q_2 = (I - Y_1 T_1 Y_1^T)(I - Y_2 T_2 Y_2^T) = (I - Y T Y^T) \quad (8.5.12)$$

where

$$\hat{Y} = (Y_1, Y_2), \quad \hat{T} = \begin{pmatrix} T_1 & -(T_1 Y_1^T)(Y_2 T_2) \\ 0 & T_2 \end{pmatrix}. \quad (8.5.13)$$

Note that Y is formed by concatenation, but computing the off-diagonal block in T requires extra operations.

For the partitioned algorithm we use the special case when $r_2 = 1$ to aggregate the Householder transformations for each processed block. Starting with $Q_1 = I - \tau_1 u_1 u_1^T$, we set $Y = u_1$, $T = \tau_1$ and update

$$Y := (Y, u_{k+1}), \quad T := \begin{pmatrix} T & -\tau_k T Y^T u_k \\ 0 & \tau_k \end{pmatrix}, \quad k = 2 : nb. \quad (8.5.14)$$

Note that Y will have a trapezoidal form and thus the matrices Y and R can overwrite the matrix A . With the representation $Q = (I - YTY^T)$ the updating of A_2 becomes

$$B = Q_1^T A = (I - YT^T Y^T) A_2 = A_2 - YT^T Y^T A_2,$$

which now involves only matrix operations. This partitioned algorithm requires more storage and operations than the point algorithm, namely those needed to produce and store the T matrices. However, for large matrices this is more than offset by the increased rate of execution.

As mentioned in Chapter 7 recursive algorithms can be developed into highly efficient algorithms for high performance computers and are an alternative to the currently more used partitioned algorithms by LAPACK. The reason for this is that recursion leads to automatic variable blocking that dynamically adjusts to an arbitrary number of levels of memory hierarchy.

Consider the partitioned QR factorization

$$A = \begin{pmatrix} A_1 & A_2 \end{pmatrix} = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

where Let A_1 consist of the first $\lfloor n/2 \rfloor$ columns of A . To develop a recursive algorithm we start with a QR factorization of A_1 and update the remaining part A_2 of the matrix,

$$Q_1^T A_1 = \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}, \quad Q_1^T A_2 = Q_1^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} = \begin{pmatrix} R_{12} \\ \tilde{A}_{22} \end{pmatrix}.$$

Next \tilde{A}_{22} is recursively QR decomposed giving Q_2 , R_{22} , and $Q = Q_1 Q_2$.

As an illustration we give below a simple implementation in Matlab, which is convenient to use since it allows for the definition of recursive functions.

```
function [Y,T,R] = recqr(A)
%
% RECQR computes the QR factorization of the m by n matrix A,
% (m >= n). Output is the n by n triangular factor R, and
% Q = (I - YTY') represented in aggregated form, where Y is
% m by n and unit lower trapezoidal, and T is n by n upper
% triangular
[m,n] = size(A);
if n == 1
    [Y,T,R] = house(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));
    Y=[Y1;Y2];
    T=[T1;T2];
    R=[R1;R2];
end
```

```

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
%

```

The algorithm uses the function `house(a)` to compute a Householder transformation $P = I - \tau uu^T$ such that $Pa = \sigma e_1$, $\sigma = -\text{sign}(a_1)\|a\|_2$. A serious defect of this algorithm is the overhead in storage and operations caused by the T matrices. In the partitioned algorithm n/nb T -matrices of size nb we formed and stored, giving a storage overhead of $\frac{1}{2}n \cdot nb$. In the recursive QR algorithm in the end a T -matrix of size $n \times n$ is formed and stored, leading to a much too large storage and operation overhead. Therefore, a better solution is to use a hybrid between the partitioned and the recursive algorithm, where the recursive QR algorithm is used to factorize the blocks in the partitioned algorithm.

8.5.3 Block Angular Least Squares Problems

There is often a substantial similarity in the structure of many large scale sparse least squares problems. In particular, the problem can often be put in the following bordered block diagonal or **block angular form**:

$$A = \begin{pmatrix} A_1 & & & \left| B_1 \right. \\ & A_2 & & \left| B_2 \right. \\ & & \ddots & \vdots \\ & & & A_M \left| B_M \right. \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \\ x_{M+1} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix}, \quad (8.5.15)$$

where

$$A_i \in \mathbf{R}^{m_i \times n_i}, \quad B_i \in \mathbf{R}^{m_i \times n_{M+1}}, \quad i = 1 : M,$$

and

$$m = m_1 + m_2 + \cdots + m_M, \quad n = n_1 + n_2 + \cdots + n_{M+1}.$$

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 problem. Applications where the form (8.5.15) arises naturally include photogrammetry, Doppler radar and 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 [122].

The normal matrix of (8.5.15) has a doubly bordered block diagonal, or ar-

rowhead, form,

$$A^T A = \left(\begin{array}{ccc|c} A_1^T A_1 & & & A_1^T B_1 \\ & A_2^T A_2 & & A_2^T B_2 \\ & & \ddots & \vdots \\ & & & A_M^T A_M & A_M^T B_M \\ \hline B_1^T A_1 & B_2^T A_2 & \cdots & B_M^T A_M & C \end{array} \right),$$

where

$$C = \sum_{k=1}^M B_k^T B_k.$$

We assume in the following that $\text{rank}(A) = n$, which implies that the Cholesky factor R of $A^T A$ is nonsingular. It will have a block structure similar to that of A ,

$$R = \left(\begin{array}{ccc|c} R_1 & & & S_1 \\ & R_2 & & S_2 \\ & & \ddots & \vdots \\ & & & R_M & S_M \\ \hline & & & & R_{M+1} \end{array} \right), \quad (8.5.16)$$

Identifying the blocks in the relation $R^T R = A^T A$ we get

$$\begin{aligned} R_i^T R_i &= A_i^T A_i, & R_i^T S_i &= A_i^T B_i, & i &= 1 : M, \\ R_{M+1}^T R_{M+1} + \sum_{i=1}^M S_i^T S_i &= C. \end{aligned}$$

Hence, $R_i \in \mathbf{R}^{n_i \times n_i}$, $i = 1 : M$, are the Cholesky factors of $A_i^T A_i$ and

$$S_i = R_i^{-T} (A_i^T B_i), \quad i = 1 : M.$$

Finally, R_{M+1} is obtained as the Cholesky factor of

$$\tilde{C} = C - \sum_{i=1}^M S_i^T S_i.$$

The matrix R in (8.5.16) can also be computed by a sequence of QR factorizations. The following algorithm solves least squares problems of block angular form based using QR factorization. It 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

$$Q_i^T(A_i, B_i) = \begin{pmatrix} R_i & S_i \\ 0 & T_i \end{pmatrix}, \quad Q_i^T b_i = \begin{pmatrix} c_i \\ d_i \end{pmatrix}.$$

where R_i is upper triangular.

(b) Apply orthogonal transformations to update

$$\begin{pmatrix} R_{M+1} & c_{M+1} \\ T_i & d_i \end{pmatrix} \text{ to yield } \begin{pmatrix} R_{M+1} & c_{M+1} \\ 0 & e_i \end{pmatrix}$$

(c) Update the residual norm $\rho = (\rho^2 + \|e_i\|_2^2)^{1/2}$.

3. Compute x_{M+1} from the triangular system

$$R_{M+1}x_{M+1} = c_{M+1},$$

4. For $i = 1 : M$ compute x_i by back-substitution in the triangular systems

$$R_i x_i = c_i - S_i x_{M+1}.$$

In steps 2 (a) and 4 the computations can be performed in parallel on the M subsystems. 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 if 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 increase the operation count by a few percent.

Using the structure of the R -factor in (8.5.15), the diagonal blocks of the variance-covariance matrix $C = (R^T R)^{-1} = R^{-1} R^{-T}$ can be written

$$\begin{aligned} C_{M+1,M+1} &= R_{M+1}^{-1} R_{M+1}^{-T}, \\ C_{i,i} &= R_i^{-1} (I + W_i^T W_i) R_i^{-T}, \quad W_i^T = S_i R_{M+1}^{-1}, \quad i = 1, \dots, M. \end{aligned} \quad (8.5.17)$$

If we compute the QR factorizations

$$Q_i \begin{pmatrix} W_i \\ I \end{pmatrix} = \begin{pmatrix} U_i \\ 0 \end{pmatrix}, \quad i = 1, \dots, M,$$

we have $I + W_i^T W_i = U_i^T U_i$ and then

$$C_{i,i} = (U_i R_i^{-T})^T (U_i R_i^{-T}), \quad i = 1, \dots, M.$$

This assumes that all the matrices R_i and S_i have been retained.

A procedure, called substructuring or dissection can often be used for obtaining a block angular form of a problem. The idea is similar but preceeded the

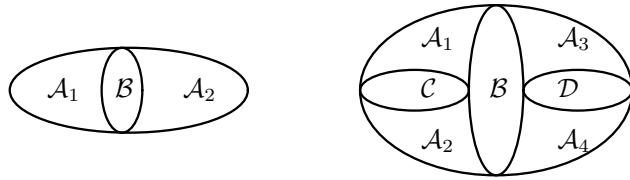


Figure 8.5.2. One and two levels of dissection of a region.

nested dissection method presented in Section 7.7.5. Such a technique dates back to Helmert [316, 1880], who used it for breaking down geodetic problems into geographically defined subproblems. Consider a geodetic position network consisting of geodetic stations connected through observations. To each station corresponds a set of unknown coordinates to be determined. The idea is to choose a set of stations \mathcal{B} , which separates the other stations into two regional blocks \mathcal{A}_1 and \mathcal{A}_2 so that station variables in \mathcal{A}_1 are not connected by observations to station variables in \mathcal{A}_2 . The variables are then ordered so that those in \mathcal{A}_1 appear first, those in \mathcal{A}_2 second, and those in \mathcal{B} last. The equations are ordered so that those including \mathcal{A}_1 come first, those including \mathcal{A}_2 next, and those only involving variables in \mathcal{B} come last. The dissection can be continued by dissecting the regions \mathcal{A}_1 and \mathcal{A}_2 each 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 Figure 8.5.1. The corresponding block structure induced in the matrix are

$$A = \left(\begin{array}{cc|cc} A_1 & & B_1 \\ & A_2 & B_2 \end{array} \right), \quad A = \left(\begin{array}{ccc|cc} A_1 & & & C_1 & B_1 \\ & A_2 & & C_2 & B_2 \\ & & A_3 & D_3 & B_3 \\ & & & D_4 & B_4 \end{array} \right),$$

The block of rows corresponding to \mathcal{A}_i , $i = 1, 2, \dots$, can be processed independently. The remaining variables are then eliminated, etc.; compare the block angular structure in (8.5.15). 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 of the dissection the number of variables in the two partitions are roughly the same. Also, the number of variables in the separator nodes should be as small as possible. If each dissection is done so that the variables contained in the two partitions are at least halved, then after at most $\log^2 n$ levels each partition contains only one variable. Of course, it is usually preferable to stop before this point.

8.5.4 Block Triangular Form

In Section 7.7.7 we showed that before solving a sparse linear system $Ax = b$ it was advantageous to permute the matrix into block triangular form. A similar preprocessing can be done for an arbitrary rectangular matrix $A \in \mathbf{R}^{m \times n}$, $m \geq n$. By row and column permutations the matrix can be brought into the block

triangular form

$$PAQ = \begin{pmatrix} A_h & U_{hs} & U_{hv} \\ & A_s & U_{sv} \\ & & A_v \end{pmatrix}. \quad (8.5.18)$$

Here the diagonal block A_h is underdetermined (i.e., has more columns than rows), A_s is square and A_v is overdetermined (has more rows than columns), and all three blocks have a nonzero diagonal; see the example in Figure 8.5.2. The submatrices A_v and A_h^T both have the strong Hall property. The off-diagonal blocks are possibly nonzero matrices of appropriate dimensions. The block decomposition (8.5.18) of A is called the **coarse decomposition** of A . One or two of the diagonal blocks may be absent in the coarse decomposition. It may be possible to further decompose the

$\times \times \otimes \times \times$	\times	
$\otimes \times$	$\times \times$	\times
$\times \times \otimes$		
	$\otimes \times$	\times
	$\times \otimes \times$	
	$\otimes \times$	
	$\times \otimes \times$	\times
		$\otimes \times$
		\otimes
		\times
		$\times \times$
		\times

Figure 8.5.3. The coarse block triangular decomposition of A .

diagonal blocks in (8.5.18) to obtain the **fine decompositions** of these submatrices. Each of the blocks A_h and A_v may be further decomposable into block diagonal form,

$$A_h = \begin{pmatrix} A_{h1} & & \\ & \ddots & \\ & & A_{hp} \end{pmatrix}, \quad A_v = \begin{pmatrix} A_{v1} & & \\ & \ddots & \\ & & A_{vq} \end{pmatrix},$$

where each A_{h1}, \dots, A_{hp} is underdetermined and each A_{v1}, \dots, A_{vq} is overdetermined. The square submatrix A_s may be decomposable in block upper triangular form

$$A_s = \begin{pmatrix} A_{s1} & U_{12} & \dots & U_{1,t} \\ & A_{s2} & \dots & U_{2,t} \\ & & \ddots & \vdots \\ & & & A_{st} \end{pmatrix} \quad (8.5.19)$$

where the square diagonal blocks A_{s1}, \dots, A_{st} have nonzero diagonal elements. The resulting Dulmage–Mendelsohn decomposition can be shown to be essentially

unique. Any one 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.

An algorithm for the more general block triangular form described above due to Pothen and Fan [490], uses the concept of matchings in bipartite graphs. The algorithm 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 submatrix 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 directed from columns to rows.

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 (8.5.18) 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

$$\min_{\tilde{x}_v} \|A_v \tilde{x}_v - \tilde{b}_v\|_2, \quad (8.5.20)$$

where $\tilde{x} = Q^T x$ and $\tilde{b} = Pb$ have been partitioned conformally with PAQ in (8.5.18). The remaining part of the solution $\tilde{x}_k, \dots, \tilde{x}_1$ is then determined by

$$A_{si} \tilde{x}_i = \tilde{b}_i - \sum_{j=i+1}^k U_{ij} \tilde{x}_j, \quad i = k : -1 : 1. \quad (8.5.21)$$

Finally, we have $x = Q\tilde{x}$. We can solve the subproblems in (8.5.20) and (8.5.21) 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.

If the matrix A has structural rank less than n , then we have an underdetermined block A_h . In this case we can still obtain the form (8.5.19) with a square block A_{11} by permuting the extra columns in the first block to the end. The least squares solution is then not unique, but a unique solution of minimum length can be found as outlined in Section 2.7.

8.5.5 Sparse Least Squares Problems

For well-conditioned least squares problems solving the normal equations

$$A^T A x = A^T b$$

(see Section 8.2) can give quite sufficiently accurate results. In this approach the matrix $C = A^T A$ is formed and its Cholesky factorization $C = LL^T$ computed. The least squares solution is then obtained by solving two triangular systems $L(L^T x) = A^T b$. Much of the well developed techniques for solving sparse symmetric positive definite systems can be used; see Section 7.7.5. Let the rows and columns of A be reordered as $B = QAP$, where P and Q are permutation matrices. Since $Q^T Q = I$, it follows that

$$B^T B = P^T A^T Q^T Q A P = (AP)^T AP.$$

This shows that a reordering of the rows will not affect the Cholesky factor L .

The first step in computing the Cholesky factorization of a sparse matrix C is the symbolic analyze phase. In this, using the nonzero structure of C , a fill reducing column permutation P of C is determined. Simultaneously a storage scheme for the Cholesky factor of PCP^T is set up. To compute the structure of the symmetric matrix $A^T A$, we partition A by rows. If we let $a_i^T = e_i^T A$ be the i th row of A , then

$$A^T A = \sum_{i=1}^m a_i a_i^T, \quad (8.5.22)$$

This expresses $A^T A$ as the sum of m rank one matrices. Invoking the no-cancellation assumption this shows that the nonzero structure of $A^T A$ is the direct sum of the structures 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 structure of $A^T A$ is not changed by dropping any row of A whose nonzero structure is a subset of another row. This observation can often speed up the algorithm considerably.

There is an efficient algorithm due to George and Ng [244] to perform the symbolic factorization of $A^T A$ directly using the structure of A . This removes the step of determining the structure of $A^T A$.

For ill-conditioned or stiff problems methods based on the QR factorization should be preferred. As is well known, the factor R in the QR factorization of A is mathematically equivalent to the upper triangular Cholesky factor L^T of $A^T A$. In particular, the nonzero structure must be the same.

However, this approach may be too generous in allocating space for nonzeros in R . To see this, consider a matrix with the structure

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{pmatrix} \quad (8.5.23)$$

For this matrix $R = A$ since A is already upper triangular. But, since $A^T A$ is full the algorithm above will predict R to be full. This overestimate occurs because we start with the structure of $A^T A$, and not with the structure of A . The elements in $A^T A$ are not independent, and cancellation *can* occur in the Cholesky factorization 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 .

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 8.5.2.

*A matrix $A \in \mathbf{R}^{m \times n}$, $m \geq n$, is said to have the **strong Hall property** if for every subset of k columns, $0 < k < m$, the corresponding submatrix has nonzeros in at least $k + 1$ rows. (Thus, when $m > n$, every subset of $k \leq n$ has the required property, and when $m = n$, every subset of $k < n$ columns has the property.)*

For matrices with 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 in (8.5.23) does not have the strong Hall property since the first column has only one nonzero element. However, the matrix with the nonzero structure

$$\begin{pmatrix} \times & \times & \times & \times & \times \\ \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{pmatrix} \quad (8.5.24)$$

obtained by deleting the first column has 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. However, the number of 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; see Section 8.5.1. 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 recommended for banded sparse matrices.

Algorithm 8.8. 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 after 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 after increasing $l_i(A)$.

An alternative row ordering has been found to work well is obtained by ordering the rows after increasing values of $l_i(A)$. With this ordering only the columns $f_i(A)$ to $l_i(A)$ of R_{i-1} when row a_i^T is being processed will be involved, since all the previous rows only have nonzeros 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 reflections. 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. However, as shown in Section 8.5.1, 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 Givens rotations. Initialize R_0 to bhave 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: we uncompress this row into a full vector and scan the nonzeros from left to right. For each $a_{kj} \neq 0$ a Givens rotation involving row j in R_{k-1} is used to annihilate a_{kj} . This may create new nonzeros 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 [242]. Assume that the first k rows of A have been processed to generate $R^{(k)}$. In Figure 8.5.3 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 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 .

$$\begin{bmatrix} R_{k-1} \\ a_k^T \end{bmatrix} = \left[\begin{array}{cccccccccc} \times & 0 & \times & 0 & 0 & \times & 0 & 0 & 0 & 0 \\ \otimes & 0 & \oplus & \otimes & 0 & 0 & 0 & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 & 0 & 0 & \times & 0 & 0 \\ \otimes & \oplus & 0 & \otimes & 0 & 0 & 0 & 0 & 0 & 0 \\ \otimes & \oplus & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \otimes & \oplus & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \times & 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 & 0 \\ \otimes & \otimes & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \otimes & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \otimes & 0 & \otimes & \oplus & 0 & \oplus & \oplus & 0 & 0 \end{array} \right]$$

Figure 8.5.4. Circled elements \otimes in R_{k-1} are involved in the elimination of a_k^T ; fill elements are denoted by \oplus .

Note that unlike the Householder method intermediate fill now only takes place in the row that is 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 Figure 8.5.3, but the Givens rotations should be applied simultaneously to b to form $Q^T b$. In the implementation by George and Heath [236] the Givens 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 orthogonal factor Q is often much less sparse than R . For example, in a grid problem with n unknowns it is known that the number of nonzero elements in R is of the order $O(n \log n)$, whereas the number of nonzeros in Q is $O(n^{3/2})$; see Gilbert, Ng and Peyton [248].

In general, one should if possible avoid computing Q explicitly. If Q is required, then the Givens rotations could be saved separately. This in general requires far less storage and fewer operations than computing and storing Q itself. However, discarding Q creates a problem if we later wish to solve additional problems having the same matrix A but a different right-hand side b , since we cannot form $Q^T b$. One could recompute the factorization, but in most cases a satisfactory method to deal with this problem is to use the **corrected seminormal equations**; see [61, Sec. 6.6.5].

Example 8.5.2 (*T. Elfving and I. Skoglund [191]*).

To illustrate the effect of different column orderings we consider a model used for substance transport in rivers. In this time series data L_{ij} , $i = 1 : n$, $j = 1 : m$, 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 in two sets $x^T = (x_1, x_2)$ and a regularization matrix is added. The following figures

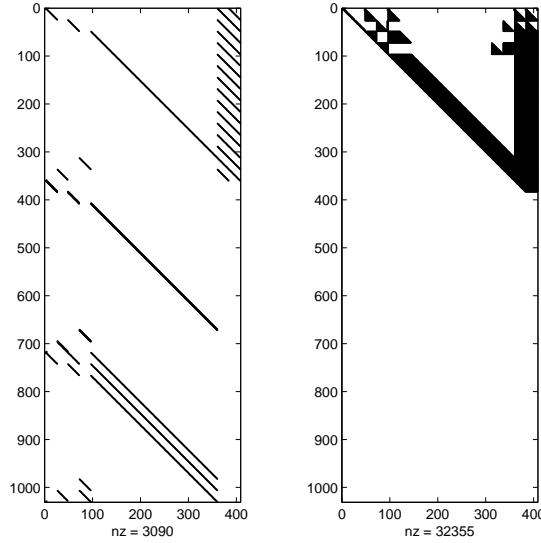


Figure 8.5.5. Nonzero pattern of A and its R factor after reordering using MATLAB's `colperm`.

shows 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. The second (`colamd` in MATLAB) is an approximate minimum degree ordering on A^TA ; see Section 7.7.6.

8.5.6 Multifrontal QR decomposition.

A significant advance in direct methods for sparse matrix factorization is the **multifrontal method** by Duff and Reid [179, 1983]. This method reorganizes the factorization of a sparse matrix into a sequence of partial factorizations of small dense matrices, and is well suited for parallelism. A multifrontal algorithm for the QR factorization was first developed by Liu [412, 1986]. Liu generalized the row-oriented scheme of George and Heath by using submatrix rotations and remarked that this scheme is essentially equivalent to a multifrontal method. He showed that his algorithm can give a significant reduction in QR decomposition time at a modest increase in working storage. George and Liu [240, 1987] presented a modified version of Liu's algorithm which uses Householder transformations instead of Givens rotations.

There are several advantages with the multifrontal approach. The solution of the dense subproblems can more efficiently be handled by vector machines. Also, it leads to independent subproblems which can be solved in parallel. The good data locality of the multifrontal method gives fewer page faults on paging systems,

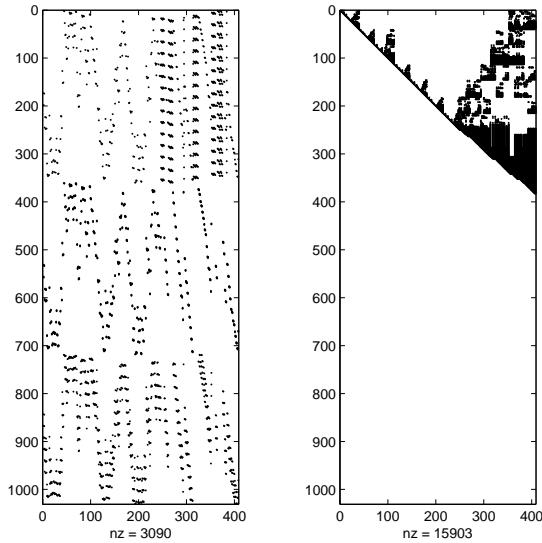


Figure 8.5.6. Nonzero pattern of A and its R factor after reordering using MATLAB's `colamd`.

and out-of-core versions can be developed. Multifrontal methods for sparse QR decompositions have been extensively studied

$$A = \begin{pmatrix} \times & & \times & \times \\ \times & & \times & \times \\ \times & & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & & \times & \times \\ & & & \times & \times \\ & & & \times & \times \end{pmatrix}.$$

Figure 8.5.7. A matrix A corresponding to a 3×3 mesh.

We first describe the multiple front idea on the small 12×9 example in Figure 6.6.3, adopted from Liu [412, 1986]. This matrix arises from a 3×3 mesh problem using a nested dissection ordering, and its graph $G(A^T A)$ is given in Figure 6.6.4.

We first perform a QR decomposition of rows 1–3. Since these rows have nonzeros only in columns $\{1, 5, 7, 8\}$ this operation can be carried out as a QR decomposition 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

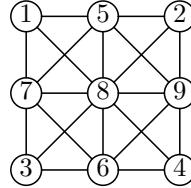


Figure 8.5.8. The graph $G(A^T A)$ and a nested dissection ordering.

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 first stage the matrix has the form shown in Figure 6.6.5. 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 .

$$\begin{pmatrix} \times & & \times & \times & \times \\ & \times & \times & \times & \\ & & \times & \times & \\ & & & \times & \\ \times & & \times & & \times & \times \\ & & & \times & \times & \\ & & & & \times & \times \\ & & & & & \times & \times \\ & & & & & & \times & \times \\ & & & & & & & \times & \times \end{pmatrix}.$$

Figure 8.5.9. The reduced matrix after the first elimination stage.

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 after the index of the first nonzero element, and then perform a QR decomposition:

$$Q^T \begin{pmatrix} \times & \times & \times \\ \times & & \times & \times \\ & \times & \times \\ & & \times & \times \end{pmatrix} = \begin{pmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{pmatrix}.$$

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 we merge the remaining two upper trapezoidal (in this example triangular) matrices and produce the final factor R . This corresponds to eliminating columns 7, 8, and 9.

The scheme described here can also be viewed as a special type of variable row pivoting method. This pivoting scheme has never become very popular because

of the difficulty of generating good orderings for the rotations and because these schemes are complicated to implement. Also, the dynamic storage structure needed tends to reduce the efficiency.

The organization of the multifrontal method is based on the elimination tree, and nodes in the tree are visited in turn given by the ordering. 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 , and is stored in a stack until needed. Hence, a formal description of the method is as follows.

Algorithm 8.9.

MULTIFRONTAL SPARSE QR ALGORITHM.

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 ;

end.

The node ordering of an elimination tree is such that children nodes are numbered before their parent node. Such orderings are called **topological orderings**. All topological orderings of the elimination tree are equivalent in the sense that they give the same triangular factor R . A **postordering** is a topological ordering in which a parent node j always has node $j - 1$ as one of its children. Postorderings are particularly suitable for the multifrontal method, and can be determined by a depth-first search; see Liu [413, 1990]. For example, the ordering of the nodes in the tree in Figure 6.6.6 can be made into a postordering by exchanging labels 3 and 5. The important advantage of using a postordering in the multifrontal method is that data management is simplified since the update matrices can be managed in a stack on a last-in-first-out basis. This also reduces the storage requirement.

The frontal matrices in the multifrontal method are often too small to make it possible to efficiently utilize vector processors and matrix-vector operations in the solution of the subproblems. A useful modification of the multifrontal method, therefore, is to amalgamate several nodes into one supernode. Instead of eliminating one column in each node, the decomposition of the frontal matrices now involves the elimination of several columns, and it may be possible to use Level 2 or even Level 3 BLAS.

In general, nodes can be grouped together to form a supernode if they correspond to a block of contiguous columns in the Cholesky factor, where the diagonal block is full triangular and these rows all have identical off-block diagonal column

structures. Because of the computational advantages of having large supernodes, it is advantageous to relax this condition and also amalgamate nodes which satisfy this condition if some local zeros are treated as nonzeros. A practical restriction is that if too many nodes are amalgamated then the frontal matrices become sparse. (In the extreme case when all nodes are amalgamated into one supernode, the frontal matrix becomes equal to the original sparse matrix!) Note also that non-numerical operations often make up a large part of the total decomposition time, which limits the possible gain.

In many implementations of the multifrontal algorithms the orthogonal transformations are not stored, and the seminormal equations are used for treating additional right-hand sides. If Q is needed then it should not be stored explicitly, but instead be 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 Q has $O(n\sqrt{n})$ nonzeros. Lu and Barlow [418] show that Storing the frontal Householder matrices only requires $O(n \log n)$ storage.

8.5.7 Kronecker Product Problems

Sometimes least squares problems occur which have a highly regular block structure. Here we consider least squares problems of the form

$$\min_x \| (A \otimes B)x - c \|_2, \quad c = \text{vec } C, \quad (8.5.25)$$

where the $A \otimes B$ is the **Kronecker product** of $A \in \mathbf{R}^{m \times n}$ and $B \in \mathbf{R}^{p \times q}$. This product is the $mp \times nq$ block matrix,

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}.$$

Problems of Kronecker structure 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 often the size of the matrices A and B is large, resulting in models involving several hundred thousand equations and unknowns, such savings may be essential.

We recall from Section 7.7.3 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 pseudo-inverses.

Lemma 8.5.3.

Let A^\dagger and B^\dagger be the pseudo-inverses of A and B . Then

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger.$$

Proof. The theorem follows by verifying that $X = A^\dagger \otimes B^\dagger$ satisfies the four Penrose conditions in (8.1.37)–(8.1.38). \square

Using Lemmas 7.7.6 and 8.5.3 the solution to the Kronecker least squares problem (8.5.25) can be written

$$x = (A \otimes B)^\dagger c = (A^\dagger \otimes B^\dagger) \text{vec } C = \text{vec}(B^\dagger C (A^\dagger)^T). \quad (8.5.26)$$

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 $O(m^2n^4)$ to $O(mn^2)$.

In some areas the most common approach to computing the least squares solution to (8.5.25) is to use the normal equations. If we assume that both A and B have full column rank, then we can use the expressions

$$A^\dagger = (A^T A)^{-1} A^T, \quad B^\dagger = (B^T B)^{-1} B^T.$$

However, because of the instability associated with the explicit formation of $A^T A$ and $B^T B$, an approach based on orthogonal factorizations should generally be preferred. If we have computed the complete QR factorizations of A and B ,

$$A\Pi_1 = Q_1 \begin{pmatrix} R_1 & 0 \\ 0 & 0 \end{pmatrix} V_1^T, \quad B\Pi_2 = Q_2 \begin{pmatrix} R_2 & 0 \\ 0 & 0 \end{pmatrix} V_2^T,$$

with R_1, R_2 upper triangular and nonsingular, then from Section 2.7.3 we have

$$A^\dagger = \Pi_1 V_1 \begin{pmatrix} R_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q_1^T, \quad B^\dagger = \Pi_2 V_2 \begin{pmatrix} R_2^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q_2^T.$$

These expressions can be used in (8.5.26) to compute the pseudo-inverse solution of problem (8.5.25) even in the rank deficient case.

The SVD of a Kronecker product $A \otimes B$ can be simply expressed in terms of the SVD of A and B . If A and B have the singular value decompositions

$$A = U_1 \Sigma_1 V_1^T, \quad B = U_2 \Sigma_2 V_2^T.$$

then using Lemma 8.5.3 it follows that

$$A \otimes B = (U_1 \otimes U_2)(\Sigma_1 \otimes \Sigma_2)(V_1 \otimes V_2)^T. \quad (8.5.27)$$

When using this to solve the least squares problem (8.5.25) we can work directly with the two small SVDs of A and B . The solution $x = (A \otimes B)^\dagger c$ can be written as $x = \text{vec}(X)$ where

$$X = B^\dagger C (A^\dagger)^T = V_2 \Sigma_2^\dagger (U_2^T C U_1) (\Sigma_1^T)^\dagger V_1^T. \quad (8.5.28)$$

8.5.8 Tensor Computations

A **tensor** is a multidimensional array. For example, a third-order tensor A has elements with three indices a_{ijk} . When one or more indices are held constant in a tensor, subarrays are formed. In a third-order tensor fixing one index gives a matrix and fixing two indices gives a vector. Thus, a third-order tensor can be thought of built up of slices of matrices in three different ways, depending on what index is fixed.

Tensor decompositions have been used for some time in psychometrics and chemometrics. Lately it has been taken up in other fields such as signal processing and linear algebra. Both with respect to notations and theory this field is still in its infancy.

In matrix computations an important role is played by different matrix decompositions. An important example is the SVD

$$A = U\Sigma V^T = \sum_{i=1}^n \sigma_i u_i v_i^T,$$

of as matrix $A \in \mathbf{R}^{m \times n}$. This expresses a matrix as the weighted sum of rank one matrices $u_i v_i^T$, with $\|u_i\|_2 = \|v_i\|_2 = 1$. Truncating the SVD expansion gives the best approximation to A by a matrix of rank $r < n$; see Eckart–Young Theorem 8.1.18.

A natural question is if the SVD can somehow be generalized to tensors. One approach is to seek an expansion of the form

$$A = \sum_{i=1}^n \sigma_i T_i,$$

where T_i are tensors of rank one. A rank one matrix $X = (x_{ij})$ can always be written as an outer product of two vectors ab^T , and $x_{ij} = \alpha_i \beta_j$. Similarly, a three-dimensional tensor of rank one has the form

$$T = (t_{ijk}), \quad t_{ijk} = \alpha_i \beta_j \gamma_k.$$

The Frobenius norm of a three-dimensional tensor is defined as

$$\|A\|_F^2 = \sum_{i,j,k} a_{ijk}^2.$$

The tensor A is said to have rank r if there is an expansion $A = \sum_{i=1}^r \sigma_i T_i$, where T_i are rank one tensors. In many applications we would like to approximate a tensor A with a tensor of lower rank, i.e., to find rank one tensors T_i so that

$$\|A - \sum_{i=1}^r \sigma_i T_i\|_F$$

is minimized. This is the so called **parafac** (parallel factorization) decomposition of A .

Tensor decompositions: see [381, 380, 396, 397, 189].

Review Questions

- 5.1** What is meant by the standard form of a banded rectangular matrix A ? Why is it important that a banded matrix is permuted into standard form before its orthogonal factorization is computed?
- 5.2** In least squares linear regression the first column of A often equals the vector $a_1 = e = (1, 1, \dots, 1)^T$ (cf. Example 8.2.1). Setting $A = (e \ A_2)$, show that performing one step in MGS is equivalent to “subtracting out the means”.

Problems

- 5.1** Consider the two-block least squares problem (8.5.5). Work out an algorithm to solve the reduced least squares problem $\min_{x_2} \|P_{\mathcal{N}(A_1^T)}(A_2 x_2 - b)\|_2$ using the method of normal equations.

Hint: First show that $P_{\mathcal{N}(A_1^T)}(A) = I - A_1(R_1^T R_1)^{-1} A_1^T$, where R_1 is the Cholesky factor of $A_1^T A_1$.

- 5.2** (a) Write a MATLAB program for fitting a straight line $c_1 x + c_2 y = d$ to given points $(x_i, y_i) \in \mathbf{R}^2$, $i = 1 : m$. The program should handle all exceptional cases, e.g., $c_1 = 0$ and/or $c_2 = 0$.
(b) Suppose we want to fit two set of points $(x_i, y_i) \in \mathbf{R}^2$, $i = 1, \dots, p$, and $i = p + 1, \dots, m$, 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 are 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.

- 5.3** Use the Penrose conditions to prove the formula

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger,$$

where \otimes denotes the Kronecker product

- 5.4** Test the recursive QR algorithm `recqr(A)` given in Section sec8.2.rec on some matrices. Check that you obtain the same result as from the built-in function `qr(A)`.

8.6 Some Generalized Least Squares Problems

8.6.1 Least Squares for General Linear Models

We consider a general univariate linear model where the error covariance matrix equals $\sigma^2 V$, where V is a positive definite symmetric matrix. Then the best unbiased linear estimate equals the minimizer of

$$(Ax - b)^T V^{-1} (Ax - b) \tag{8.6.1}$$

For the case of a general positive definite covariance matrix we assume that the Cholesky factorization $V = R^T R$ of the covariance matrix can be computed. Then the least squares problem (8.6.1) becomes

$$(Ax - b)^T V^{-1} (Ax - b) = \|R^{-T}(Ax - b)\|_2^2. \quad (8.6.2)$$

This could be solved in a straightforward manner by forming $\tilde{A} = R^{-T} A$, $\tilde{b} = R^{-T} b$ and solving the standard least squares problem

$$\min_x \|\tilde{A}x - \tilde{b}\|_2.$$

A simple special case of (8.6.1) is when the covariance matrix V is a positive diagonal matrix,

$$V = \sigma^2 \text{diag}(v_1, v_2, \dots, v_m) > 0.$$

This leads to the **weighted** linear least squares problem (8.1.17)

$$\min_x \|D(Ax - b)\|_2 = \min_x \|(DA)x - Db\|_2 \quad (8.6.3)$$

where $D = \text{diag}(v_{ii}^{-1/2})$. Note that the weights will be large when the corresponding error component in the linear model has a small variance. We assume in the following that the matrix A is row equilibrated, that is,

$$\max_{1 \leq j \leq n} |a_{ij}| = 1, \quad i = 1 : m.$$

and that the weights $D = \text{diag}(d_1, d_2, \dots, d_n)$ have been ordered so that

$$\infty > d_1 \geq d_2 \geq \dots \geq d_m > 0. \quad (8.6.4)$$

Note that only the *relative size* of the weights influences the solution.

If the ratio $\gamma = d_1/d \gg 1$ we call the weighted problems **stiff**. Stiff problems arise, e.g., in electrical networks, certain classes of finite element problems, and in interior point methods for constrained optimization. Special care is needed in solving stiff weighted linear least squares problems. The method of normal equations is not well suited for solving such problems. To illustrate this, we consider the special case where only the first $p < n$ equations are weighted,

$$\min_x \left\| \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} x - \begin{pmatrix} \gamma b_1 \\ b_2 \end{pmatrix} \right\|_2^2, \quad (8.6.5)$$

$A_1 \in \mathbf{R}^{p \times n}$ and $A_2 \in \mathbf{R}^{(m-p) \times n}$. Such problems occur, for example, when the method of weighting is used to solve a least squares problem with the linear equality constraints $A_1 x = b_1$; see Section 5.1.4. For this problem the matrix of normal equations becomes

$$B = (\gamma A_1^T \quad A_2^T) \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} = \gamma^2 A_1^T A_1 + A_2^T A_2.$$

If $\gamma > u^{-1/2}$ (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 , then this is a disaster, since the solution depends critically on the less precise data in A_2 . (The matrix in Example 2.2.1 is of this type.)

Clearly, if the problem is stiff the condition number $\kappa(DA)$ will be large. An upper bound of the condition number 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 $\{D, A, b\}$ is ill-conditioned when $\gamma \gg 1$.

We now examine the use of methods based on the QR factorization of A for solving weighted problems. We first note that it is essential that *column pivoting* is performed when QR factorization is used for weighted problems. To illustrate the need for column pivoting, consider an example of the form (8.6.5), where

$$A_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \end{pmatrix},$$

Then stability is lost without column pivoting because the first two columns of the matrix A_1 are linearly dependent. When column pivoting is introduced this difficulty disappears.

The following example shows that the Householder QR method can give poor accuracy for weighted problems even if column pivoting is used.

Example 8.6.1 (Powell and Reid [492]).

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 equal to $x = (1, 1, 1)$. Using exact arithmetic we obtain after the first step of QR factorization of A by Householder transformations 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. However, 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 . (However, compared to the method of normal equations, which fails already when $\gamma > u^{-1/2}$, this is an improvement!)

As shown by Cox and Higham [120], provided that an initial **row sorting** is performed the Householder QR method with column pivoting has very good stability properties for weighted problems. The rows of $\tilde{A} = DA$ and $\tilde{b} = Db$ should be sorted after decreasing ∞ -norm,

$$\max_j |\tilde{a}_{1j}| \geq \max_j |\tilde{a}_{2j}| \geq \cdots \geq \max_j |\tilde{a}_{mj}|. \quad (8.6.6)$$

(In Example 8.6.1 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 QR with column pivoting can be used.

If the Cholesky factor R is ill-conditioned there could be a loss of accuracy in forming \tilde{A} and \tilde{b} . But assume that column pivoting has been used in the Cholesky factorization and set $R = D\hat{R}$, where \hat{R} is *unit upper triangular* and D diagonal. Then any ill-conditioning is usually reflected in D . If necessary a presorting of the rows of $\tilde{A} = D^{-1}\hat{R}^{-T}A$ can be performed before applying the Householder QR method to the transformed problem.

In **Paige's method** [463] the general linear model with covariance matrix $V = BB^T$ is reformulated as

$$\min_{v,x} \|v\|_2 \quad \text{subject to} \quad Ax + Bv = b. \quad (8.6.7)$$

This formulation has the advantage that it is less sensitive to an ill-conditioned V and allows for rank deficiency in both A and V . For simplicity we consider here only the case when V is positive definite and $A \in \mathbf{R}^{m \times n}$ has full column rank n .

The first step is to compute the QR factorization

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix}, \quad Q^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad (8.6.8)$$

where R is nonsingular. The same orthogonal transformation is applied also to B , giving

$$Q^T B = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \begin{matrix} n \\ m-n \end{matrix},$$

where by the nonsingularity of B it follows that $\text{rank}(C_2) = m - n$. The constraints in (8.6.7) can now be written in the partitioned form

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} v + \begin{pmatrix} R \\ 0 \end{pmatrix} x = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix},$$

or

$$Rx = c_1 - C_1 v, \quad C_2 v = c_2. \quad (8.6.9)$$

where $C_2 = \mathbf{R}^{(m-n) \times m}$. For any vector v we can determine x so that the first block of these equations is satisfied.

An orthogonal matrix $P \in \mathbf{R}^{m \times m}$ can then be determined such that

$$C_2 P = (0 \quad S), \quad (8.6.10)$$

where S is upper triangular and nonsingular. Setting $u = P^T v$ the second block of the constraints in (8.6.9) becomes

$$C_2 P(P^T v) = (0 \quad S) u = c_2.$$

Since P is orthogonal $\|v\|_2 = \|u\|_2$ and so the minimum in (8.6.7) is found by taking

$$v = P \begin{pmatrix} 0 \\ u_2 \end{pmatrix}, \quad u_2 = S^{-1} c_2. \quad (8.6.11)$$

Then x is obtained from 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 (8.6.7) with covariance matrix $\sigma^2 C$, where

$$C = R^{-1} L^T L R^{-T}, \quad L^T = C_1^T P_1. \quad (8.6.12)$$

Paige's algorithm (8.6.8)–(8.6.11) as described above does not take advantage of any special structure the matrix B may have. It requires a total of about $4m^3/3 + 2m^2n$ flops. If $m \gg n$ the work in the QR factorization of C_2 dominates. It is not suitable for problems where B is sparse, e.g., diagonal as in the case of a weighted least squares problem.

Several generalized least squares problems can be solved by using a generalized SVD (GSVD) or QR (GQR) factorization involving a pair of matrices A, B . One motivation for using this approach is to avoid the explicit computation of products and quotients of matrices. For example, let A and B be square and nonsingular matrices and assume we need the SVD of AB^{-1} (or AB). Then the explicit calculation of AB^{-1} (or AB) may result in a loss of precision and should be avoided.

The factorization used in Paige's method is a special case of the generalized QR (GQR) factorization. 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. The GQR factorization of A and B is a factorization of the form

$$A = QR, \quad B = QTZ, \quad (8.6.13)$$

where $Q \in \mathbb{R}^{m \times m}$ and $Z \in \mathbb{R}^{p \times p}$ are orthogonal matrices and R and T have one of the forms

$$R = \begin{pmatrix} R_{11} \\ 0 \end{pmatrix} \quad (m \geq n), \quad R = (R_{11} \quad R_{12}) \quad (m < n), \quad (8.6.14)$$

and

$$T = (0 \quad T_{12}) \quad (m \leq p), \quad T = \begin{pmatrix} T_{11} \\ T_{21} \end{pmatrix} \quad (m > p). \quad (8.6.15)$$

If B is square and nonsingular GQR implicitly gives the QR factorization of $B^{-1}A$. There is also a similar generalized RQ factorization related to the QR factorization of AB^{-1} . Routines for computing a GQR factorization are included in LAPACK. These factorizations allow the solution of very general formulations of several least squares problems.

8.6.2 Indefinite Least Squares

A matrix $Q \in \mathbf{R}^{n \times n}$ is said to be J -orthogonal if

$$Q^T J Q = J, \quad (8.6.16)$$

where J is a **signature matrix**, i.e. a diagonal matrix with elements equal to ± 1 . This implies that Q is nonsingular and $Q J Q^T = J$. Such matrices are useful in the treatment of problems where there is an underlying indefinite inner product.

In order to construct J -orthogonal matrices we introduce the **exchange operator**. Consider the block 2×2 system

$$Qx = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}. \quad (8.6.17)$$

Solving the first equation for x_1 and substituting in the second equation gives

$$\text{exc}(Q) \begin{pmatrix} y_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_2 \end{pmatrix},$$

where x_1 and y_1 have been exchanged and

$$\text{exc}(Q) = \begin{pmatrix} Q_{11}^{-1} & -Q_{11}^{-1}Q_{12} \\ Q_{21}Q_{11}^{-1} & Q_{22} - Q_{21}Q_{11}^{-1}Q_{12} \end{pmatrix} \quad (8.6.18)$$

Here the $(2, 2)$ block is the Schur complement of Q_{11} in Q . From the definition it follows that the exchange operator satisfies

$$\text{exc}(\text{exc}(Q)) = Q,$$

that is it is involuntary.

Theorem 8.6.1.

Let $Q \in \mathbf{R}^{n \times n}$ be partitioned as in (8.6.17). 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. A proof due to Chris Paige is given in Higham [330]. \square

The indefinite least squares problem (ILS) has the form

$$\min_x (b - Ax)^T J (b - Ax), \quad (8.6.19)$$

where $A \in \mathbf{R}^{m \times n}$, $m \geq n$, and $b \in \mathbf{R}^m$ are given. In the following we assume for simplicity that

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad J = \begin{pmatrix} I_{m_1} & 0 \\ 0 & -I_{m_2} \end{pmatrix}, \quad (8.6.20)$$

where $m_1 + m_2 = m$, $m_1 m_2 \neq 0$. If the Hessian matrix $A^T J A$ is positive definite this problem has a unique solution which satisfies the normal equations $A^T J Ax = A^T J b$, or

$$(A_1^T A_1 - A_2^T A_2)x = A_1^T b_1 - A_2^T b_2. \quad (8.6.21)$$

Hence, if we form the normal equations and compute the Cholesky factorization $A^T J A = R^T R$ the solution is given by $x = R^{-1} R^{-T} c$, where $c = (A^T J b)$. However, by numerical stability considerations we should avoid explicit formation of $A_1^T A_1$ and $A_2^T A_2$. We now show how J -orthogonal transformations can be used to solve this problem more directly.

Consider the orthogonal 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 8.6.1 it follows that

$$\check{G} = \text{exc}(G) = \frac{1}{c} \begin{pmatrix} 1 & -s \\ -s & 1 \end{pmatrix}, \quad (8.6.22)$$

is J -orthogonal

$$\check{G}^T J \check{G} = I, \quad J = \text{diag}(1, -1).$$

The matrix \check{G} is called a hyperbolic plane rotation since it can be written as

$$\check{G} = \begin{pmatrix} \cosh \theta & -\sinh \theta \\ -\sinh \theta & \cosh \theta \end{pmatrix}, \quad \check{c}^2 - \check{s}^2 = 1.$$

A hyperbolic rotation \check{G} can be used to zero a selected component in a vector. Provided that $|\alpha| > |\beta|$ we have

$$\check{G} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{c} \begin{pmatrix} 1 & -s \\ -s & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix},$$

provided that

$$\sigma = \sqrt{\alpha^2 - \beta^2} = \sqrt{(\alpha + \beta)(\alpha - \beta)}, \quad c = \sigma/\alpha, \quad s = \beta/\alpha. \quad (8.6.23)$$

Note that the elements of a hyperbolic rotation \check{G} are unbounded. Therefore, such transformations must be used with care. The direct application of \check{G} to a vector does not lead to a stable algorithm. Instead, as first suggested by Chambers [95], we note the equivalence of

$$\check{G} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \Leftrightarrow G \begin{pmatrix} y_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_2 \end{pmatrix},$$

where $\check{G} = \text{exc}(G)$. We first determine y_1 from the hyperbolic rotation and then y_2 from the Given's rotation giving

$$y_1 = (x_1 - sx_2)/c, \quad y_2 = cx_2 - sy_1. \quad (8.6.24)$$

We now describe an alternative algorithm for solving the indefinite least squares problem (8.6.19), which combines Householder transformations and hyperbolic rotations. In the first step we use two Householder transformations. The first transforms rows $1 : m_1$ in A_1 and zeros the elements $2 : m_1$. The second transforms rows $1 : m_2$ in A_2 and zeros elements $2 : m_2$. We now zero out the element left in the first column of A_2 using a hyperbolic rotation in the plane $(1, m_1 + 1)$. In the next step we proceed similarly. We first zero elements $3 : m_1$ in the second column of A_1 and elements $1 : m_2$ in the second column of A_2 . A hyperbolic rotation in the plane $(2 : m_1 + 1)$ is then used to zero the remaining element in the second column of A_2 . After the first two steps we have reduced the matrix A to the form

This method uses n hyperbolic rotations and n Householder transformations for the reduction. Since the cost for the hyperbolic rotations is $= (n^2)$ flops the total cost is about the same as for the usual Householder QR factorization.

Note that this can be combined with column pivoting so that at each step we maximize the diagonal element in R . It suffices to consider the first step; all remaining steps are similar. Changing notations to $A = (a_1, \dots, a_n) \equiv A_1$, $B = (b_1, \dots, b_n) \equiv A_2$, we do a modified Golub pivoting. Let p be the smallest index for which

$$s_p \geq s_j, \quad s_j = \|a_j\|_2^2 - \|b_j\|_2^2, \quad \forall j = 1 : n,$$

and interchange columns 1 and p in A and B .

A special case of particular interest is when A_2 consists of a single row. In this case only hyperbolic transformations on A_2 occur.

8.6.3 Linear Equality Constraints

In some least squares problems in which 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. Such problems can be considered as the limit of a sequence of weighted problems when some weights tend to infinity.

Given matrices $A \in \mathbf{R}^{m \times n}$ and $B \in \mathbf{R}^{p \times n}$ we consider the problem **LSE** to find a vector $x \in \mathbf{R}^n$ which solves

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad Bx = d. \quad (8.6.25)$$

A solution to problem (8.6.25) exists if and only if the linear system $Bx = d$ is consistent. If $\text{rank}(B) = p$ then B has linearly independent rows, and $Bx = d$ is consistent for any right hand side d . A solution to problem (8.6.25) is unique if and only if the null spaces of A and B intersect only trivially, i.e., if $\mathcal{N}(A) \cap \mathcal{N}(B) = \{0\}$, or equivalently

$$\text{rank} \begin{pmatrix} A \\ B \end{pmatrix} = n. \quad (8.6.26)$$

If (8.6.26) is not satisfied then there is a vector $z \neq 0$ such that $Az = Bz = 0$. Hence, if x solves (8.6.25) then $x + z$ is a different solution. In the following we therefore assume that $\text{rank}(B) = p$ and that (8.6.26) is satisfied.

A robust algorithm for problem LSE should check for possible inconsistency of the constraints $Bx = d$. If it is not known a priori that the constraints are consistent, then problem LSE may be reformulated as a **sequential least squares problem**

$$\min_{x \in S} \|Ax - b\|_2, \quad S = \{x \mid \|Bx - d\|_2 = \min\}. \quad (8.6.27)$$

The most efficient way to solve problem LSE is to derive an equivalent unconstrained least squares problem of lower dimension. There are basically 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 B to upper trapezoidal form. It is essential that column pivoting is used in this step. In order to be able to solve also the more general problem (8.6.27) we compute a QR factorization of B such that

$$Q_B^T B \Pi_B = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}, \quad R_{11} \in \mathbf{R}^{r \times r}, \quad (8.6.28)$$

where $r = \text{rank}(B) \leq p$ and R_{11} is upper triangular and nonsingular. Using this factorization, and setting $\bar{x} = \Pi_B^T x$, the constraints become

$$(R_{11}, R_{12})\bar{x} = R_{11}\bar{x}_1 + R_{12}\bar{x}_2 = \bar{d}_1, \quad \bar{d} = Q_B^T d = \begin{pmatrix} \bar{d}_1 \\ \bar{d}_2 \end{pmatrix}, \quad (8.6.29)$$

where $\bar{d}_2 = 0$ if and only if the constraints are consistent. If we apply the permutation Π_B also to the columns of A and partition the resulting matrix conformally with (8.6.28), $\bar{A}\Pi_B = (A_1, A_2)$, then $Ax - b = A_1\bar{x}_1 + A_2\bar{x}_2 - b$. Solving (8.6.29) for $\bar{x}_1 = R_{11}^{-1}(\bar{d}_1 - R_{12}\bar{x}_2)$, and substituting, we find that the unconstrained least squares problem

$$\begin{aligned} \min_{\bar{x}_2} \|\hat{A}_2\bar{x}_2 - \hat{b}\|_2, \quad \hat{A}_2 &\in \mathbf{R}^{m \times (n-r)} \\ \hat{A}_2 &= \bar{A}_2 - \bar{A}_1 R_{11}^{-1} R_{12}, \quad \hat{b} = b - \bar{A}_1 R_{11}^{-1} \bar{d}_1. \end{aligned} \quad (8.6.30)$$

is equivalent to the original problem LSE. Here \hat{A}_2 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 the condition in (8.6.26) 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 coding of this algorithm can be kept remarkably compact as exemplified by the Algol program of Björck and Golub [63, 1967].

In the null space method we postmultiply B with an orthogonal matrix Q to transform B to lower triangular form. We also apply Q to the matrix A , which gives

$$\begin{pmatrix} B \\ A \end{pmatrix} Q = \begin{pmatrix} B \\ A \end{pmatrix} (Q_1 \quad Q_2) = \begin{pmatrix} L & 0 \\ AQ_1 & AQ_2 \end{pmatrix}, \quad L \in \mathbf{R}^{p \times p}. \quad (8.6.31)$$

where Q_2 is an orthogonal basis for the null space of B . Note that this is equivalent to computing the QR factorization of B^T . The matrix Q can be constructed as a product of Householder transformations. The solution is now split into the sum of two orthogonal components by setting

$$x = Qy = x_1 + x_2 = Q_1y_1 + Q_2y_2, \quad y_1 \in \mathbf{R}^p, \quad y_2 \in \mathbf{R}^{(n-p)}, \quad (8.6.32)$$

where $Bx_2 = BQ_2y_2 = 0$. From the assumption that $\text{rank}(B) = p$ it follows that L is nonsingular and the constraints equivalent to $y_1 = L^{-1}d$ and

$$b - Ax = b - AQy = c - AQ_2y_2, \quad c = b - (AQ_1)y_1.$$

Hence, y_2 is the solution to the unconstrained least squares problem

$$\min_{y_2} \|(AQ_2)y_2 - c\|_2. \quad (8.6.33)$$

This can be solved, for example, by computing the QR factorization of AQ_2 . If (8.6.26) is satisfied then $\text{rank}(AQ_2) = n - p$, then the solution to (8.6.33) is unique. If $y_2 = (AQ_2)^\dagger(b - AQ_1y_1)$ is the minimum length solution to (8.6.33), then since

$$\|x\|_2^2 = \|x_1\|_2^2 + \|Q_2y_2\|_2^2 = \|x_1\|_2^2 + \|y_2\|_2^2$$

$x = Qy$ is the minimum norm solution to problem LSE.

The representation in (8.6.32) of the solution x can be used as a basis for a perturbation theory for problem LSE. A strict analysis is given by Eldén [187, 1982], but the result is too complicated to be given here. If the matrix B is well conditioned, then the sensitivity is governed by $\kappa(AQ_2)$, for which $\kappa(A)$ is an upper bound.

The method of direct elimination and the null space method both 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.

8.6.4 Quadratic Constraints and Tikhonov Regularization

Least squares problems with quadratic constraints arise, e.g., when one wants to balance a good fit to the data points and a smooth solution. Such problems arise naturally from inverse problems where one tries to determine the structure of a physical system from its behavior. An example in the form of an integral equation of the first kind was given in Example 7.1.5. Typically, the singular values of the discretized problem will decay exponentially to zero. As shown in Section 8.4.6, any attempt to solve such a problem without restricting x will lead to a meaningless solution of very large norm, or even to failure of the algorithm.

One of the most successful methods for solving ill-conditioned problems is **Tikhonov regularization**³⁵ (see [564, 1963]). In this method the solution space is

³⁵ Andrei Nicholaevich Tikhonov (1906–1993), Russian mathematician. He made deep contributions in topology and function analysis, but was also interested in applications to mathematical physics. In the 1960's he introduced the concept of “regularizing operator” for ill-posed problems, for which he was awarded the Lenin medal.

restricted by imposing an a priori bound on $\|Lx\|_2$ for a suitably chosen matrix $L \in \mathbf{R}^{p \times n}$. Typically L is taken to be the identity matrix I or a discrete approximation to some derivative operator, e.g.,

$$L = \begin{pmatrix} 1 & -1 & & \\ & 1 & -1 & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{pmatrix} \in \mathbf{R}^{(n-1) \times n}, \quad (8.6.34)$$

which, except for a scaling factor, approximates the first derivative operator.

The above approach leads us to take x as the solution to the problem

$$\min_f \|Ax - b\|_2 \quad \text{subject to} \quad \|Lx\|_2 \leq \gamma. \quad (8.6.35)$$

Here the parameter γ governs the balance between a small residual and a smooth solution. The determination of a suitable γ is often a major difficulty in the solution process. Alternatively, we could consider the related problem

$$\min_f \|Lx\|_2 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \rho. \quad (8.6.36)$$

In the statistical literature the solution of problem (8.6.35) is called a **ridge estimate**. Problems (8.6.35) and (8.6.36) are special cases of the general problem LSQI.

Problem LSQI:

Least Squares with Quadratic Inequality Constraint.

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad \|Lx - d\|_2 \leq \gamma, \quad (8.6.37)$$

where $A \in \mathbf{R}^{m \times n}$, $L \in \mathbf{R}^{p \times n}$, $\gamma > 0$.

Conditions for existence and uniqueness and properties of solutions to problem LSQI have been given by Gander [222, 1981]. Clearly, problem LSQI has a solution if and only if

$$\min_x \|Lx - d\|_2 \leq \gamma, \quad (8.6.38)$$

and in the following we assume that this condition is satisfied. We define a L -generalized solution $x_{A,L}$ to the problem $\min_x \|Ax - b\|_2$ to be a solution to the problem (cf. Section 2.7.4)

$$\min_{x \in S} \|Lx - d\|_2, \quad S = \{x \in \mathbf{R}^n \mid \|Ax - b\|_2 = \min\}. \quad (8.6.39)$$

These observation gives rise to the following theorem.

Theorem 8.6.2.

Assume that problem LSQI has a solution. Then either $x_{A,L}$ is a solution or (8.6.46) holds and the solution occurs on the boundary of the constraint region. In the latter case the solution $x = x(\lambda)$ satisfies the generalized normal equations

$$(A^T A + \lambda L^T L)x(\lambda) = A^T b + \lambda L^T d, \quad (8.6.40)$$

where $\lambda \geq 0$ satisfies the **secular equation**

$$\|Lx(\lambda) - d\|_2 = \gamma. \quad (8.6.41)$$

Proof. Since the solution occurs on the boundary we can use the method of Lagrange multipliers and minimize $\psi(x, \lambda)$, where

$$\psi(x, \lambda) = \frac{1}{2}\|Ax - b\|_2^2 + \frac{1}{2}\lambda(\|Lx - d\|_2^2 - \gamma^2). \quad (8.6.42)$$

A necessary condition for a minimum is that the gradient of $\psi(x, \lambda)$ with respect to x equals zero, which gives (8.6.40). \square

As we shall see, only positive values of λ are of interest. Note that (8.6.40) are the normal equations for the least squares problem

$$\min_x \left\| \begin{pmatrix} A \\ \mu L \end{pmatrix} x - \begin{pmatrix} b \\ \mu d \end{pmatrix} \right\|_2, \quad \mu = \sqrt{\lambda}. \quad (8.6.43)$$

Hence, to solve (8.6.40) for a given value of λ , it is not necessary to form the cross-product matrices $A^T A$ and $L^T L$. Instead we can solve (8.6.43) by QR factorization.

In the following we assume that the constraint $\|Lx(\lambda) - d\|_2 \leq \gamma$ is binding, which is case if (8.6.46) holds. Then there is a unique solution to problem LSQI if and only if the null spaces of A and L intersect only trivially, i.e., $\mathcal{N}(A) \cap \mathcal{N}(L) = \{0\}$, or equivalently

$$\text{rank} \begin{pmatrix} A \\ L \end{pmatrix} = n. \quad (8.6.44)$$

A particularly simple but important case is when $L = I_n$ and $d = 0$, i.e.,

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad \|x\|_2 \leq \gamma, \quad (8.6.45)$$

We call this the **standard form** of LSQI. Notice that for this case we have $x_{A,I} = A^T b$ and the constraint is binding only if

$$\|Lx_{A,L} - d\|_2 > \gamma. \quad (8.6.46)$$

The special structure can be taken advantage of when computing the Householder QR factorization of the matrix in (8.6.43). Below the shape of the transformed matrix after $k = 2$ steps is shown ($m = n = 5$):

$$\begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & + & + & + \\ 0 & + & + & + & + \\ \times & & & & \\ & & \times & & \times \end{pmatrix}$$

Notice that the first two rows of D have filled in, but the remaining rows of D are still not touched. For each step $k = 1 : n$ there are m elements in the current column to be annihilated. Therefore, the operation count for the Householder QR factorization will increase with $2n^3/3$ to $2mn^2$ flops. If $A = R$ already is in upper triangular form then the flop count for the reduction is reduced to approximately $2n^3/3$ (cf. Problem 8.1b).

If $L = I$ the singular values of the modified matrix in (8.6.43) are equal to

$$\tilde{\sigma}_i = (\sigma_i^2 + \lambda)^{1/2}, \quad i = 1 : n.$$

In this case the solution can be expressed in terms of the SVD as

$$x(\lambda) = \sum_{i=1}^n f_i \frac{c_i}{\sigma_i} v_i, \quad f_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda}. \quad (8.6.47)$$

The quantities f_i are often called **filter factors**. Notice that as long as $\sqrt{\lambda} \ll \sigma_i$ we have $f_i \approx 1$, and if $\sqrt{\lambda} \gg \sigma_i$ then $f_i \ll 1$. This establishes a relation to the truncated SVD solution (8.4.39) which corresponds to a filter factor which is a step function $f_i = 1$ if $\sigma_i > \delta$ and $f_i = 0$ otherwise. Rules based on the **discrepancy principle** are perhaps the most reliable for choosing the regularization parameter. However, these requires knowledge of the noise level in the data.

Transformation to Standard Form

A Problem LSQI with $L \neq I$ can be transformed to standard form as we now describe. If L is nonsingular we can achieve this by the change of variables $x = L^{-1}y$. This gives the problem

$$\min_y \left\| \begin{pmatrix} AL^{-1} \\ \mu I \end{pmatrix} y - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2. \quad (8.6.48)$$

The matrix $\tilde{A} = AL^{-1}$ can be formed by solving the matrix equation $\tilde{A}L = A$.

The general case with no restrictions on L has been treated by Eldén [187]. For simplicity, we assume here that $L \in \mathbf{R}^{(n-t) \times n}$ has full row rank, which is often the case in practice. The transformation to standard form can then be achieved using the pseudo-inverse of L . For example, with L as in (8.6.34) the rank deficiency is $t = 1$. Let the QR factorization of L^T be

$$L^T = (V_1 \quad V_2) \begin{pmatrix} R_L \\ 0 \end{pmatrix} \in \mathbf{R}^{n \times (n-t)},$$

where R_2 nonsingular and the orthogonal columns of V_2 span the null space of L . We split x into two orthogonal components

$$x = L^\dagger y + V_2 w = V_1 R_L^{-T} y + V_2 w, \quad (8.6.49)$$

where $L^\dagger = V_1 R_L^{-T}$ is the pseudo-inverse of L . Form $AV = (AV_1 \quad AV_2)$ and compute

$$AL^\dagger = AV_1 R_L^{-T}$$

by solving the upper triangular system $R_L \tilde{A}^T = (AV_1)^T$. We further need the Householder QR factorization

$$AV_2 = Q \begin{pmatrix} U \\ 0 \end{pmatrix} \in \mathbf{R}^{m \times t}, \quad Q = (Q_1 \quad Q_2).$$

If A and L have no null space in common, then AV_2 has rank t and U is nonsingular. Combining these results gives

$$\begin{aligned} Q^T(Ax - b) &= Q^T(AV_1 R_L^{-T} y + AV_2 w - b) \\ &= \begin{pmatrix} Q_1^T(AL^\dagger y - b) + Uw \\ Q_2^T(AL^\dagger y - b) \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}. \end{aligned}$$

Thus, we can always determine w so that $r_1 = 0$. Hence, the problem decouples and y is the solution to

$$\min_y \left\| \begin{pmatrix} \tilde{A} \\ \mu I \end{pmatrix} y - \begin{pmatrix} \tilde{b} \\ 0 \end{pmatrix} \right\|_2, \quad \tilde{A} = Q_2^T A V_1 R_L^{-T}, \quad \tilde{b} = Q_2^T b, \quad (8.6.50)$$

which is of standard form. We then solve $Uw = Q_1^T(AL^\dagger y - b)$ and retrieve x from (8.6.49).

An important special case is when in LSQI we have $A = K$, $L = L$, and both K and L are upper triangular Toeplitz matrices, i.e.,

$$K = \begin{pmatrix} k_1 & k_2 & \dots & k_{n-1} & k_n \\ & k_1 & k_2 & & k_{n-1} \\ & & \ddots & \ddots & \vdots \\ & & & k_1 & k_2 \\ & & & & k_1 \end{pmatrix}$$

and L is as in (8.6.34). Such systems arise when convolution-type Volterra integral equations of the first kind,

$$\int_0^t K(t-s)f(s)dt = g(s), \quad 0 \leq t \leq T,$$

are discretized. Eldén [188] has developed a method for solving problems of this kind which only uses $\frac{9}{2}n^2$ flops for each value of μ . It can be modified to handle the case when K and L also have a few nonzero diagonals below the main diagonal. Although K can be represented by n numbers this method uses $n^2/2$ storage locations. A modification of this algorithm which uses only $O(n)$ storage locations is given in Bojanczyk and Brent [69, 1986].

Solving the Secular Equation.

Methods for solving problem LSQI are usually based on solving the secular equation (8.6.41) using Newton's method. The secular equation can be written in the form

$$f_p(\lambda) = \|x(\lambda)\|^p - \gamma^p = 0. \quad (8.6.51)$$

where $p = \pm 1$ and $x(\lambda)$ be the solution to the least squares problem (8.6.43). From $\|x(\lambda)\|_2^p = (x(\lambda)^T x(\lambda))^{p/2}$, taking derivatives with respect to λ , we find

$$f'_p(\lambda) = p \frac{x^T(\lambda)x'(\lambda)}{\|x(\lambda)\|_2^{2-p}}, \quad x'(\lambda) = -(A^T A + \lambda I)^{-1} x(\lambda). \quad (8.6.52)$$

Since $x(\lambda) = (A^T A + \lambda I)^{-1} A^T b$, we obtain

$$x(\lambda)^T x(\lambda)' = -x(\lambda)^T (A^T A + \lambda I)^{-1} x(\lambda) = -\|z(\lambda)\|_2^2.$$

The choice $p = +1$ gives rise to the iteration

$$\lambda_{k+1} = \lambda_k + \left(1 - \frac{\gamma}{\|x(\lambda_k)\|_2}\right) \frac{\|x(\lambda_k)\|_2^2}{\|z(\lambda_k)\|_2^2}, \quad (8.6.53)$$

whereas $p = -1$ gives

$$\lambda_{k+1} = \lambda_k - \left(1 - \frac{\|x(\lambda_k)\|_2}{\gamma}\right) \frac{\|x(\lambda_k)\|_2^2}{\|z(\lambda_k)\|_2^2}. \quad (8.6.54)$$

due to to Hebden [314] and Reinsch [501]). The asymptotic rate of convergence for Newton's method is quadratic. For $p = \pm 1$ converge is monotonic, provided that the initial approximation satisfies $0 \leq \lambda^{(0)} < \lambda$. Therefore, $\lambda^{(0)} = 0$ is often used as a starting approximation. Close to the solution $\|x(\lambda_k)\| \approx \gamma$, and then the Newton correction is almost the same independent of p . However, for small λ , we can have $\|x(\lambda_k)\| \gg \gamma$ and then $p = -1$ gives much more rapid convergence than $p = 1$.

It has been shown (Reinsch [501]) that $h(\lambda)$ is convex, and hence that the iteration (8.6.54) is monotonically convergent to the solution λ^* if started within $[0, \lambda^*]$. Note that the correction to λ_k in (8.6.54) equals the Newton correction in (8.6.53) multiplied by the factor $\|x(\lambda)\|/\gamma$. Using the QR factorization

$$Q(\lambda)^T \begin{pmatrix} A \\ \sqrt{\lambda} I \end{pmatrix} = \begin{pmatrix} R(\lambda) \\ 0 \end{pmatrix}, \quad c_1(\lambda) = (I_n \quad 0) Q(\lambda)^T \begin{pmatrix} b \\ 0 \end{pmatrix}, \quad (8.6.55)$$

we have

$$x(\lambda) = R(\lambda)^{-1} c_1(\lambda), \quad z(\lambda) = R(\lambda)^{-T} x(\lambda). \quad (8.6.56)$$

The main cost in this method is for computing the QR factorization (8.6.55) in each iteration step. On the other hand computing the derivative costs only one triangular solve. Assume that the function qr computes the “thin” QR factorization, with $Q \in \mathbf{R}^{m \times n}$. Then Hebden's algorithm is:

Algorithm 8.10. Hebden's method.

The algorithm performs p steps of Hebden's iteration to the constrained least squares problem $\min \|Ax - b\|_2$ subject to $\|x\|_2 = gamma > 0$, using QR factorization starting from a user supplied value λ_0 .

```

 $\lambda = \lambda_0;$ 
for  $k = 1 : p \dots$ 
 $[Q, R] = \text{qr}([A; \sqrt{\lambda}I]);$ 
 $c = Q^T * b;$ 
 $x = R^{-1} * c;$ 
if  $k \leq p$  %update  $\lambda$ 
 $z = R^{-T} * x;$ 
 $n_x = \|x\|_2; n_z = \|z\|_2;$ 
 $\lambda = \lambda + (n_x/\gamma - 1) * (n_x/n_z)^2;$ 
end
end

```

It is slightly more efficient to initially compute the QR factorizations of A in $mn^2 - n^3/3$ multiplications. Then for each new value of λ the QR factorization of

$$Q^T(\lambda) \begin{pmatrix} R(0) \\ \sqrt{\lambda}I \end{pmatrix} = \begin{pmatrix} R(\lambda) \\ 0 \end{pmatrix}$$

can be computed in just $n^3/3$ multiplications. Then p Newton iterations will require a total of $mn^2 + (p-1)n^3/3$ multiplications. In practice $p \approx 6$ iterations usually suffice to achieve full accuracy. Further, savings are possible by initially transforming A to bidiagonal form; see Eldén [188].

8.6.5 Linear Orthogonal Regression

Let P_i , $i = 1 : m$, be a set of given points in \mathbf{R}^n . In the linear **orthogonal regression** problem we want to fit a hyper plane 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

$$c_1x + c_2y + d = 0, \quad (8.6.57)$$

where $c_1^2 + c_2^2 = 1$. Then the orthogonal distance from the point $P_i = (x_i, y_i)$ to the line equals $r_i = c_1x_i + c_2y_i + d$. Thus, we want to minimize

$$\sum_{i=1}^m (c_1x_i + c_2y_i + d)^2, \quad (8.6.58)$$

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 $c = (c_1 \ c_2)^T$ and

$$(e \ Y) = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_m & y_m \end{pmatrix}.$$

By computing the QR factorization of the matrix $(e \ Y)$ and using the invariance of the Euclidean norm this problem is reduced 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 we can always determine d so that $r_{11}d + r_{12}c_1 + r_{13}c_2 = 0$. Thus, 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 8.1.15) the solution equals the right singular vector corresponding to the smallest singular value of the matrix B . Let the SVD be

$$\begin{pmatrix} r_{21} & r_{22} \\ 0 & r_{33} \end{pmatrix} = (u_1 \ u_2) \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix},$$

where $\sigma_1 \geq \sigma_2 \geq 0$. (A stable algorithm for computing the SVD of an upper triangular matrix is given in Algorithm 9.4.2; see also Problem 9.4.5.) 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 [226, Chapter 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 $P_i \in \mathbf{R}^n$ to a hyper plane M so that the sum of squares of the orthogonal distances is minimized. The equation for the hyper plane can be written

$$c^T z = d, \quad z, c \in \mathbf{R}^n, \quad \|c\|_2 = 1,$$

where $c \in \mathbf{R}^n$ is the normal vector of M , and $|d|$ is the orthogonal distance from the origin to the plane. Then the orthogonal projections of the points y_i onto M are given by

$$z_i = y_i - (c^T y_i - d)c. \quad (8.6.59)$$

It is readily verified that the point z_i lies on M and the residual $(z_i - y_i)$ is parallel to c and hence orthogonal to M . It follows that the problem is equivalent to minimizing

$$\sum_{i=1}^m (c^T y_i - d)^2, \quad \text{subject to } \|c\|_2 = 1.$$

If we put $Y^T = (y_1, \dots, y_m) \in \mathbf{R}^{n \times m}$ and $e = (1, \dots, 1)^T \in \mathbf{R}^m$, this problem can be written in matrix form

$$\min_{c,d} \left\| \begin{pmatrix} -e & Y \end{pmatrix} \begin{pmatrix} d \\ c \end{pmatrix} \right\|_2, \quad \text{subject to } \|c\|_2 = 1. \quad (8.6.60)$$

For a fixed c , this expression is minimized when the residual vector $(Yc - de)$ is orthogonal to e , that is $e^T(Yc - de) = e^T Y c - d e^T e = 0$. Since $e^T e = m$ it follows that

$$d = \frac{1}{m} c^T Y^T e = c^T \bar{y}, \quad \bar{y} = \frac{1}{m} Y^T e, \quad (8.6.61)$$

where \bar{y} is the mean value of the given points y_i . Hence, d is determined by the condition that the mean value \bar{y} lies on the optimal plane M . Note that this property is shared by the usual linear regression.

We now subtract the mean value \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 (8.6.61)

$$\begin{pmatrix} -e & Y \end{pmatrix} \begin{pmatrix} d \\ c \end{pmatrix} = Yc - e\bar{y}^T c = (Y - e\bar{y}^T)c = \bar{Y}c,$$

problem (8.6.60) is equivalent to

$$\min_n \|\bar{Y}c\|_2, \quad \|c\|_2 = 1 \quad (8.6.62)$$

By the min-max characterization of the singular values a solution to (8.6.62) is given by $c = v_n$, 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 equals 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 and then $\bar{Y} = 0$, and any plane going through \bar{y} is a solution.

The above method solves the problem of fitting a $(n - 1)$ dimensional linear manifold to a given set of points in \mathbf{R}^n . It is readily generalized to the fitting of an $(n - p)$ dimensional manifold by orthogonalizing the shifted points y against the p right singular vectors of Y corresponding to p smallest singular values. A least squares problem that often arises is to fit to given data points a geometrical element, which may be defined in implicit form. For example, the problem of fitting circles, ellipses, spheres, and cylinders arises in applications such as computer graphics, coordinate meteorology, and statistics. Such problems are nonlinear and will be discussed in Section 9.2.6.

8.6.6 The Orthogonal Procrustes Problem

Let A and B be given matrices in $\mathbf{R}^{m \times n}$. A problem that arises, e.g., in factor analysis in statistics is the **orthogonal Procrustes**³⁶ problem

$$\min_Q \|A - BQ\|_F \quad \text{subject to} \quad Q^T Q = I. \quad (8.6.63)$$

Another example is in determining rigid body movements. Suppose that a_1, a_2, \dots, a_m are measured positions of $m \geq n$ landmarks in a rigid body in \mathbf{R}^n . Let b_1, b_2, \dots, b_m be the measured positions after the body has been rotated. We seek an orthogonal matrix $Q \in \mathbf{R}^{n \times n}$ representing the rotation of the body.

The solution can be computed from the polar decomposition of $B^T A$ as shown by the following generalization of Theorem 8.1.19 by P. Schönemann [522]:

Theorem 8.6.3.

Let $\mathcal{M}_{m \times n}$ denote the set of all matrices in $\mathbf{R}^{m \times n}$ with orthogonal columns. Let A and B be given matrices in $\mathbf{R}^{m \times n}$. If $B^T A = PH$ is the polar decomposition then

$$\|A - BQ\|_F \geq \|A - BP\|_F$$

for any matrix $Q \in \mathcal{M}_{m \times n}$.

Proof. Recall from (7.1.61) 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

$$\|A - BQ\|_F^2 = \text{trace}(A^T A) + \text{trace}(B^T B) - 2 \text{trace}(Q^T B^T A).$$

It follows that the problem (8.6.63) is equivalent to maximizing $\text{trace}(Q^T B^T A)$. Let the SVD of $B^T A$ be $B^T A = U \Sigma V^T$, where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$. Set $Q = U Z V^T$,

³⁶Procrustes was a giant of Attica in Greece who seized travelers, tied them to a bedstead, and either stretched them or chopped off their legs to make them fit it.

where Z is orthogonal. Then we have $\|Z\|_2 = 1$ and hence the diagonal elements of Z must satisfy $|z_{ii}| \leq 1$, $i = 1 : n$. Hence,

$$\begin{aligned}\text{trace}(Q^T B^T A) &= \text{trace}(VZ^T U^T B^T A) = \text{trace}(Z^T U^T B^T A V) \\ &= \text{trace}(Z^T \Sigma) = \sum_{i=1}^n z_{ii} \sigma_i \leq \sum_{i=1}^n \sigma_i.\end{aligned}$$

The upper bound is obtained for $Q = UV^T$. This solution is not unique unless $\text{rank}(A) = n$. \square

In many applications it is important that Q corresponds to a pure rotation, that is $\det(Q) = 1$. If $\det(UV^T) = 1$, the optimal is $Q = UV^T$ as before. If $\det(UV^T) = -1$, the optimal solution can be shown to be (see [311])

$$Q = UZV^T, \quad Z = \text{diag}(1, \dots, 1, -1)$$

which has determinant equal to 1. For this choice

$$\sum_{i=1}^n z_{ii} \sigma_i = \text{trace}(\Sigma) - 2\sigma_n.$$

Thus, in both cases the optimal solution can be written

$$Q = UZV^T, \quad Z = \text{diag}(1, \dots, 1, \det(UV^T)).$$

Perturbation bounds for the polar factorization are derived in Barrlund [33]. A perturbation analysis of the orthogonal Procrustes problem is given by Söderlind [537].

In analysis of rigid body movements there is also a translation $c \in R^n$ involved. Then we have the model

$$A = BQ + ec^T, \quad e = (1, 1, \dots, 1)^T \in \mathbf{R}^m,$$

where we now want to estimate also the translation vector $c \in \mathbf{R}^n$. The problem now is

$$\min_{Q,c} \|A - BQ - ec^T\|_F \quad \text{subject to } Q^T Q = I \tag{8.6.64}$$

and $\det(Q) = 1$. For any Q including the optimal Q we do not yet know, the best least squares estimate of c is characterized by the condition that the residual is orthogonal to e . Multiplying by e^T we obtain

$$0 = e^T(A - BQ - ec^T) = e^T A - (e^T B)Q - mc^T = 0,$$

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 satisfies

$$c = \frac{1}{m}((B^T e)Q - A^T e). \tag{8.6.65}$$

Substituting this expression into (8.6.64) 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 the matrix A is close to an orthogonal matrix, then an iterative method for computing the polar decomposition can be used. Such methods are developed in Section 10.8.4.

Review Questions

- 7.1** What is meant by a saddle-point system? Which two optimization problems give rise to saddle-point systems?

Problems

- 7.1** Consider the overdetermined linear system $Ax = b$ in Example 8.2.4. Assume that $\epsilon^2 \leq u$, where u is the unit roundoff, so that $fl(1 + \epsilon^2) = 1$.
- Show that the condition number of A is $\kappa = \epsilon^{-1}\sqrt{3 + \epsilon^2} \approx \epsilon^{-1}\sqrt{3}$.
 - 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

$$\text{CGS : } |q_3^T q_2| = 1/2, \quad \text{MGS : } |q_3^T q_1| = \frac{\epsilon}{\sqrt{6}} \leq \frac{\kappa}{3\sqrt{3}}u.$$

Note that for CGS orthogonality has been completely lost!

- 7.2** (Stewart and Stewart [553]).

If properly implemented, hyperbolic Householder transformations have the same good stability as the mixed scheme of hyperbolic rotations.

- Show that the hyperbolic rotations \check{G} can be rewritten as

$$\check{G} = \frac{1}{c} \begin{pmatrix} 1 & -s \\ -s & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \frac{1}{c} \begin{pmatrix} t & \\ -s/t & \end{pmatrix} (t \quad -s/t), \quad t = \sqrt{1 - c},$$

which now has the form of a hyperbolic Householder transformation. If \check{G} is J -orthogonal so is

$$J\check{G} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{c} \begin{pmatrix} t & \\ s/t & \end{pmatrix} (t \quad -s/t), \quad t = \sqrt{1 - c},$$

(b) Show that the transformation can be computed from

$$SG \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \gamma \begin{pmatrix} 1 \\ s/(1-c) \end{pmatrix}, \quad \gamma = \frac{1}{c}((1-c)x - sy).$$

7.3 Assume that $A \in \mathbf{R}^{m \times m}$ is symmetric and positive definite and $B \in \mathbf{R}^{m \times n}$ a matrix with full column rank. Show that

$$M = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & -S \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix},$$

where $S = B^T A^{-1} B$ is the Schur complement (cf. (7.1.19)). Conclude that M is indefinite! (M is called a saddle point matrix.)

8.7 The Total Least Squares Problem

8.7.1 Total Least Squares and the SVD

In the standard linear model (8.1.13) it is assumed that the vector $b \in \mathcal{R}^m$ is related to the unknown parameter vector $x \in \mathcal{R}^n$ by a linear relation $Ax = b + e$, where $A \in \mathcal{R}^{m \times n}$ is an exactly known matrix and e a vector of random errors. If the components of e are uncorrelated, have zero means and the same variance, then by the Gauss–Markov theorem (Theorem 8.1.6) the best unbiased estimate of x is obtained by solving the least squares problem

$$\min_x \|r\|_2, \quad Ax = b + r. \quad (8.7.1)$$

The assumption in the least squares problem that all errors are confined to the right hand side b is frequently unrealistic, and sampling or modeling errors often will affect also the matrix A . In the **errors-in-variables model** it is assumed that a linear relation

$$(A + E)x = b + r,$$

where the rows of the errors (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 scaling matrices $D = \text{diag}(d_1, \dots, d_m)$ and $T = \text{diag}(d_1, \dots, d_{n+1})$ such that $D(A, b)T$ satisfies this assumptions.

Estimates of the unknown parameters x in this model can be obtained from the solution of the **total least squares** (TLS) problem. The term “total least squares problem” was coined by Golub and Van Loan in [276]. The concept has been independently developed in statistics where it is known as “latent root regression”.

$$\min_{E, r} \|(r, E)\|_F, \quad (A + E)x = b + r, \quad (8.7.2)$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm defined by

$$\|A\|_F^2 = \sum_{i,j} a_{ij}^2 = \text{trace}(A^T A).$$

The constraint in (8.7.2) implies that $b + r \in \mathcal{R}(A + E)$. Thus, the 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 (8.7.2) then any x satisfying $(A + E)x = b + r$ is said to solve the TLS problem.

The TLS solution will depend on the scaling of the data (A, b) . In the following we assume that this scaling has been carried out in advance, so that any statistical knowledge of the perturbations 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**

$$\min_{E, r} \|(E, \gamma r)\|_F \quad (A + E)x = b + r.$$

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

$$\min_E \|E\|_F, \quad (A + E)x = b, \quad (8.7.3)$$

where it is assumed that the errors in the data is confined to the matrix A .

In the following we assume that $b \notin \mathcal{R}(A)$, for otherwise the system is consistent. The constraint in (8.7.2) can be written

$$(b + r \ A + E) \begin{pmatrix} -1 \\ x \end{pmatrix} = 0.$$

This constraint 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 matrix having minimal Frobenius norm, which lowers the rank of the matrix $(b \ A)$.

The total least squares problem can be analyzed in terms of the SVD

$$(b \ A) = U\Sigma V^T = \sum_{i=1}^{k+1} \sigma_i u_i v_i^T, \quad (8.7.4)$$

where $\sigma_1 \geq \dots \geq \sigma_n \geq \sigma_{n+1} \geq 0$ are the singular values of $(b \ A)$. By the minimax characterization of singular values (Theorem 8.1.17) the singular values of $\hat{\sigma}_i$ of A interlace those of $(b \ A)$, that is

$$\sigma_1 \geq \hat{\sigma}_1 \geq \sigma_2 > \dots > \sigma_n \geq \hat{\sigma}_n \geq \sigma_{n+1}. \quad (8.7.5)$$

Assume first that $\hat{\sigma}_n > \sigma_{n+1}$. Then it follows that $\text{rank}(A) = n$ and by (8.7.5) $\sigma_n > \sigma_{n+1}$. If $\sigma_{n+1} = 0$, then $Ax = b$ is consistent; otherwise by Theorem 8.1.18 the unique perturbation of minimum norm $\|(r \ E)\|_F$ that makes $(A+E)x = b+r$ consistent is the rank one perturbation

$$(r \ E) = -\sigma_{n+1} u_{n+1} v_{n+1}^T \quad (8.7.6)$$

for which $\min_{E, r} \| (r - E) \|_F = \sigma_{n+1}$. Multiplying (8.7.6) from the right with v_{n+1} gives

$$(b - A)v_{n+1} = -(r - E)v_{n+1}. \quad (8.7.7)$$

Writing the relation $(A + E)x = b + r$ in the form

$$(b - A) \begin{pmatrix} 1 \\ -x \end{pmatrix} = -(r - E) \begin{pmatrix} 1 \\ -x \end{pmatrix}$$

and comparing with (8.7.7) it is easily seen that the TLS solution can be written in terms of the right singular vector v_{n+1} as

$$x = -\frac{1}{\omega}y, \quad v_{n+1} = \begin{pmatrix} \omega \\ y \end{pmatrix}, \quad (8.7.8)$$

If $\omega = 0$ then the TLS problem has no solution. From (8.7.4) it follows that $(b - A)^T U = V\Sigma^T$ and taking the $(n+1)$ st column of both sides

$$\begin{pmatrix} b^T \\ A^T \end{pmatrix} u_{n+1} = \sigma_{n+1} v_{n+1}. \quad (8.7.9)$$

Hence, if $\sigma_{n+1} > 0$, then $\omega = 0$ if and only if $b \perp u_{n+1}$. (This case can only occur when $\hat{\sigma}_n = \sigma_{n+1}$, since otherwise the TLS problem has a unique solution.) The case when $b \perp u_{n+1}$ is called **nongeneric**. It can be treated by adding constraints on the solution; see the discussion [587].

Example 8.7.1.

For

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad E = \begin{pmatrix} 0 & 0 \\ 0 & \epsilon \\ 0 & 0 \end{pmatrix}, \quad (8.7.10)$$

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, $\hat{\sigma}_2 = \sigma_3 = 0$ and $b \perp u_3 = e_3$.

Suppose now that σ_{n+1} is a repeated singular value,

$$\sigma_p > \sigma_{p+1} = \cdots = \sigma_{n+1}, \quad p < n.$$

Set $V_2 = (v_{p+1}, \dots, v_{n+1})$, and let $V_2 z$ be any unit vector in the subspace spanned by the right singular vectors corresponding to the multiple minimal singular value. Then any vector such that

$$x = -\frac{1}{\omega}y, \quad V_2 z = \begin{pmatrix} \omega \\ y \end{pmatrix},$$

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 $\omega = e_1^T V_2 z$. Let take $z = Qe_1$, where Q is a Householder transformation such that

$$V_2 Q = \begin{pmatrix} \omega & 0 \\ y & V'_2 \end{pmatrix}$$

Then a TLS solution of minimum norm is given by (8.7.8). If $\omega \neq 0$ there is no solution and the problem is nongeneric. By an argument similar to the case when $p = n$ this can only happen if $b \perp u_j$, $j = p : n$.

8.7.2 Conditioning of the TLS Problem

We now consider the conditioning of the total least squares problem and its relation to the least squares problem. We denote those solutions by x_{TLS} and x_{LS} , respectively.

In Section 7.1.6 we showed that the SVD of a matrix A is related to the eigenvalue problem for the symmetric matrix $A^T A$. From this it follows that in the generic case the TLS solution can also be characterized by

$$\begin{pmatrix} b^T \\ A^T \end{pmatrix} (b \ A) \begin{pmatrix} -1 \\ x \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} -1 \\ x \end{pmatrix}, \quad (8.7.11)$$

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” $(b \ A)$. From the properties of the Rayleigh quotient of symmetric matrices (see Section 9.3.4) it follows that x_{TLS} is characterized by minimizing

$$\rho(x) = \frac{(b - Ax)^T (b - Ax)}{x^T x + 1} = \frac{\|b - Ax\|_2^2}{\|x\|_2^2 + 1}, \quad (8.7.12)$$

Thus, whereas the LS solution minimizes $\|b - Ax\|_2^2$ the TLS solution minimizes the “orthogonal distance” function $\rho(x)$ in (8.7.11).

From the last block row of (8.7.11) it follows that

$$(A^T A - \sigma_{n+1}^2 I)x_{TLS} = A^T b. \quad (8.7.13)$$

Note that if $\hat{\sigma}_n > \sigma_{n+1}$ then the matrix $(A^T A - \sigma_{n+1}^2 I)$ is symmetric positive definite, which ensures that the TLS problem has a unique solution. This can be compared with the corresponding normal equations for the least squares solution

$$A^T A x_{LS} = A^T b. \quad (8.7.14)$$

In (8.7.13) a positive multiple of the unit matrix is *subtracted* from the matrix $A^T A$ of normal equations. Thus, TLS can be considered as a *deregularizing* procedure. (Compare Section 8.1.5, where a multiple of the unit matrix was added to improve the conditioning.) Hence, the TLS solution is always *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 $A^T A$. Subtracting (8.7.14) from (8.7.14) we get

$$x_{TLS} - x_{LS} = \sigma_{n+1}^2 (A^T A - \sigma_{n+1}^2 I)^{-1} x_{LS}.$$

Taking norms we obtain

$$\frac{\|x_{TLS} - x_{LS}\|_2}{\|x_{LS}\|_2} \leq \frac{\sigma_{n+1}^2}{\hat{\sigma}_n^2 - \sigma_{n+1}^2}.$$

It can be shown that an approximate condition number for the TLS solution is

$$\kappa_{TLS} \approx \frac{\hat{\sigma}_1}{\hat{\sigma}_n - \sigma_{n+1}} = \kappa(A) \frac{\hat{\sigma}_n}{\hat{\sigma}_n - \sigma_{n+1}}. \quad (8.7.15)$$

When $\hat{\sigma}_n - \sigma_{n+1} \ll \hat{\sigma}_n$ the TLS condition number can be much worse than for the LS problem.

Example 8.7.2.

Consider the overdetermined system

$$\begin{pmatrix} \hat{\sigma}_1 & 0 \\ 0 & \hat{\sigma}_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \beta \end{pmatrix}. \quad (8.7.16)$$

Trivially, the LS solution is $x_{LS} = (c_1/\hat{\sigma}_1, c_2/\hat{\sigma}_2)^T$, $\|r_{LS}\|_2 = |\beta|$. If we take $\hat{\sigma}_1 = c_1 = 1$, $\hat{\sigma}_2 = c_2 = 10^{-6}$, then $x_{LS} = (1, 1)^T$ is independent of β , and hence does not reflect the ill-conditioning of A . However,

$$\kappa_{LS}(A, b) = \kappa(A) \left(1 + \frac{\|r_{LS}\|_2}{\|\hat{\sigma}_1 x_{LS}\|_2} \right)$$

will increase proportionally to β . The TLS solution is of similar size as the LS solution as long as $|\beta| \leq \hat{\sigma}_2$. However, when $|\beta| \gg \hat{\sigma}_2$ then $\|x_{TLS}\|_2$ becomes large.

In Figure 8.7.1 the two condition numbers are plotted as a function of $\beta \in [10^{-8}, 10^{-4}]$. For $\beta > \hat{\sigma}_2$ the condition number κ_{TLS} grows proportionally to β^2 . It can be verified that $\|x_{TLS}\|_2$ also grows proportionally to β^2 .

Setting $c_1 = c_2 = 0$ gives $x_{LS} = 0$. If $|\beta| \geq \sigma_2(A)$, then $\sigma_2(A) = \sigma_3(A, b)$ and the TLS problem is nongeneric.

8.7.3 Some Generalized TLS Problems

We now consider the more general TLS problem with $d > 1$ right-hand sides

$$\min_{E, F} \| (E \quad F) \|_F, \quad (A + E)X = B + F, \quad (8.7.17)$$

where $B \in \mathbf{R}^{m \times d}$. The consistency relations can be written

$$(B + F \quad A + E) \begin{pmatrix} -I_d \\ X \end{pmatrix} = 0,$$

Thus, we now seek perturbations (E, F) that reduces 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 error 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 an improved predicted power of the TLS solution.

The solution to the TLS problem with multiple right-hand sides can be expressed in terms of the SVD

$$(B \ A) = U\Sigma V^T = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T, \quad (8.7.18)$$

where

$$\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_n), \quad \Sigma_2 = \text{diag}(\sigma_{n+1}, \dots, \sigma_{n+d}),$$

and U and V partitioned conformally with $(B \ A)$. Assuming that $\sigma_n > \sigma_{n+1}$, the minimizing perturbation is unique and given by the rank d matrix

$$(F \ E) = -U_2\Sigma_2 V_2^T = -(B \ A)V_2 V_2^T,$$

for which $\|(F \ E)\|_F = \sum_{j=1}^d \sigma_{n+j}^2$ and $(B + F \ A + E)V_2 = 0$. Assume that

$$V_2 = \begin{pmatrix} V_{12} \\ V_{22} \end{pmatrix}.$$

with $V_{12} \in \mathbf{R}^{d \times d}$ nonsingular. Then the solution to the TLS problem is unique and given by

$$X = -V_{22}V_{12}^{-1} \in \mathbf{R}^{n \times d}.$$

We show that if $\sigma_n(A) > \sigma_{n+1}(B \ A)$, then V_{12} is nonsingular. From (8.7.18) it follows that $BV_{12} + AV_{22} = U_2\Sigma_2$. Now, suppose that V_{12} is singular. Then $V_{12}x = 0$ for some unit vector x . It follows that $U_2\Sigma_2x = AV_{12}x$. 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 hence $\|V_{22}x\|_2 = 1$. But then

$$\sigma_{n+1}(B \ A) \geq \|U_2\Sigma_2x\|_2 = \|AV_{12}x\|_2 \geq \sigma_n(A),$$

a contradiction. Hence, V_{12} is nonsingular.

From the above characterization it follows that the TLS solution satisfies

$$\begin{pmatrix} B^T \\ A^T \end{pmatrix} (B \ A) \begin{pmatrix} -I_d \\ -X \end{pmatrix} = \begin{pmatrix} -I_d \\ X \end{pmatrix} C, \quad (8.7.19)$$

where

$$C = V_{22}\Sigma_2^2 V_{22}^{-1} \in \mathcal{R}^{d \times d}. \quad (8.7.20)$$

Note that C is symmetrizable but not symmetric! Multiplying (8.7.19) from the left with $(I_d \ X^T)$ gives

$$(B - AX)^T(B - AX) = (X^T X + I_d)C,$$

and if $(X^T X + I_d)$ is nonsingular,

$$C = (X^T X + I_d)^{-1} (B - AX)^T (B - AX), \quad (8.7.21)$$

The multidimensional TLS solution X_{TLS} minimizes $\|C\|_F$, which generalizes the result for $d = 1$.

The last block component of (8.7.19) reads

$$A^T A X - X C = A^T B,$$

which is a Sylvester equation for X . This has a unique solution if and only if $A^T A$ and C have no common eigenvalues, which is the case if $\hat{\sigma}_n > \sigma_{n+1}$.

Now assume that $\sigma_k > \sigma_{k+1} = \dots = \sigma_{n+1}$, $k < n$, and set $V_2 = (v_{k+1}, \dots, v_{n+d})$. Let Q be a product of Householder transformations such that

$$V_2 Q = \begin{pmatrix} \Gamma & 0 \\ Z & Y \end{pmatrix},$$

where $\Gamma \in \mathbf{R}^{d \times d}$ is lower triangular. If Γ is nonsingular, then the TLS solution of minimum norm is given by

$$X = -Z\Gamma^{-1}.$$

In many parameter estimation problems, some of the columns are known exactly. It is no restriction 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

$$(A_1, A_2 + E_2)X = B + F, \quad A_1 \in \mathbf{R}^{m \times n_1},$$

where $A = (A_1, A_2) \in \mathbf{R}^{m \times n}$, $n = n_1 + n_2$. It is assumed that the rows of the errors (E_2, F) are independently and identically distributed with zero mean and the same variance. The mixed LS-TLS problem can then be expressed

$$\min_{E_2, F} \|(E_2, F)\|_F, \quad (A_1, A_2 + E_2)X = B + F. \quad (8.7.22)$$

When A_2 is empty, this reduces to solving an ordinary least squares problem. When A_1 is empty this is the standard TLS problem. Hence, this mixed problem includes both extreme cases.

The solution of the mixed LS-TLS problem can be obtained by first computing a QR factorization of A and then solving a TLS problem of reduced dimension.

Algorithm 8.11. Mixed LS-TLS problem.

Let $A = (A_1, A_2) \in \mathbf{R}^{m \times n}$, $n = n_1 + n_2$, $m \geq n$, and $B \in \mathbf{R}^{m \times d}$. Assume that the columns of A_1 are linearly independent. Then the following algorithm solves the mixed LS-TLS problem (8.7.22).

Step 1. Compute the QR factorization

$$(A_1, A_2, B) = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

where Q is orthogonal, and $R_{11} \in \mathbf{R}^{n_1 \times n_1}$, $R_{22} \in \mathbf{R}^{(n_2+d) \times (n_2+d)}$ are upper triangular. If $n_1 = n$, then the solution X is obtained by solving $R_{11}X = R_{12}$ (usual least squares); otherwise continue (solve a reduced TLS problem).

Step 2. Compute the SVD of R_{22}

$$R_{22} = U\Sigma V^T, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_{n_2+d}),$$

where the singular values are ordered in decreasing order of magnitude.

Step 3a. Determine $k \leq n_2$ such that

$$\sigma_k > \sigma_{k+1} = \dots = \sigma_{n_2+d} = 0,$$

and set $V_{22} = (v_{k+1}, \dots, v_{n_2+d})$. If $n_1 > 0$ then compute V_2 by back-substitution from

$$R_{11}V_{12} = -R_{12}V_{22}, \quad V_2 = \begin{pmatrix} V_{12} \\ V_{22} \end{pmatrix},$$

else set $V_2 = V_{22}$.

Step 3b. Perform Householder transformations such that

$$V_2 Q = \begin{pmatrix} \Gamma & 0 \\ Z & Y \end{pmatrix},$$

where $\Gamma \in \mathbf{R}^{d \times d}$ is upper triangular. If Γ is nonsingular then the solution is

$$X = -Z\Gamma^{-1}.$$

Otherwise the TLS problem is nongeneric and has no solution.

Note that the QR factorization in the first step would be the first step in computing the SVD of A .

8.7.4 Bidiagonalization and TLS Problems.

One way to avoid the complications of nongeneric problems is to compute a regular core TLS problem by bidiagonalizing of the matrix $(b \ A)$. Consider the TLS problem

$$\min_{E,r} \|(E, r)\|_F, \quad (A + E)x = b + r.$$

It was shown in Section 8.4.5 that we can always find square orthogonal matrices \tilde{U}_{k+1} and $\tilde{V}_k = P_1 P_2 \cdots P_k$ such that

$$\tilde{U}_{k+1}^T (b \ A \tilde{V}_k) = \begin{pmatrix} \beta_1 e_1 & B_k & 0 \\ 0 & 0 & A_k \end{pmatrix}, \quad (8.7.23)$$

where

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} \in \mathbf{R}^{(k+1) \times k},$$

and

$$\beta_j \alpha_j \neq 0, \quad j = 1 : k. \quad (8.7.24)$$

Setting $x = \tilde{V}_k \begin{pmatrix} y \\ z \end{pmatrix}$, the approximation problem $Ax \approx b$ then decomposes into the two subproblems

$$B_k y \approx \beta_1 e_1, \quad A_k z \approx 0.$$

It seems reasonable to simply take $z = 0$, and separately solve the first subproblem, which is the minimally dimensioned **core subproblem**. Setting

$$V_k = \tilde{V}_k \begin{pmatrix} I_k \\ 0 \end{pmatrix}, \quad U_{k+1} = \tilde{U}_{k+1} \begin{pmatrix} I_{k+1} \\ 0 \end{pmatrix},$$

it follows that

$$(b - AV_k) = U_{k+1} (\beta_1 e_1 \ B_k).$$

If $x = V_k y \in \mathcal{R}(V_k)$ then

$$(A + E)x = (A + E)V_k y = (U_{k+1} B_k + EV_k)y = \beta_1 U_{k+1} e_1 + r,$$

Hence, the consistency relation $(A + E)x = b + r$ becomes

$$(B_k + F)y = \beta_1 e_1 + s, \quad F = U_{k+1}^T EV_k, \quad s = U_{k+1}^T r. \quad (8.7.25)$$

Using the orthogonality of U_{k+1} and V_k it follows that

$$\|(E, r)\|_F = \|(F, s)\|_F. \quad (8.7.26)$$

Hence, to minimize $\|(E, r)\|_F$ we should take y_k to be the solution to the TLS core subproblem

$$\min_{F, s} \|(F, s)\|_F, \quad (B_k + F)y = \beta_1 e_1 + s. \quad (8.7.27)$$

From (8.7.24) and Theorem 8.4.3 it follows that the singular values of the matrix B_k are simple and that the right hand side $\beta_1 e_1$ has nonzero components along each left singular vector. This TLS problem therefore must have a unique solution. Note that we can assume that $\beta_{k+1} \neq 0$, since otherwise the system is compatible.

To solve this subproblem we need to compute the SVD of the bidiagonal matrix

$$(\beta_1 e_1, \ B_k) = \begin{pmatrix} \beta_1 & \alpha_1 & & & \\ & \beta_2 & \alpha_2 & & \\ & & \ddots & & \\ & & & \ddots & \alpha_k \\ & & & & \beta_{k+1} \end{pmatrix} \in \mathbf{R}^{(k+1) \times (k+1)}. \quad (8.7.28)$$

The SVD of this matrix

$$(\beta_1 e_1, B_k) = P \text{diag}(\sigma_1, \dots, \sigma_{k+1}) Q^T, \quad P, Q \in \mathbf{R}^{(k+1) \times (k+1)}$$

can be computed, e.g., by the implicit QR-SVD algorithm; see Section 9.7.6. (Note that the first stage in this is a transformation to bidiagonal form, so the work in performing the reduction (8.7.23) has not been wasted!) Then with

$$q_{k+1} = Q e_{k+1} = \begin{pmatrix} \omega \\ z \end{pmatrix}.$$

Here it is always the case that $\omega \neq 0$ and the solution to the original TLS problem (8.7.27) equals

$$x_{TLS} = V_k y = -\omega^{-1} V_k z.$$

Further, the norm of the perturbation equals

$$\min_{E, r} \| (E, r) \|_F = \sigma_{k+1}.$$

8.7.5 Solving Overdetermined Systems in the l_p Sense.

In some applications it might be more adequate to minimize some l_p -norm of the residual vector other than the l_2 norm. We consider the problem of minimizing

$$\|r\|_p = \left(\sum_{i=1}^m |r_i|^p \right)^{1/p}, \quad 1 \leq p \leq \infty. \quad (8.7.29)$$

For $1 < p < \infty$ this problem is strictly convex and hence has exactly one solution. For $p = 1$ the solution may not be unique. Minimization in the l_1 -norm or l_∞ -norm is complicated by the fact that the function $f(x) = \|Ax - b\|_p$ is only piecewise differentiable when $p = 1$ or $p = \infty$.

Example 8.7.3.

To illustrate the effect of using a different norm we consider the problem of estimating the scalar x from m observations $b \in \mathbf{R}^m$. This is equivalent to minimizing $\|Ax - b\|_p$, with $A = e = (1, 1, \dots, 1)^T$. It is easily verified that if $b_1 \geq b_2 \geq \dots \geq b_m$, then the solution x_p for some different values p are

$$\begin{aligned} x_1 &= b_{\frac{m+1}{2}}, \quad (m \text{ odd}) \\ x_2 &= \frac{1}{m}(b_1 + b_2 + \dots + b_m), \\ x_\infty &= \frac{1}{2}(b_1 + b_m). \end{aligned}$$

These estimates correspond to the median, mean, and midrange respectively. Note that the estimate x_1 is insensitive to the extreme values of b_i , while x_∞ depends

only on the extreme values. The l_∞ solution has the property that the absolute error in at least n equations equals the maximum error.

A similar effect is also achieved with p greater than but close to 1. Approximation in the maximum norm is often called Chebyshev approximation. We avoid this terminology, since Chebyshev's name is also naturally associated with two entirely different approximation methods: interpolation in the Chebyshev abscissae, and expansion in a series of Chebyshev polynomials.

To determine the l_1 solution of an overdetermined linear system $Ax = b$, $A \in \mathbf{R}^{m \times n}$, we want to find a vector x that minimizes

$$\phi(x) = \|Ax - b\|_1 = \sum_{i=1}^m |a_i^T x - b_i|. \quad (8.7.30)$$

Here a_i^T denotes the i th row of A . This problem can be cast in the form

$$\begin{aligned} & \text{minimize } e^T w = \sum_{i=1}^m w_i, \\ & \text{subject to } w_i \geq (a_i^T x - b_i) \\ & \quad \text{and } w_i \geq (b_i - a_i^T x), \quad i = 1 : m, \end{aligned}$$

where $e = (1, \dots, 1)^T$. This is a linear programming problem with $2m$ linear inequalities. In principle this problem can be solved by the simplex method; see Section 9.3.3.

The simple example above illustrates that the l_1 norm of the residual vector has the advantage of giving a **robust** solution, i.e., a small number of isolated large errors will usually not have a big effect on the solution. More generally, in robust linear regression possible outsiders among the data points are identified and given less weight. The most popular among the robust estimators is Huber's M-estimator. This uses the least squares estimator for "normal" data but the l_1 norm estimator for data points that disagree with the normal picture. More precisely, the Huber M-estimate minimizes the objective function

$$\psi(x) = \sum_{i=1}^m \rho(r_j(x)/\sigma), \quad (8.7.31)$$

where

$$\rho(t) = \begin{cases} \frac{1}{2}t^2, & \text{if } |t| \leq \gamma; \\ \gamma|t| - \frac{1}{2}\gamma^2, & \text{if } |t| > \gamma, \end{cases} \quad (8.7.32)$$

γ is a tuning parameter, and σ a scaling factor which depends on the data to be estimated. In the following we assume that σ is a constant, and then it is no restriction to take $\sigma = 1$.

Like the l_p estimator for $1 < p < 2$, the Huber estimator can be viewed as a compromise between l_2 and l_1 approximation. For large values of γ it will be close to the least squares estimator; for small values of γ it is close to the l_1 estimator.

An efficient continuation algorithm for solving the l_1 problem, which uses a finite sequence of Huber estimates with decreasing parameter γ has been developed by Madsen and Nielsen [422].

O'Leary [454, 1990] has studied different implementations of Newton-like methods. The Newton step s for minimizing $\psi(x)$ in (8.7.31) ($\sigma = 1$) is given by the solution of

$$A^T D A s = A^T y,$$

where

$$y_i = \rho'(r_i), \quad D = \text{diag}(\rho''(r_i)), \quad i = 1 : m.$$

O'Leary recommends that at the initial iterations the cutoff value γ in the Huber function (8.7.32) 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 H .

Another approach to solve the l_p norm problem when $1 < p < 3$, is the method of **iteratively reweighted least squares** (IRLS) see Osborne [458]. This amounts to solving a certain sequence of weighted least squares problems. We start by noting that, provided that $|r_i(x)| = |b - Ax|_i > 0$, $i = 1 : m$, the problem (8.7.29) can be restated in the form $\min_x \psi(x)$, where

$$\psi(x) = \sum_{i=1}^m |r_i(x)|^p = \sum_{i=1}^m |r_i(x)|^{p-2} r_i(x)^2. \quad (8.7.33)$$

This can be interpreted as a weighted least squares problem

$$\min_x \|D(r)^{(p-2)/2}(b - Ax)\|_2, \quad D(r) = \text{diag}(|r|), \quad (8.7.34)$$

where $\text{diag}(|r|)$ denotes the diagonal matrix with i th component $|r_i|$. The diagonal weight matrix $D(r)^{(p-2)/2}$ in (8.7.34) depends on the unknown solution x , but we can attempt to use the following iterative method.

Algorithm 8.12.

IRLS for l_p Approximation $1 < p < 2$

Let $x^{(0)}$ be an initial approximation such that $r_i^{(0)} = (b - Ax^{(0)})_i \neq 0$, $i = 1, \dots, n$.

```

for k = 0, 1, 2, ...
    r_i^{(k)} = (b - Ax^{(k)})_i;
    D_k = diag((|r_i^{(k)}|)^{(p-2)/2});
    solve δx^{(k)} from
        min_{δx} \|D_k(r^{(k)} - Aδx)\|_2;
    x^{(k+1)} = x^{(k)} + δx^{(k)};
end

```

Since $D_k b = D_k(r^{(k)} - Ax^{(k)})$, it follows that $x^{(k+1)}$ in IRLS solves $\min_x \|D_k(b - Ax)\|_2$, but the implementation above is to be preferred. It has been assumed that

in the IRLS algorithm, at each iteration $r_i^{(k)} \neq 0$, $i = 1, \dots, n$. In practice this cannot be guaranteed, and it is customary to modify the algorithm so that

$$D_k = \text{diag}((100ue + |r^{(k)}|)^{(p-2)/2}),$$

where u is the machine precision and $e^T = (1, \dots, 1)$ is the vector of all ones. Because the weight matrix D_k is not constant, the simplest implementations of IRLS recompute, e.g., the QR factorization of $D_k A$ in each step. It should be pointed out that the iterations can be carried out entirely in the r space without the x variables. Upon convergence to a residual vector r_{opt} the corresponding solution can be found by solving the consistent linear system $Ax = b - r_{\text{opt}}$.

It can be shown that in the l_p case any fixed point of the IRLS iteration satisfies the necessary conditions for a minimum of $\psi(x)$. The IRLS method is convergent for $1 < p < 3$, and also for $p = 1$ provided that the l_1 approximation problem has a unique nondegenerate solution. However, the IRLS method can be extremely slow when p is close to unity.

Review Questions

- 8.1** Formulate the total least squares (TLS) problem. The solution of the TLS problem is related to a theorem on matrix approximation. Which?

Problems and Computer Exercises

- 8.1** Consider a TLS problem where $n = 1$ and

$$C = (A, b) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

Show that the unique minimizing ΔC gives

$$C + \Delta C = (A + E, b + r) = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$$

so the perturbed system is not compatible, but that an arbitrary small perturbation ϵ in the (2,1) element will give a compatible system with solution $x = 2/\epsilon$.

- 8.2** (a) Let $A \in \mathbf{R}^{m \times n}$, $m \geq n$, $b \in \mathbf{R}^m$, and consider the total least squares (TLS) problem. $\min_{E, r} \|(E, r)\|_F$, where $(A + E)x = b + r$. If we have the QR factorization

$$Q^T(A, b) = \begin{pmatrix} S \\ 0 \end{pmatrix}, \quad S = \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}.$$

then the ordinary least squares solution is $x_{LS} = R^{-1}z$, $\|r\|_2 = \rho$.

Show that if a TLS solution x_{TLS} exists, then it holds

$$\begin{pmatrix} R^T & 0 \\ z^T & \rho \end{pmatrix} \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix} \begin{pmatrix} x_{TLS} \\ -1 \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} x_{TLS} \\ -1 \end{pmatrix},$$

where σ_{n+1} is the smallest singular value of (A, b) .

(b) Write a program using inverse iteration to compute x_{TLS} , i.e., for $k = 0, 1, 2, \dots$, compute a sequence of vectors $x^{(k+1)}$ by

$$\begin{pmatrix} R^T & 0 \\ z^T & \rho \end{pmatrix} \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix} \begin{pmatrix} y^{(k+1)} \\ -\alpha \end{pmatrix} = \begin{pmatrix} x^{(k)} \\ -1 \end{pmatrix}, \quad x^{(k+1)} = y^{(k+1)} / \alpha.$$

As starting vector use $x^{(0)} = x_{LS}$ on the assumption that x_{TLS} is a good approximation to x_{LS} . Will the above iteration always converge? Try to make it fail!

(c) Study the effect of scaling the right hand side in the TLS problem by making the substitution $z := \theta z$, $\rho := \theta \rho$. Plot $\|x_{TLS}(\theta) - x_{LS}\|_2$ as a function of θ and verify that when $\theta \rightarrow 0$, then $x_{TLS} \rightarrow x_{LS}$.

Hint For generating test problems it is suggested that you use the function qmult(A) from the MATLAB collection of test matrices by N. Higham to generate a matrix $C = (A, b) = Q_1 * D * Q_2^T$, where Q_1 and Q_2 are random real orthogonal matrices and D a given diagonal matrix. This allows you to generate problems where C has known singular values and vectors.

Notes and Further References

Modern numerical methods for solving least squares problems are surveyed in the two comprehensive monographs [399] and [61]. The latter contains a bibliography of 860 references, indicating the considerable research interest in these problems. Hansen [310] gives an excellent survey of numerical methods for the treatment of numerically rank deficient linear systems arising, for example, from discrete ill-posed problems.

Several of the great mathematicians at the turn of the 19th century worked on methods for solving overdetermined linear systems. Laplace in 1799 used the principle of minimizing the sum of absolute errors $|r_i|$, with the added conditions that the errors sum to zero. This leads to a solution x that satisfies at least n equations exactly. The method of least squares was first published as an algebraic procedure by Legendre 1805 in [401]. Gauss justified the least squares principle as a statistical procedure in [233], where he claimed to have used the method since 1795. This led to one of the most famous priority dispute in the history of mathematics. Gauss further developed the statistical aspects in 1821–1823. For an interesting accounts of the history of the invention of least squares, see Stiegler [555, 1981].

Because of its success in analyzing astronomical data the method of least squares rapidly became the method of choice when analyzing observation. Geodetic calculations was another early area of application of the least squares principle. In the last decade applications in control and signal processing has been a source of inspiration for developments in least squares calculations.

Section 8.1

Gauss [232] in 1821 put the method of least squares on a sound theoretical basis. In his textbook from 1912 Markov [424] refers to Gauss' work and may have clarified some implicit assumptions but proves nothing new; see Placket [487, 488].

A detailed account of the interesting early history of the SVD is given by Stewart [547, 1993]. Beltrami [42]³⁷ in 1873 derived the SVD for a real, square, nonsingular matrix having distinct singular values. A year later Jordan [356] independently published a derivation of the SVD which handled multiple singular values. Jordan also stated the variational characterization of the largest singular value as the maximum of a function. Auton [16] extended the SVD to complex matrices and Eckart and Young [183] to rectangular matrices. The relation to the polar decomposition is due to Autonne [16]. Picard [486] seems to have been the first to call the numbers σ_k singular values.

Both Beltrami and Jordan were concerned with diagonalizing a finite bilinear form $P = x^T A y$. Schmidt [521] developed in 1907 the infinite-dimensional analogue of the SVD as a tool for solving integral equations. He used it to obtain a low-rank approximation to the integral operator. Weyl [607] developed a general perturbation theory and gave a more elegant proof of Schmidt's approximation theorem.

The first stable algorithm for computing the SVD the singular value was developed by Golub, Reinsch, and Wilkinson in the late 1960's. Several applications of the SVD to matrix approximation can be found in Golub and Van Loan [277, Sec. 12.4].

Theorem 8.1.2 can be generalized to the semidefinite case, see Gulliksson and Wedin [294, Theorem 3.2]. A case when B is indefinite and nonsingular is considered in Section 8.6.2.

A good introduction to generalized inverses is given by Ben-Israel and Greenville [43, 1976]. Generalized inverses should be used with caution since the notation tends to hide intrinsic computational difficulties associated with rank deficient matrices. A more complete and thorough treatment is given in the monograph by the same authors [44, 2003]. The use of generalized inverses in geodetic calculations is treated in Bjerhammar [56, 1973].

Section 8.2

Peters and Wilkinson [484, 1970] developed methods based on Gaussian elimination from a uniform standpoint and the excellent survey by Noble [451, 1976]. Sautter [520, 1978] gives a detailed analysis of stability and rounding errors of the LU algorithm for computing pseudo-inverse solutions. For a fuller treatment of weighted least squares and the general linear model, see Björck [61, Chap.3].

How to find the optimal backward error for the linear least squares problem was an open problem for many years, until it was elegantly answered by Karlsson et al. [598]; see also [368]. Gu [287] gives several approximations to that are optimal up

³⁷Eugenio Beltrami (1835–1900), was born in Cremona, Italy, which then belonged to Austria. He studied applied mathematics in Pavia and Milan. In 1864 was appointed to the chair of geodesy at the University of Pisa and from 1966 he was professor of rational mechanics in Bologna. In 1973 he moved to Rome, which in 1870 had become the new capital of Italy. After three years there he moved back to Pavia. Beltrami contributed to work in differential geometry on curves and surfaces and gave a concrete realization of the non-euclidian geometry.

to a factor less than 2. Optimal backward perturbation bounds for underdetermined systems are derived in [560]. The extension of backward error bounds to the case of constrained least squares problems is discussed by Cox and Higham [121].

Section 8.3

Jacobi [350] used Givens rotations already in 1845 to achieve diagonal dominance in systems of normal equations and then applying a simple iterative scheme that became known as Jacobi's method. See Section 10.1.

Complex Givens rotations and complex Householder transformations are treated in detail by Wilkinson [611, pp. 47–50]. Lehoucq [402, 1996] gives a comparison of different implementations of complex Householder transformations. The reliable construction of real and complex Givens rotations are considered in great detail in Bindel, Demmel and Kahan [53]. Wilkinson [611] showed the backward stability of algorithm based on a sequence of Householder transformations.

The different computational variants of Gram–Schmidt have an interesting history. The “modified” Gram–Schmidt (MGS) algorithm was in fact already derived by Laplace in 1816 as an elimination method using weighted row sums. Laplace did not interpret his algorithm in terms of orthogonalization, nor did he use it for computing least squares solutions! Bienaymé in 1853 gave a similar derivation of a slightly more general algorithm; see Björck [60, 1994]. What is now called the “classical” Gram–Schmidt (CGS) algorithm first appeared explicitly in papers by Gram 1883 and Schmidt 1908. Schmidt treats the solution of linear systems with infinitely many unknowns and uses the orthogonalization as a theoretical tool rather than a computational procedure.

In the 1950’s algorithms based on Gram–Schmidt orthogonalization were frequently used, although their numerical properties were not well understood at the time. Björck [57] analyzed the modified Gram–Schmidt algorithm and showed its stability for solving linear least squares problems.

The systematic use of orthogonal transformations to reduce matrices to simpler form was initiated by Givens [257] and Householder [341, 1958]. The application of these transformations to linear least squares is due to Golub [261, 1965], where it was shown how to compute a QR factorization of A using Householder transformations. An Algol implementation of this method was given in [87].

Section 8.4

An efficient algorithm for solving rank-deficient least squares problem is given by Foster and Kommu [216]. This uses a truncated pivoted QR factorization where the rank of the trailing diagonal block is estimated by a condition estimator. A different approach to the subset selection problem has been given by de Hoog and Mattheij [141]. They consider choosing the square subset A_1 of rows (or columns), which maximizes $\det(A_1)$.

Matlab templates for computing UTV decompositions has been given by Fierro, Hansen, and Hansen [202].

Partial least squares originated in statistical applications, specifically economy Herman Wold [615]. It became very popular in chemometrics, where it was introduced by Svante Wold; see [616]. Now it is becoming a method of choice also in a large number of applications in the social sciences. Unfortunately the statisticians

use a different implementation, the numerical stability of which has not be proved.

Section 8.5

The QR algorithm for banded rectangular matrices was first given by Reid [498]. Rank-revealing QR (RRQR) factorizations have been studied by a number of authors. A good survey can be found in Hansen [310]. The URV and ULV decompositions were introduced by Stewart [546, 548].

For a detailed discussion of dissection and orthogonal factorizations in geodetic survey problems, see Golub and Plemmons [260].

This block triangular form (8.5.18) 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 [180].

The row sequential sparse QR algorithm employing Givens rotations is due to George and Heath [236, 1980]. Liu [412] introduces the notion of *row merge tree* for sparse QR factorization by Givens rotations. This row merging scheme can be viewed as the implicitly using the multifrontal method on $A^T A$. This algorithm can give a significant reduction in QR factorization time at a modest increase in working storage. George and Liu [240] give a modified version of Liu's algorithm, which uses Householder transformations instead of Givens rotations. For other multifrontal codes developed for sparse QR factorizations, see Matstoms [430]. ,Supernodes and other modifications of the multifrontal method are discussed by Liu [413] and Matstoms [429].

Section 8.6

An early reference to the exchange operator is in network analysis; see the survey of Tsatsomeros [579]. J -orthogonal matrices also play a role in the solution of the generalized eigenvalue problem $Ax = \lambda Bx$; see Section 10.7. For a systematic study of J -orthogonal matrices and their many applications we refer to Higham [330]. An error analysis of Chamber's algorithm is given by Bojanczyk et al. [70].

The systematic use of GQR as a basic conceptual and computational tool are explored by [464]. These generalized decompositions and their applications are discussed in [11]. Algorithms for computing the bidiagonal decomposition are due to Golub and Kahan [265, 1965]. The partial least squares (PLS) method, which has become a standard tool in chemometrics, goes back to Wold et al. [616].

The term “total least squares problem”, which was coined by Golub and Van Loan [276], renewed the interest in the “errors in variable model”. A thorough and rigorous treatment of the TLS problem is found in Van Huffel and Vandewalle [587]. The important role of the core problem for weighted TLS problems was discovered by Paige and Strakoš [472].

Section 8.7

For the the l_1 problem algorithms which use linear programming techniques have been developed by Barrodale and Roberts [34]. The Harwell Subroutine Library uses a version of their algorithm. Other algorithms for this problem based on projected gradient techniques are given by Bartels, Conn, and Sinclair [35] A general treatment of robust statistical procedures is given by Huber [344]. A great deal of work has been devoted to developing methods for computing the Huber M-

estimator; see, e.g., Clark and Osborne [112, 1986], Ekblom [186, 1988], and Madsen and Nielsen [421, 1990]

Chapter 9

Nonlinear and Constrained Problems

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, Algorithms for Scientific Computation. Communications of the ACM (1966).

9.1 Systems of Nonlinear Equations

Many problems can be written in generic form as a **nonlinear system** of equations.

$$f_i(x_1, x_2, \dots, x_n) = 0, \quad i = 1 : n. \quad (9.1.1)$$

or using the vector notations, more compactly,

$$f(x) = 0, \quad f : \mathbf{R}^n \rightarrow \mathbf{R}^n. \quad (9.1.2)$$

In this section we consider the numerical solution of such systems. More generally, f can be an operator acting on some function space, in which case (9.1.1) is called a **functional equation**. Applications where nonlinear systems arise include initial and boundary value problems for nonlinear differential equations, and nonlinear integral equations.

The problem of finding *all* solutions of equation (9.1.1) in some subregion $\mathcal{B} \subset \mathbf{R}^n$ can be a very difficult problem. Note that in \mathbf{R}^n there is no efficient method, like the bisection method, that can be used as a global method to get initial approximations. In general, we must therefore be content with finding local solutions, to which reasonable good initial approximations are known.

If $m > n$ in (9.1.1) then we have have an overdetermined nonlinear system. This leads to the **nonlinear least squares problem**

$$\min_{x \in \mathbf{R}^n} = \frac{1}{2} \|f(x)\|_2^2, \quad (9.1.3)$$

This is an optimization problem, where the function to be minimized has a special form. Methods for this problem are described in Section 9.2.2.

9.1.1 Calculus in Vector Spaces

We shall introduce some notions and notations from the calculus in vector spaces that will be useful in this chapter. A more general and rigorous treatment can be found, e.g., in Dieudonné [163], where the reader may find some proofs that we omit here. There are in the literature several different notations for these matters beside vector-matrix notation, e.g., **multilinear mapping** notation, **tensor** notation. None of them seems to be perfect or easy to handle correctly in some complex situations. This may be a reason to become familiar with several notations.

Consider $k+1$ vector spaces X_1, X_2, \dots, X_k, Y , and let $x_\nu \in X_\nu$. A function $A: X_1 \times X_2 \times \dots \times X_k \rightarrow Y$ is called **k -linear**, if it is linear in each of its arguments x_i separately. For example, the expression $(Px_1)^T Qx_2 + (Rx_3)^T Sx_4$ defines a 4-linear function, mapping or operator (provided that the constant matrices P, Q, R, S have appropriate size). If $k = 2$ such a function is usually called **bilinear**, and more generally one uses the term **multilinear**.

Let $X_\nu = \mathbf{R}^{n_\nu}$, $\nu = 1 : k$, $Y = \mathbf{R}^m$, and let e_{j_i} be one of the basis vectors of X_i . We use *superscripts* to denote coordinates in these spaces. Let $a_{j_1, j_2, \dots, j_k}^i$ denote the i th coordinate of $A(e_{j_1}, e_{j_2}, \dots, e_{j_k})$. Then, because of the linearity, the i th coordinate of $A(x_1, x_2, \dots, x_k)$ reads

$$\sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \dots \sum_{j_k=1}^{n_k} a_{j_1, j_2, \dots, j_k}^i x_1^{j_1} x_2^{j_2} \dots x_k^{j_k}, \quad x_\nu \in X_\nu. \quad (9.1.4)$$

In the **sum convention** of tensor analysis, if an index occurs both as a subscript and as a superscript the product should be summed over the range of this index. Thus, the i th coordinate of $A(x_1, x_2, \dots, x_k)$ reads shorter

$$a_{j_1, j_2, \dots, j_k}^i x_1^{j_1} x_2^{j_2} \dots x_k^{j_k}.$$

(Remember always that here the superscripts are no exponents.)

Suppose that $X_i = X$, $i = 1 : k$. Then, the set of k -linear mappings from X^k to Y is itself a linear space called $L_k(X, Y)$. For $k = 1$, we have the space of linear functions, denoted more shortly by $L(X, Y)$. Linear functions can, of course, also be described in vector-matrix notation; $L(\mathbf{R}^n, \mathbf{R}^m) = \mathbf{R}^{m \times n}$, the set of matrices defined in Section 7.1.1. Matrix notation can also be used for each coordinate of a bilinear function. These matrices are in general unsymmetric.

Norms of multilinear operators are defined analogously to subordinate matrix norms. For example,

$$\|A(x_1, x_2, \dots, x_k)\|_\infty \leq \|A\|_\infty \|x_1\|_\infty \|x_2\|_\infty \dots \|x_k\|_\infty, \quad (9.1.5)$$

where

$$\|A\|_\infty = \max_{i=1:m} \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \dots \sum_{j_k=1}^{n_k} |a_{j_1, j_2, \dots, j_k}^i|. \quad (9.1.6)$$

A multilinear function A is called *symmetric* if $A(x_1, x_2, \dots, x_k)$ is symmetric with respect to its arguments. In the cases mentioned above, where matrix notation can be used, the matrix becomes symmetric, if the multilinear function is symmetric.

We next consider a function $f: X \rightarrow Y$, not necessarily multilinear, where X and Y are normed vector spaces. This function is *continuous*, at the point $x_0 \in X$ if $\|f(x) - f(x_0)\| \rightarrow 0$ as $\|x - x_0\| \rightarrow 0$.

Definition 9.1.1.

Let X and Y be normed vector spaces. The function $f: X \rightarrow Y$ is differentiable at x_0 , in the sense of Fréchet, if there exists a linear mapping A such that

$$\|f(x) - f(x_0) - A(x - x_0)\| = o(\|x - x_0\|), \quad x \rightarrow x_0. \quad (9.1.7)$$

This linear mapping is called the **Fréchet derivative** of f at x_0 , and we write $A = f'(x_0)$ or $A = f_x(x_0)$.

Note that (the value of) $f'(x_0) \in L(X, Y)$. Considered as a function of x_0 , $f'(x_0)$ is, of course, usually non-linear. A **differential** reads, in the multilinear mapping notation, $df = f'dx$ or $df = f_x dx$. In tensor notation with the sum convention, it reads $df^i = f_j^i dx^j$. These definitions apply also to infinite dimensional spaces. In the finite dimensional case, the Fréchet derivative is represented by the **Jacobian** matrix,

$$f'(x) = J(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \in \mathbf{R}^{n \times n}, \quad (9.1.8)$$

the elements of which are the partial derivatives $\partial f^i / \partial x^j$. These are also written f_j^i , in an established notation, e.g., in tensor analysis; superscripts for coordinates and subscripts for partial derivation.

If vector-matrix notation is used, it is important to note that the derivative g' of a real-valued function g is a *row* vector, since

$$g(x) = g(x_0) + g'(x_0)(x - x_0) + o(\|x - x_0\|).$$

We suggest that the notation **gradient** is used for the transpose of $g'(x)$

$$\nabla g(x) = \left(\frac{\partial g}{\partial x_1}, \dots, \frac{\partial g}{\partial x_n} \right)^T \in \mathbf{R}^n \quad (9.1.9)$$

Many results from elementary calculus carry over to vector space calculus, such as the rules for the differentiation of products. The proofs are in principle the same.

If $z = f(x, y)$ where $x \in \mathbf{R}^k$, $y \in \mathbf{R}^l$, $z \in \mathbf{R}^m$, then we define *partial derivatives* f_x, f_y with respect to the vectors x, y by the differential formula

$$df(x, y) = f_x dx + f_y dy, \quad \forall dx \in \mathbf{R}^k, \quad dy \in \mathbf{R}^l. \quad (9.1.10)$$

If x, y are functions of $s \in \mathbf{R}^n$, then a general version of the *chain rule* reads

$$f'(x(s), y(s)) = f_x x'(s) + f_y y'(s). \quad (9.1.11)$$

can be derived in the same way as for real valued variables. The extension to longer chains is straightforward. These equations can also be used in infinite dimensional spaces.

Consider a function $f: \mathbf{R}^k \rightarrow \mathbf{R}^k$, and consider the equation $x = f(y)$. By formal differentiation, $dx = f'(y)dy$, and we obtain $dy = (f'(y))^{-1}dx$, provided that the Jacobian $f'(y)$ is non-singular. Another important example: if $f(x, y) = 0$ then, by (9.1.11), $f_x dx + f_y dy = 0$. If $f_y(x_0, y_0)$ is a non-singular matrix, then, by the *implicit function theorem* (see Dieudonné [163, Sec. 10.2]) y becomes, under certain additional conditions, a differentiable function of x in a neighborhood of (x_0, y_0) , and we obtain $dy = -(f_y)^{-1}f_x dx$, hence

$$y'(x) = -(f_y)^{-1}f_x|_{y=y(x)}.$$

One can also show that

$$\lim_{\epsilon \rightarrow +0} \frac{f(x_0 + \epsilon v) - f(x_0)}{\epsilon} = f'(x_0)v.$$

There are, however, functions f , where such a **directional derivative** exists for any v but, for some x_0 , is not a linear function of v . An important example is $f(x) = \|x\|_\infty$, where $x \in \mathbf{R}^n$. (Look at the case $n = 2$.) The name **Gateaux derivative** is sometimes used in such cases, in order to distinguish it from the Fréchet derivative $f'(x_0)$ previously defined.

If $f'(x)$ is a differentiable function of x at the point x_0 , its derivative is denoted by $f''(x_0)$. This is a linear function that maps X into the space $L(X, Y)$ that contains $f'(x_0)$, i.e.,

$$f''(x_0) \in L(X, L(X, Y)).$$

This space may be identified in a natural way with the space $L_2(X, Y)$ of bilinear mappings $X^2 \rightarrow Y$; if $A \in L(X, L(X, Y))$ then the corresponding $\bar{A} \in L_2(X, Y)$ is defined by $(Au)v = \bar{A}(u, v)$ for all $u, v \in X$; in the future it is not necessary to distinguish between A and \bar{A} . So,

$$f''(x_0)(u, v) \in Y, \quad f''(x_0)u \in L(X, Y), \quad f''(x_0) \in L_2(X, Y).$$

It can be shown that $f''(x_0): X^2 \rightarrow Y$, is a symmetric bilinear mapping, i.e., $f''(x_0)(u, v) = f''(x_0)(v, u)$. The second order partial derivatives are denoted f_{xx} , f_{xy} , f_{yx} , f_{yy} .

If $X = \mathbf{R}^n$, $Y = \mathbf{R}^m$, $m > 1$, $f''(x_0)$ reads $f_{ij}^p(x_0) = f_{ji}^p(x_0)$ in tensor notation. It is thus characterized by a three-dimensional array, which one rarely needs to store or write. Fortunately, most of the numerical work can be done on a lower level, e.g., with directional derivatives. For each fixed value of p we obtain a symmetric $n \times n$ matrix, named the **Hessian** matrix

$$H_p = \begin{pmatrix} f_{11}^p & \dots & f_{1n}^p \\ \vdots & & \vdots \\ f_{n1}^p & \dots & f_{nn}^p \end{pmatrix} \in \mathbf{R}^{n \times n}. \quad (9.1.12)$$

Note that

$$f''(x_0)(u, v) = u^T H(x_0)v.$$

If the Hessian exists and are continuous then it is *symmetric*, i.e., $f_{xy} = f_{yx}$. The Hessian can be looked upon as the derivative of the gradient. An element of this Hessian is, in the multilinear mapping notation, the p th coordinate of the vector $f''(x_0)(e_i, e_j)$.

We suggest that the vector-matrix notation is replaced by the multilinear mapping formalism when handling derivatives of vector-valued functions of order higher than one. The latter formalism has the further advantage that it can be used also in infinite-dimensional spaces. In finite dimensional spaces the tensor notation with the summation convention is another alternative.

Higher derivatives are recursively defined. If $f^{(k-1)}(x)$ is differentiable at x_0 , then its derivative at x_0 is denoted $f^{(k)}(x_0)$ and called the k th derivative of f at x_0 . One can show that $f^{(k)}(x_0) : X^k \rightarrow Y$ is a *symmetric* k -linear mapping. The fundamental Taylor's formula then admits the following elegant generalization:

Theorem 9.1.2 (Taylor's Formula).

Let $f : X \rightarrow Y$ be a function having a finite derivative of order $k + 1$ and let when $a, u \in X$. Then Taylor's formula reads

$$f(a + u) = f(a) + f'(a)u + \frac{1}{2}f''(a)u^2 + \dots + \frac{1}{k!}f^{(k)}(a)u^k + R_{k+1}, \quad (9.1.13)$$

where

$$R_{k+1} = \int_0^1 \frac{(1-t)^k}{k!} f^{(k+1)}(a + ut)u^{k+1} dt, \quad (9.1.14)$$

and

$$\|R_{k+1}\| \leq \frac{\|u\|^{k+1}}{(k+1)!} \max_{0 \leq t \leq 1} \|f^{(k+1)}(a + ut)\|. \quad (9.1.15)$$

After some hesitation, we use here u^2 , u^k , etc. as abbreviations for the lists of input vectors (u, u) , (u, u, \dots, u) etc.

Proof. Define a new function of one real variable t by setting

$$g(t) = f(a + tu), \quad 0 \leq t \leq 1.$$

Then $g(0) = f(a)$ and $g(1) = f(a + u)$. The theorem is proved by applying the one dimensional Mclaurin formula to g ; see Vol. I, Section 3.1.2. This gives

$$g(1) = g(0) + g'(0) + \frac{1}{2}g''(0) + \dots + \frac{1}{k!}g^{(k)}(0) + R_{k+1}, \quad (9.1.16)$$

where the integral form of the remainder is

$$R_{k+1} = \int_0^1 \frac{(1-t)^k}{k!} g^{(k+1)}(t) dt. \quad (9.1.17)$$

Now g is a composite function given by $g(t) = f(p(t))$, where $p(t) = a+tu$. Applying the chain rule we see that g' exists in $0 \leq t \leq 1$ and given by $g'(0) = f'(a)u$. Again applying the chain rule we obtain

$$g^{(p)}(0) = f^{(p)}(a)u^p, \quad 1 \leq p \leq k+1.$$

Substitution in (9.1.16)–(9.1.17) yields the theorem. \square

The simplified notation used in the theorem exemplifies what you may allow yourself (and us) to use when you have got a good hand with the notation and its interpretation. Abbreviations that reduce the number of parentheses often increase the clarity; there may otherwise be some risk for ambiguity, since parentheses are used around the arguments for both the usually non-linear function $f^{(k)}: X \rightarrow L_k(X, Y)$ and the k -linear function $f^{(k)}(x_0): X^k \rightarrow Y$. You may also write, e.g., $(f')^3 = f'f'f'$; beware that you do not mix up $(f')^3$ with f'' .

The mean value theorem of differential calculus and Lagrange's form for the remainder of Taylor's formula are not true, but they can in many places be replaced by the above *integral form of the remainder*. All this holds in complex vector spaces too.

9.1.2 The Contraction Mapping Theorem

We will now generalize the theory of fixed-point iteration developed in Volume I, Section 6.1.4 for a single nonlinear equation. Consider the nonlinear equation $f(x) = 0$, where $f: \mathbf{R}^n \rightarrow \mathbf{R}^n$. This can be rewritten in the form $x = g(x)$, which suggests an iterative method where, for $k = 0, 1, 2, \dots$, we compute

$$x_{k+1} = g(x_k), \quad k = 0, 1, 2, \dots \quad (9.1.18)$$

This is known as a **fixed point iteration**. If g is continuous and $\lim_{k \rightarrow \infty} x_k = x^*$, then $x^* = g(x^*)$ and x^* solves the system $x = g(x)$. (Recall that a vector sequence is said to converge to a limit x^* if $\lim_{k \rightarrow \infty} \|x_k - x^*\| = 0$ for some norm $\|\cdot\|$.)

Example 9.1.1.

The nonlinear system

$$\begin{aligned} x^2 - 2x - y + 1 &= 0 \\ x^2 + y^2 - 1 &= 0 \end{aligned}$$

defines the intersection between a circle and a parabola. The two real roots are $(1, 0)$ and $(0, 1)$. Taking $x_0 = 0.9$ and $y_0 = 0.2$ and using the following fixed point iteration

$$\begin{aligned} x_{k+1} &= (y_k - 1)/(x_k - 2), \\ y_{k+1} &= 1 - x_k^2/(y_k + 1), \end{aligned}$$

we obtain the results

k	x_k	y_k
1	0.72727273	0.32500000
2	0.53035714	0.60081085
3	0.27162323	0.82428986
4	0.10166194	0.95955731
5	0.02130426	0.99472577
6	0.00266550	0.99977246
7	0.00011392	0.99999645

Note that although we started close to the root $(1, 0)$ the sequence converges to the other real root $(0, 1)$. (See also Problem 9.1.1.)

We now derive sufficient conditions for the convergence of the fixed point iteration (9.1.18). The following theorem not only provides a solid basis for iterative numerical techniques, but also is an important tool in theoretical analysis. Note that, *the existence of a fixed point is not assumed a priori*. We first recall some definitions.

Definition 9.1.3.

A sequence of points $\{x_n\}$ in a metric space \mathcal{S} is said to **converge** to a limit $x^* \in \mathcal{S}$ if $d(x_n, x^*) \rightarrow 0$. As $n \rightarrow \infty$, we write $x_n \rightarrow x^*$ or $\lim_{n \rightarrow \infty} x_n = x^*$. A sequence $\{x_n\}$ in \mathcal{S} is called a **Cauchy sequence** if for every $\epsilon > 0$ there is an integer $N(\epsilon)$ such that $d(x_m, x_n) < \epsilon$ for all $m, n \geq N(\epsilon)$. Every convergent sequence is a Cauchy sequence, but the converse is not necessarily true. \mathcal{S} is called a **complete space** if every Cauchy sequence in \mathcal{S} converges to a limit in \mathcal{S} .

It is well known that \mathbf{R}^n satisfies the characterization of a complete space.

Definition 9.1.4.

The function f satisfies a **Lipschitz condition** in a domain $D \subset X$, if a constant γ , called a **Lipschitz constant**, can be chosen so that

$$\|f(x') - f(x'')\| \leq \gamma \|x' - x''\| \quad \forall x', x'' \in D. \quad (9.1.19)$$

If $\gamma < 1$ then f is called a **contraction mapping**.

The existence of a Lipschitz condition is somewhat more general than a differentiability condition and the Lipschitz constant for $f(x)$ can be expressed in terms of the derivative $f'(x)$.

Lemma 9.1.5.

Let function $f : \mathbf{R}^n \rightarrow \mathbf{R}^n$ be differentiable in a convex set $\mathcal{D} \subset \mathbf{R}^n$. Then

$$\gamma = \max_{y \in \mathcal{D}} \|f'(y)\|$$

is a Lipschitz constant for f .

Proof. Let $0 \leq t \leq 1$ and consider the function $g(t) = f(a + t(x - a))$, $a, x \in \mathcal{D}$. By the chain rule $g'(t) = f'(a + t(x - a))(x - a)$ and

$$f(x) - f(a) = g(1) - g(0) = \int_0^1 g'(t)dt = \int_0^1 f'(a + t(x - a))(x - a)dt.$$

Since \mathcal{D} is convex the whole line segment between the points a and x belongs to \mathcal{D} . Applying the triangle inequality we obtain

$$\|f(x) - f(a)\| < \int_0^1 \|f'(a + t(x - a))\| \|x - a\| dt \leq \max_{y \in \mathcal{D}} \|f'(y)\| \|x - a\|$$

(remember that an integral is the limit of a sum) \square

Theorem 9.1.6 (The Contraction Mapping Theorem).

Let $T : E \rightarrow F$, where $E = F = \mathbf{R}^n$, be an iteration function, and $S_r = \{u \mid \|u - u_0\| < r\}$ be a ball of radius r around a given starting point $u_0 \in \mathbf{R}^n$. Assume that T is a contraction mapping in S_r , i.e.,

$$u, v \in S_r \Rightarrow \|T(u) - T(v)\| \leq \gamma \|u - v\|, \quad (9.1.20)$$

where $\gamma < 1$. Then if

$$\|u_0 - T(u_0)\| \leq (1 - \gamma)r \quad (9.1.21)$$

the equation $u = T(u)$ has a unique solution u^* in the closure $\overline{S_r} = \{u \mid \|u - u_0\| \leq r\}$. This solution can be obtained by the convergent iteration process $u_{k+1} = T(u_k)$, $k = 0, 1, \dots$, and we have the error estimate

$$\|u_k - u^*\| \leq \|u_k - u_{k-1}\| \frac{\gamma}{1 - \gamma} \leq \|u_1 - u_0\| \frac{\gamma^k}{1 - \gamma}. \quad (9.1.22)$$

Proof. We first prove the uniqueness. If there were two solutions u' and u'' , we would get $u' - u'' = T(u') - T(u'')$ so that

$$\|u' - u''\| = \|T(u') - T(u'')\| \leq \gamma \|u' - u''\|.$$

Since $\gamma < 1$, it follows that $\|u' - u''\| = 0$, i.e., $u' = u''$.

By (9.1.21) we have $\|u_1 - u_0\| = \|T(u_0) - u_0\| \leq (1 - \gamma)r$, and hence $u_1 \in S_r$. We now use induction to prove that $u_n \in S_r$ for $n < j$, and that

$$\|u_j - u_{j-1}\| \leq \gamma^{j-1}(1 - \gamma)r, \quad \|u_j - u_0\| \leq (1 - \gamma^j)r.$$

We already know that these estimates are true for $j = 1$. Using the triangle inequality and (9.1.20) we get

$$\begin{aligned} \|u_{j+1} - u_j\| &= \|T(u_j) - T(u_{j-1})\| \leq \gamma \|u_j - u_{j-1}\| \leq \gamma^j(1 - \gamma)r, \\ \|u_{j+1} - u_0\| &\leq \|u_{j+1} - u_j\| + \|u_j - u_0\| \leq \gamma^j(1 - \gamma)r + (1 - \gamma^j)r \\ &= (1 - \gamma^{j+1})r. \end{aligned}$$

This proves the induction step, and it follows that the sequence $\{u_k\}_{k=0}^{\infty}$ stays in S_r . We also have for $p > 0$

$$\begin{aligned}\|u_{j+p} - u_j\| &\leq \|u_{j+p} - u_{j+p-1}\| + \cdots + \|u_{j+1} - u_j\| \\ &\leq (\gamma^{j+p-1} + \cdots + \gamma^j)(1 - \gamma)r \leq \gamma^j(1 - \gamma^p)r \leq \gamma^j r,\end{aligned}$$

and hence $\lim_{j \rightarrow \infty} \|u_{j+p} - u_j\| = 0$. The sequence $\{u_k\}_{k=0}^{\infty}$ therefore is a Cauchy sequence, and since \mathbf{R}^n is complete has a limit u^* . Since $u_j \in S_r$ for all j it follows that $u^* \in \overline{S_r}$.

Finally, by (9.1.20) T is continuous, and it follows that $\lim_{k \rightarrow \infty} T(u_k) = T(u^*) = u^*$. The demonstration of the error estimates (9.1.22) is left as exercises to the reader. \square

Theorem 9.1.6 holds also in a more general setting, where $T : S_r \rightarrow \mathcal{B}$, and \mathcal{B} is a **Banach space**, i.e. a normed vector space which is complete. The proof goes through with obvious modifications. In this form the theorem can be used, e.g., to prove existence and uniqueness for initial value problems for ordinary differential equations.

The Lipschitz constant γ is a measure of the rate of convergence of a fixed-point iteration. Convergence is in general linear; in each iteration the upper bound for the norm of the error is multiplied by a factor at most equal to γ . It is useful to have a precise measure of the asymptotic rate of convergence for a vector sequence converging to a limit point.

Definition 9.1.7.

A convergent sequence $\{x_k\}$ with $\lim_{k \rightarrow \infty} \{x_k\} = x^*$, and $x_k \neq x^*$, is said to have **order of convergence** equal to p ($p \geq 1$), if

$$\lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} = C, \quad (9.1.23)$$

where $|C| < 1$ for $p = 1$ and $|C| < \infty$, $p > 1$. C is called the **asymptotic error constant**. The sequence has exact convergence order p if (9.1.23) holds with $C \neq 0$. We say the convergence is **superlinear** if $C = 0$ for some $p \geq 1$.

Note that for *finite dimensional* vector sequences, the order of convergence does not depend on the choice of norm, and that the definitions are consistent with those for scalar sequences; see Volume I, Def. 6.1.6. More detailed discussions of convergence rates are found, e.g., in Dennis and Schnabel [156, pp. 19–21], and Ortega and Rheinboldt [456, Chap. 9].

9.1.3 Newton's Method and Its Variants

Consider the nonlinear system of equations $f(x) = 0$ in several variables, where f is twice continuously differentiable. Let x_k be the current approximation to a

solution.³⁸ Taylor's formula (9.1.2) for $k = 1$ gives

$$f(x_k + h) = f(x_k) + f'(x_k)h + R_2 = 0, \quad (9.1.24)$$

where $f'(x_k)$ is the Jacobian matrix evaluated at x_k , and

$$\|R_2\| \leq \frac{\|h\|^2}{2} \max_{0 \leq t \leq 1} \|f''(x_k + ht)\|.$$

Neglecting higher order terms in (9.1.24), we get the equation

$$f(x_k) + f'(x_k)h = 0,$$

The next iterate is taken to be $x_{k+1} = x_k + h_k$, where the correction h_k , satisfies the linear system

$$f'(x_k)h_k = -f(x_k). \quad (9.1.25)$$

If $f'(x_k)$ is nonsingular, then (9.1.25) has a unique solution h_k and the resulting algorithm can then be written

$$x_{k+1} = x_k - f'(x_k)^{-1}f(x_k). \quad (9.1.26)$$

which is **Newton's method**. Of course, the inverse Jacobian matrix should not be explicitly computed. Instead the Newton correction is determined by solving the linear system (9.1.25), e.g., using Gaussian elimination. If n is very large and the Jacobian sparse it may be preferable to use an iterative method. In this case x_k is a natural initial approximation.

Example 9.1.2.

The nonlinear system

$$\begin{aligned} x^2 + y^2 - 4x &= 0 \\ y^2 + 2x - 2 &= 0 \end{aligned}$$

has a solution close to $x_0 = 0.5$, $y_0 = 1$. The Jacobian matrix is

$$J(x, y) = \begin{pmatrix} 2x - 4 & 2y \\ 2 & 2y \end{pmatrix},$$

and Newton's method becomes

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - J(x_k, y_k)^{-1} \begin{pmatrix} x_k^2 + y_k^2 - 4x_k \\ y_k^2 + 2x_k - 2 \end{pmatrix}.$$

We get the results:

k	x_k	y_k
1	0.35	1.15
2	0.35424528301887	1.13652584085316
3	0.35424868893322	1.13644297217273
4	0.35424868893541	1.13644296914943

³⁸In the following we use vector notations so that x_k will denote the k th approximation and not the k th component of x .

All digits are correct in the last iteration. The quadratic convergence is obvious; the number of correct digits approximately doubles in each iteration.

We will give rigorous conditions for the quadratic convergence of Newton's method. and also shows that Newton's method converges under weak conditions provided that the initial approximation x_0 is chosen sufficiently close to a solution x^* .

It will be assumed that $\|f'(x)\|$ satisfies a Lipschitz condition and we first show the following.

Lemma 9.1.8.

Let $f : \mathbf{R}^n \rightarrow \mathbf{R}^n$ be continuously differentiable in a convex region S and assume that

$$\|f'(x) - f'(y)\| \leq \gamma \|x - y\| \quad \forall x, y \in S.$$

Then, we have the estimate

$$\|f(x) - f(y) - f'(y)(x - y)\| \leq \frac{\gamma}{2} \|x - y\|^2 \quad \forall x, y \in S.$$

Proof. Introduce the function $g(t) = f(y + t(x - y))$ of a single real variable t . By the chain rule the function g is differentiable for all $0 \leq t \leq 1$ and $g'(t) = f'(y + t(x - y))(x - y)$. It follows that $\|g'(t) - g'(0)\| \leq \gamma t \|x - y\|^2$, and further

$$\Delta := f(x) - f(y) - f'(y)(x - y) = g(1) - g(0) - g'(0) = \int_0^1 (g'(t) - g'(0)) dt.$$

This yields the inequality

$$\|\Delta\| \leq \int_0^1 \|g'(t) - g'(0)\| dt \leq \|x - y\|^2 \int_0^1 t dt = \frac{\gamma}{2} \|x - y\|^2.$$

□

Theorem 9.1.9.

Let C be a given open convex set and $f : C \rightarrow \mathbf{R}^n$ a function which is differentiable for all $x \in C$. Assume that $f(x^*) = 0$, for $x^* \in C$. Let positive constants r, β, γ be given such that

- (a) $\|f'(x) - f'(y)\| \leq \gamma \|x - y\|, \quad \forall x, y \in S_r(x^*)$,
- (b) $f'(x^*)^{-1}$ exists and $\|f'(x^*)^{-1}\| \leq \beta$,

where

$$S_r(x^*) = \{x \mid \|x - x^*\| < r\} \subseteq C.$$

Then there exists an $\epsilon > 0$ such that for all $x_0 \in S_\epsilon(x^*)$ the sequence generated by

$$x_{k+1} = x_k - f'(x_k)^{-1} f(x_k), \quad k = 0, 1, \dots$$

is well defined, $\lim_{n \rightarrow \infty} = x^*$, and satisfies

$$\|x_{k+1} - x^*\| \leq \beta\gamma\|x_k - x^*\|^2.$$

Proof. We choose $\epsilon = \min\{r, 1/(2\beta\gamma)\}$. Then by (a) and (b) we have

$$\|f'(x^*)^{-1}(f'(x_0) - f'(x^*))\| \leq \beta\gamma\|x_0 - x^*\| \leq \beta\gamma\epsilon \leq 1/2.$$

From this and Corollary 7.5.2 it follows that

$$\|f'(x_0)^{-1}\| \leq \frac{1}{1 - 1/2} \|f'(x^*)^{-1}\| = 2\beta.$$

Thus x_1 is well defined and

$$\begin{aligned} x_1 - x^* &= x_0 - x^* - f'(x_0)^{-1}(f(x_0) - f(x^*)) \\ &= f'(x_0)^{-1}(f(x^*) - f(x_0) - f'(x_0)(x^* - x_0)). \end{aligned}$$

Taking norms and using Lemma 9.1.8 we get

$$\begin{aligned} \|x_1 - x^*\| &\leq \|f'(x_0)^{-1}\| \|f(x^*) - f(x_0) - f'(x_0)(x^* - x_0)\| \\ &\leq 2\beta\gamma/2\|x_0 - x^*\|^2, \end{aligned}$$

The proof follows by induction. \square

We remark that a result by Kantorovich [366] shows quadratic convergence under weaker conditions. In particular, it is not necessary to assume the existence of a solution or the nonsingularity of $f'(x)$ at the solution. For a discussion and proof of these results we refer to Ortega and Rheinboldt [456, Chap. 12.6].

An important observation is that Newton's method is affine invariant in the following sense. Let $A \in \mathbf{R}^{n \times n}$ be an arbitrary nonsingular matrix and consider the nonlinear equation

$$g(x) = Af(x) = 0.$$

Then $g'(x) = Af'(x)$ and since

$$g'(x)^{-1}g(x) = (Af'(x))^{-1}Af(x) = f'(x)^{-1}f(x)$$

the Newton step is independent of A . The above convergence proof is not given in affine invariant terms. An affine invariant version of the Newton–Kantorovich theorem is proved by Deuflhard [158]. In particular, the conditions (a) and (b) in the theorem can be replaced by an affine invariant condition

$$\|f'(x)^{-1}(f'(x) - f'(y))(x - y)\| \leq \omega\|x - y\|^2, \quad \forall x, y \in C.$$

Each step of Newton's method requires the evaluation of the n^2 entries of the Jacobian matrix $f'(x_k)$, and the solution of the resulting linear system $n^3/3$ arithmetic operations are needed. This may be a time consuming task if n is large.

In many situations it might be preferable to reevaluate $f'(x_k)$ only occasionally using the same Jacobian in $m > 1$ steps,

$$f'(x_p)(x_{k+1} - x_k) = -f(x_k), \quad k = p : p + m - 1. \quad (9.1.27)$$

Once we have computed the LU factorization of $f'(x_p)$ the linear system can be solved in n^2 arithmetic operations. The motivation for this approach is that if either the iterates or the Jacobian matrix are not changing too rapidly $f'(x_p)$ is a good approximation to $f'(x_k)$. (These assumptions do not usually hold far away from the solution, and may cause divergence in cases where the unmodified algorithm converges.)

The modified Newton method can be written as a fixed point iteration with

$$g(x) = x - f'(x_p)^{-1}f(x), \quad g'(x) = I - f'(x_p)^{-1}f'(x).$$

We have, using assumptions from Theorem 9.1.9

$$\|g'(x)\| \leq \|f'(x_p)^{-1}\| \|f'(x_p) - f'(x)\| \leq \beta\gamma\|x_p - x\|.$$

Since $g' \neq 0$ the modified Newton method will only have *linear* rate of convergence. Also, far away from the solution the modified method may diverge in cases where Newton's method converges.

Example 9.1.3.

Consider the nonlinear system in Example 9.1.2. Using the modified Newton method with fixed Jacobian matrix evaluated at $x_1 = 0.35$ and $y_1 = 1.15$

$$J(x_1, y_1) = \begin{pmatrix} 2x_1 - 4 & 2y_1 \\ 2 & 2y_1 \end{pmatrix} = \begin{pmatrix} -3.3 & 2.3 \\ 2.0 & 2.3 \end{pmatrix}.$$

we obtain the result

k	x_k	y_k
1	0.35	1.15
2	0.35424528301887	1.13652584085316
3	0.35424868347696	1.13644394786146
4	0.35424868892666	1.13644298069439
5	0.35424868893540	1.13644296928555
6	0.35424868893541	1.13644296915104

Modifications for Global Convergence

Under certain assumptions Newton's method is locally convergent from a sufficiently good initial approximation. In general, it is not globally convergent, i.e., it does not converge from an arbitrary starting point. Far away from a solution Newton's method may not be well behaved, e.g., the Jacobian matrix can be illconditioned

or even singular. This is a serious drawback since it is often difficult to find a good starting point in \mathbf{R}^n .

We now discuss techniques to modify Newton's method, which attempt to ensure better global convergence, i.e., convergence from a large set of starting approximations. One strategy is to seek modifications which will make

$$\phi(x) = \frac{1}{2}\|f(x)\|_2^2 = \frac{1}{2}f(x)^T f(x). \quad (9.1.28)$$

decrease at each step. (Note that this is not an affine invariant condition.)

We call d a **descent direction** for $\phi(x)$ if $\phi(x + \alpha d) < \phi(x)$, for all sufficiently small $\alpha > 0$. This will be the case if the directional derivative is negative, i.e.

$$\nabla\phi(x)^T d = f(x)^T J(x)d < 0.$$

The steepest-descent direction $-g = -\nabla\phi(x) = -J(x)^T f(x)$ is the direction in which $\phi(x)$ decreases most rapidly.

Assuming that $J(x)$ is nonsingular, the Newton direction $h = -J(x)^{-1}f(x)$ is also a descent direction if $f(x) \neq 0$ since

$$\nabla\phi(x)^T h = -f(x)^T J(x)J(x)^{-1}f(x) = -\|f(x)\|_2^2 < 0.$$

In the **damped Newton** method we take

$$x_{k+1} = x_k + \alpha_k h_k, \quad J(x_k)h_k = -f(x_k). \quad (9.1.29)$$

where the step length α_k is computed by a **line search**. Ideally α_k should be chosen to minimize the scalar function

$$\psi(\alpha) = \|f(x_k + \alpha h_k)\|_2^2.$$

Algorithms for solving such an one-dimensional minimization are discussed in Volume I, Section 6.4. In practice this problem need not be solved accurately. It is only necessary to ensure that the reduction $\|f(x_k)\|_2^2 - \|f(x_{k+1})\|_2^2$ is sufficiently large.

In the **Armijo–Goldstein criterion** α_k is taken to be the largest number in the sequence $1, \frac{1}{2}, \frac{1}{4}, \dots$ for which

$$\psi(0) - \psi(\alpha_k) \geq \frac{1}{2}\alpha_k\psi'(0)$$

is satisfied. Close to a simple zero x^* this criterion will automatically chose $\alpha_k = 1$. It then becomes identical to Newton's method and convergence becomes quadratic. Another common choice is to require that α_k satisfies the two conditions

$$\psi(\alpha_k) \leq \psi(0) + \mu\alpha_k\psi'(0), \quad |\psi'(\alpha_k)| \leq \eta|\psi'(0)|$$

where typically $\mu = 0.001$ and $\eta = 0.9$. The first condition ensures a sufficient decrease in $\|f(x)\|_2^2$ and the second that the gradient is decreased by a significant amount.

The addition of line searches to the Newton iteration greatly increases the range of nonlinear equations that can successfully be solved. However, if the Jacobian $J(x_k)$ is nearly singular, then h_k determined by (9.1.29) will be large and the linear model

$$f(x_k + \alpha_k h_k) \approx f(x_k) + \alpha_k J(x_k)h_k$$

inadequate. In this case the Newton direction tends to be very inefficient.

The idea in **trust region methods** is to avoid using a linear model outside its range of validity, see also Section 9.2.3. Here one takes $x_{k+1} = x_k + d_k$, where d_k solves the constrained linear least squares problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2^2, \text{ subject to } \|d_k\|_2 \leq \Delta_k,$$

where Δ_k is a parameter, which is updated recursively. If the constraint is binding, this problem is modified by introducing a Lagrange parameter λ and considering the unconstrained problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2^2 + \lambda \|d_k\|_2^2. \quad (9.1.30)$$

Here λ is determined by the **secular equation** $\|d_k(\lambda)\|_2 = \Delta_k$. This problem is equivalent to the linear least squares problem

$$\min_{d_k} \left\| \begin{pmatrix} f(x_k) \\ 0 \end{pmatrix} + \begin{pmatrix} J(x_k) \\ \lambda^{1/2} I \end{pmatrix} d_k \right\|_2^2,$$

where the matrix always has full column rank for $\lambda > 0$.

A typical rule for updating Δ_{k+1} is to first calculate the ratio ρ_k of $\|f(x_k)\|_2^2 - \|f(x_k + d_k)\|_2^2$ to the reduction $\|f(x_k)\|_2^2 - \|f(x_k) + J(x_k)d_k\|_2^2$ predicted by the linear model. Then we take

$$\Delta_{k+1} = \begin{cases} \frac{1}{2}\|d_k\|, & \text{if } \rho_k \leq 0.1; \\ \Delta_k, & \text{if } 0.1 < \rho_k \leq 0.7; \\ \max\{2\|d_k\|, \Delta_k\}, & \text{if } \rho_k > 0.7. \end{cases}$$

The trust region is made smaller if the model is unsuccessful and is increased if a substantial reduction in the objective function is found. A difference to line search methods is that if $\Delta_{k+1} < \Delta_k$ we set $x_{k+1} = x_k$.

A related idea is used in **Powell's hybrid method**, where a linear combination of the steepest descent and the Newton (or the quasi-Newton) direction is used. Powell takes

$$x_{k+1} = x_k + \beta_k d_k + (1 - \beta_k)h_k, \quad 0 \leq \beta_k \leq 1,$$

where h_k is the Newton direction in (9.1.29), and

$$d_k = -\mu_k g_k, \quad g_k = J(x_k)^T f(x_k), \quad \mu_k = \|g_k\|_2^2 / \|J(x_k)g_k\|_2^2.$$

The choice of β_k is monitored by a parameter Δ_k , which equals the maximum allowed step size. The algorithm also includes a prescription for updating Δ_k . Powell chooses x_{k+1} as follows:

- i. If $\|h_k\|_2 \leq \Delta_k$ then $x_{k+1} = x_k + h_k$.
- ii. If $\|g_k\|_2 \leq \Delta_k \leq \|h_k\|_2$, choose $\beta_k \in (0, 1]$ so that $\|x_{k+1} - x_k\|_2 = \Delta_k$.
- iii. Otherwise set $x_{k+1} = x_k + \Delta_k g_k / \|g_k\|_2$.

The convergence is monitored by $\|f(x_k)\|_2$. When convergence is slow, Δ_k can be decreased, giving a bias towards steepest descent. When convergence is fast, Δ_k is increased, giving a bias towards the Newton direction.

Global methods for nonlinear systems may introduce other problems not inherent in the basic Newton method. The modification introduced may lead to slower convergence and even lead to convergence to a point where the equations are not satisfied.

9.1.4 Derivative Free Methods

In many applications the Jacobian matrix is not available or too expensive to evaluate. Then we can use the **discretized Newton method**, where each of the derivative elements in $J(x_k)$ is discretized separately by a difference quotient. There are many different variations depending on the choice of discretization. A frequently used approximation for the j th column of $J(x)$ is the forward difference quotient

$$\frac{\partial f(x)}{\partial x_j} \approx \Delta_j f(x) \equiv \frac{f(x + h_j e_j) - f(x)}{h_j}, \quad j = 1 : n, \quad (9.1.31)$$

where e_j denotes the j th unit vector and $h_j > 0$ is a suitable scalar. If the resulting approximation is denoted by $J(x, D)$, then we can write

$$J(x, D) = (f(x + h_1 e_1) - f(x), \dots, f(x + h_n e_n) - f(x)) D^{-1},$$

where $D = \text{diag}(h_1, h_2, \dots, h_n)$ is a nonsingular diagonal matrix. This shows that $J(x_k, D)$ is nonsingular if and only if the vectors

$$f(x_k + h_j e_j) - f(x_k), \quad j = 1 : n,$$

are linearly independent.

It is important that the step sizes h_j are chosen carefully. If h_j is chosen too large then the derivative approximation will have a large truncation error; if it is chosen too small then roundoff errors may be dominate (cf. numerical differentiation). As a rule of thumb one should choose h_j so that $f(x)$ and $f(x + h_j e_j)$ have roughly the first half digits in common, i.e.,

$$|h_j| \|\Delta_j f(x)\| \approx u^{1/2} \|f(x)\|.$$

In the discretized Newton method the vector function $f(x)$ needs to be evaluated at $n + 1$ points, including the point x_k . Hence it requires $n^2 + n$ component function evaluations per iteration. Methods which only require $(n^2 + 3n)/2$

component function evaluations have been proposed by Brown (1966) and Brent (1973). Brent's method requires the computation of difference quotients

$$\frac{f(x + h_j q_k) - f(x)}{h_j}, \quad j = 1 : n, \quad k = j : n,$$

where $Q = (q_1, \dots, q_n)$ is a certain orthogonal matrix determined by the method. Note that because of common subexpressions, in some applications a component function evaluation may be almost as expensive as a vector function evaluation. In such cases the original Newton method is still to be preferred.

Taking $h_j = f_j(x_k)$ in (9.1.31) gives a generalization of **Steffensen's method**; see Volume I, Section 6.2.3. In this method $f(x)$ is evaluated at the $(n+1)$ points x_k and

$$x_k + f_j(x_k)e_j, \quad j = 1 : n.$$

Thus when $n > 1$ the number of function evaluations for Steffensen's method is the same as for the discretized Newton's method, but the order of convergence equals two. A generalization of Steffensen's method to nonlinear operator in Banach space is given in [355], where conditions are given for convergence, uniqueness and the existence of a fixed point.

When derivatives are difficult to evaluate one can also resort to numerical differentiation. With the multilinear mapping formalism, the general case of vector-valued dependent and independent variables becomes almost as simple as the scalar case. Let η be a small positive number. By Taylor's formula (Theorem 9.1.2) we get

$$f'(x_0)v = \frac{f(x_0 + \eta v) - f(x_0 - \eta v)}{2\eta} + R_T, \quad R_T \approx -\frac{\eta^2 f'''(x_0)v^3}{6}, \quad (9.1.32)$$

Here, as previously, we use v^3 as an abbreviation for the list (v, v, v) of vector arguments. The Jacobian $f'(x_0)$ is obtained by the application of this to $v = e_j$, $j = 1 : k$. If the Jacobian has a band structure, then it can be computed by means of fewer vectors v ; see Problem 9.1.3.

First note that (9.1.32) is exact if f is a quadratic function. In this case η can be chosen rather large, so the rounding error causes no trouble either. Suppose that the rounding error of $g(x)$ is (approximately) bounded by $\epsilon\|g\|/\eta$. (The norms here are defined on a neighborhood of x_0 .) The total error is therefore (approximately) bounded by

$$B(\eta) = \|g\| \frac{\epsilon}{\eta} + \|g'''\| \|v\|^3 \frac{\eta^2}{6}.$$

Set $\|g\|/\|g'''\| = \xi^3$, and note that ξ measures a local length scale of the variation of the function g , (if we interpret x as a length). A good choice of η is found by straightforward optimization:

$$\min_{\eta} B(\eta) = (3\epsilon)^{2/3} \|g\| \|v\| / (2\xi), \quad \eta \|v\| = (3\epsilon)^{1/3} \xi. \quad (9.1.33)$$

For $\epsilon = 10^{-16}$, we should choose $\eta\|v\| = 7 \cdot 10^{-6}\xi$. The error estimate becomes $2.5 \cdot 10^{-11}\|g\|\|v\|/\xi$. In many applications this accuracy is higher than necessary. If uncentered differences are used instead of centered differences, the error becomes $O(\epsilon^{1/2})$ with optimal choice of η , while the amount of computation may be reduced by almost 50%; see Problem 9.1.1.

It may be a little cumbersome to estimate ξ by its definition, but since we need a very rough estimate only, we can replace it by some simpler measure of the length scale of $g(x)$, e.g. a rough estimate of (say) $\frac{1}{2}\|g\|/\|g'\|$.³⁹ Then for $\epsilon = 10^{-16}$ the error estimate simplifies to

$$(3\epsilon)^{2/3}\|g'\|\|v\| \approx 5 \cdot 10^{-11}\|g'\|\|v\|.$$

This is usually an overestimate, though not always.

The result of a similar study of the directional *second derivative* reads

$$\begin{aligned} f''(x_0)v^2 &= \frac{f(x_0 + \eta v) - 2f(x_0) + f(x_0 - \eta v)}{\eta^2} + R_T, \\ R_T &\approx -\frac{\eta^2 f^{iv}(x_0)v^4}{12}, \\ B(\eta) &= \|f\| \frac{4\epsilon}{\eta^2} + \frac{\|f^{iv}\| \|v\|^4 \eta^2}{12}, \\ \xi &= (\|f\|/\|f^{iv}\|)^{1/4} \approx \left(\frac{1}{3}\|f\|/\|f''\|\right)^{1/2}, \\ \min_{\eta} B(\eta) &= 2(\epsilon/3)^{1/2}\|f\| \|v\|^2 / \xi^2 \approx \epsilon^{1/2}\|f''\| \|v\|^2, \quad \eta\|v\| = (48\epsilon)^{1/4}\xi. \end{aligned} \tag{9.1.34}$$

Note that:

- if g is a cubic function, there is no truncation error, and $\eta\|v\|$ can be chosen independent of ϵ . Otherwise, for $\epsilon = 10^{-16}$, we should choose $\eta\|v\| \approx 3 \cdot 10^{-4}\xi$. The simplified error estimate becomes $2 \cdot 10^{-8}\|f''\|\|v\|^2$;
- if $f'(x)$ is available, we can obtain $f''(x_0)v^2$ more accurately by setting $g(x) = f'(x)v$ into (9.1.32), since the value of η can then usually be chosen smaller;
- if $f(x)$ is a quadratic form, then $f''(x)$ is a constant bilinear operator and $f''(x)v^2 = f(v)$. If f is a non-homogeneous quadratic function, its affine part must be subtracted from the right hand side;
- in order to compute $f''(x_0)(u, v)$, it is sufficient to have a subroutine for $f''(x_0)v^2$, since the following formula can be used. It is easily derived by the bilinearity and symmetry of $f''(x_0)$.

$$f''(x_0)(u, v) = \frac{1}{4}(f''(x_0)(u+v)^2 - f''(x_0)(u-v)^2) \tag{9.1.35}$$

Numerical differentiation should in general be avoided, when function values are subject to irregular errors. However, the harmful effect of rounding errors in the

³⁹The factor $\frac{1}{2}$ is a safety factor. So is the factor $\frac{1}{3}$ in the equation for ξ in the group (9.1.34).

context of numerical differentiation should not be exaggerated. We shall see that for most purposes a satisfactory accuracy in the first and second derivatives can be achieved provided the step size is chosen appropriately.

The goal of *symbolic* differentiation is to obtain analytic *expressions* for derivatives of functions given in analytic form. This is handled by computer algebra systems, for example, Maple or Mathematica. In contrast, the goal of **automatic differentiation** is to extend an algorithm (a program) for the computation of the *numerical values* of a few functions to an algorithm that also computes the *numerical values* of a few derivatives of these functions, without truncation errors. Such techniques are used in optimization for the computation of Jacobian and Hessian matrices. At the time of this writing, lively and active research is being performed on theory, software development, and applications of automatic differentiation.

If the function $f(x)$ is complicated to evaluate even the above method may be too expensive. In the methods above we obtain the next approximation x_{k+1} by a step along the direction h_k , computed by solving the linear system

$$B_k h_k = -f(x_k), \quad (9.1.36)$$

where B_k is an approximation to the Jacobian $J(x_k)$. The class of **quasi-Newton methods** can be viewed as a generalization of the secant method to functions of more than one variable. The approximate Jacobian B_{k+1} is required to satisfy the **secant equation**

$$B_{k+1} s_k = y_k \quad (9.1.37)$$

where s_k and y_k are the vectors

$$s_k = x_{k+1} - x_k, \quad y_k = f(x_{k+1}) - f(x_k).$$

This means that B_{k+1} correctly imitates the Jacobian along the direction of change s_k . Of course many matrices satisfy this condition.

In the very successful **Broyden's method** it is further required that the difference $B_{k+1} - B_k$ has minimal Frobenius norm. It is left as an exercise to verify that these conditions lead to

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T}{s_k^T s_k}. \quad (9.1.38)$$

This is generally referred to as Broyden's “good” updating formula. Note that $B_{k+1} - B_k$ is a *matrix of rank one*, and that $B_{k+1}p = B_k p$ for all vectors p such that $p^T(x_{k+1} - x_k) = 0$. (To generate an initial approximation B_1 we can use finite differences along the coordinate directions.)

We can compute B_{k+1} from (9.1.38) in only $O(n^2)$ flops without any new function evaluations. To solve the linear system (9.1.36) for the quasi-Newton direction still seems to require $O(n^3)$ operations. Assume that the QR factorization $B_k = Q_k R_k$ has been computed in the previous step. In Section 8.3.5 it was shown how to stably update a QR factorization subject to a rank one change in $O(n^2)$ operation. We write

$$B_{k+1} = Q_k(R_k + u_k v_k^T),$$

where

$$u_k = Q_k^T(y_k - B_k s_k), \quad v_k = s_k / \|s_k\|_2^2$$

Hence if the QR factorization $R_k + u_k v_k^T = P_k R_{k+1}$, then

$$B_{k+1} = Q_{k+1} R_{k+1}, \quad Q_{k+1} = Q_k P_k.$$

It can be shown that Broyden's modification of Newton's method has superlinear convergence.

Theorem 9.1.10.

Let $f(x) = 0$, $f : \mathbf{R}^n \rightarrow \mathbf{R}^n$, be sufficiently smooth, and let x^* be a regular zero point of f . Let

$$x_{k+1} = x_k - B_k^{-1} f(x_k)$$

be the Newton type method where B_k is updated according to Broyden's formula (9.1.38). If x_0 is sufficiently close to x^* , and B_0 sufficiently close to $f'(x_0)$, then the sequence $\{x_k\}$ is defined and converges superlinearly to x^* , i.e.,

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \rightarrow 0, \quad n \rightarrow \infty.$$

Proof. See Dennis and Moré [155]. \square

If the Jacobian $f'(x)$ is sparse the advantages of Broyden's method is lost, since the update in general is not sparse. One possibility is then to keep the LU factors of the most recently computed sparse Jacobian and save several Broyden updates as pairs of vectors $y_k - B_k s_k$ and s_k .

9.1.5 Parametrized Nonlinear Systems

In many applications one has to solve not just a single nonlinear system, but a family of systems depending on a parameter

$$F(x, \lambda) = 0, \quad \lambda \in \mathbf{R}^p. \quad (9.1.39)$$

In many problems the parameter is a scalar, e.g., the time $\lambda = t$. We restrict our attention to this situation (see Deuflhard [158, Sec. 4.4].)

One source of parametrized system is the following. Suppose it is hard to solve the nonlinear system $f(x) = 0$. For example, we cannot find an initial approximation that works. Then, an idea is to find a simpler system $g(x) = 0$, for which the solution $x = x_0$ can be obtained without difficulty. We then define a convex embedding (or **homotopy**) by setting

$$H(x, t) = t f(x) + (1 - t) g(x), \quad (9.1.40)$$

so that

$$H(x, 0) = g(x), \quad H(x, 1) = f(x).$$

If the functions $f(x)$ and $g(x)$ are sufficiently smooth then a solution curve $x(t)$ exists, which satisfies the conditions $x(0) = x_0$, and $x(1) = x^*$. One now attempts to trace the solution curve $x(t)$ of (9.1.40) by computing $x(t_j)$ for an increasing sequence of values of t , $0 = t_0 < t_1 < \dots < t_p = 1$ by solving the nonlinear systems

$$H(x, t_{j+1}) = 0, \quad j = 0 : p - 1, \quad (9.1.41)$$

by some appropriate method. The starting approximations can be obtained from previous results, e.g.,

$$x_0(t_{j+1}) = x(t_j),$$

or, if $j \geq 1$, by linear interpolation

$$x_0(t_{j+1}) = x(t_j) + \frac{t_{j+1} - t_j}{t_j - t_{j-1}} (x(t_j) - x(t_{j-1})).$$

This technique can be used in connection with any of the methods previously mentioned. For example, Newton's method can be used to solve (9.1.41)

$$x_{k+1} = x_k - \left(\frac{\partial H(x_k, t_{j+1})}{\partial x} \right)^{-1} H(x_k, t_{j+1}).$$

The step size should be adjusted automatically to approximately minimize the total number of iterations. A simpler strategy is to choose the number of increments M and take a constant step $\Delta t = 1/M$. If m is sufficiently large, then the iterative process will generally converge. However, the method may fail when turning points of the curve with respect of parameter t are encountered. In this case the embedding family has to be changed, or some other special measure must be taken. Poor performance can also occur because t is not well suited for parametrization. Often the arclength s of the curve provides a better parametrization

Embedding has important applications to the nonlinear systems encountered when finite-difference or finite-element methods are applied to nonlinear boundary-value problems; see Chapter 14. It is also an important tool in nonlinear optimization, e.g., in interior point methods. Often a better choice than (9.1.40) can be made for the embedding, where the systems for $t_j \neq 1$ also contribute to the insight into the questions which originally lead to the system. In elasticity, a technique called **incremental loading** is used, because $t = 1$ may correspond to an unloaded structure for which the solution is known, while $t = 0$ correspond to the actual loading. The technique is also called the **continuation** method.

If the equation $H(x, t) = 0$ is differentiated we obtain

$$\frac{\partial H}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial H}{\partial t} = 0.$$

This gives the differential equation

$$\frac{dx}{dt} = F(x, t), \quad F(x, t) = -\left(\frac{\partial H}{\partial x} \right)^{-1} \frac{\partial H}{\partial t}.$$

Sometimes it is recommended to use a numerical method to integrate this differential equation with initial value $x(1) = x_1$ to obtain the solution curve $x(s)$, and in particular $x(1) = x^*$. However, to use a general purpose method for solving the differential equation is an unnaturally complicated approach. One should instead numerically integrate (9.1.41) very coarsely and then locally use a Newton-type iterative method for solving (9.1.40) as a corrector. This has the advantage that one takes advantage of the fact that the solution curve consists of solutions of (9.1.41), and uses the resulting strong contractive properties of Newton's method. Such predictor corrector continuation methods have been very successful. The following algorithm uses Euler's method for integration as a predictor step and Newton's method as a corrector:

Algorithm 9.1. *The Euler–Newton Method.*

Assume that $g(x_0) = 0$. Let $t_0 = 0$, and $h_0 > 0$ be an initial step length.

```

 $x := x_0; \quad t_1 = t_0 + h_0;$ 
for  $j = 1, 2, \dots,$ 
     $x_j := x_{j-1} + h_{j-1}F(x_{j-1}, t_{j-1}); \quad$  Euler step
    repeat
         $x_j := x_j - (H'(x_j, t_j))^{-1}H(x_j, t_j); \quad$  Newton step
    until convergence
    if  $t_j \equiv 1$  then stop
    else  $t_{j+1} = t_j + h_j; \quad h_j > 0; \quad$  new steplength
    end

```

Note the possibility of using the same Jacobian in several successive steps. The convergence properties of the Euler–Newton Method and other predictor-corrector methods are discussed in Allgower and Georg [6].

Review Questions

- 1.1 (a) Define the Fréchet derivative at $x_0 \in X$ of a function $f : X \rightarrow Y$.
 (b) For the finite dimensional space, what is meant by the gradient, Jacobian, and Hessian of a vector valued function f ?
- 1.2 If the function f is differentiable how can the Lipschitz constant for f be expressed in terms of the derivative $f'(x)$?
- 1.3 Formulate the Contraction Mapping theorem. You don't need to work out the full proof, but tell where in the proof the completeness is needed.

- 1.4** (a) What is meant by the completeness of a space. What is a Banach space?
- 1.5** Describe the nonlinear Gauss–Seidel method.
- 1.6** Describe Newton’s method for solving a nonlinear system of equations.
- 1.7** In order to get global convergence Newton’s method has to be modified. Two different approaches are much used. Describe the main features of these modifications.
- 1.8** For large n the main cost of an iteration step in Newton’s method is the evaluation and factorizing of the matrix of first derivatives. Describe some ways to reduce this cost.
- 1.9** Give the essential features of the assumptions needed in the theorem in the text which is concerned with the convergence of Newton’s method for a nonlinear system. What is the order of convergence for simple roots?
- 1.10** Describe the essential features of numerical continuation methods for solving a nonlinear system $f(x) = 0$. How is a suitable convex embedding constructed?

Problems

- 1.1** The fixed point iteration in Example 9.1.1 can be written $u_{k+1} = \phi(u_k)$, where $u = (x, y)^T$. Compute $\|\phi(u^*)\|_\infty$ for the two roots $u^* = (1, 0)^T$ and $(0, 1)^T$, and use the result to explain the observed convergence behavior.
- 1.2** Consider the system of equations
- $$\begin{aligned} x_1^2 - x_2 + \alpha &= 0, \\ -x_1 + x_2^2 + \alpha &= 0. \end{aligned}$$
- Show that for $\alpha = 1, 1/4$, and 0 there is no solution, one solution, and two solutions, respectively.
- 1.3** (a) Describe graphically in the (x, y) -plane nonlinear Gauss–Seidel applied to the system $f(x, y) = 0$, $g(x, y) = 0$. Consider all four combinations of orderings for the variables and the equations.
(b) Do the same thing for nonlinear Jacobi. Consider both orderings of the equations.
- 1.4** The system of equations

$$\begin{aligned} x &= 1 + h^2(e^{y\sqrt{x}} + 3x^2) \\ y &= 0.5 + h^2 \tan(e^x + y^2), \end{aligned}$$

can, for small values of h , be solved by fixed point iteration. Write a program which uses this method to solve the system for $h = 0, 0.01, \dots, 0.10$. For $h = 0$ take $x_0 = 1$, $y_0 = 0.5$, else use the solution for the previous value of h . The iterations should be broken off when the changes in x and y are less than $0.1h^4$.

- 1.5** For each of the roots of the system in Example 9.1.1,

$$\begin{aligned}x^2 - 2x - y + 1 &= 0 \\x^2 + y^2 - 1 &= 0\end{aligned}$$

determine whether or not the following iterations are locally convergent:

- (a) $x^{k+1} = (1 - y_k^2)^{1/2}$, $y_{k+1} = (x_k - 1)^2$.
- (b) $x_{k+1} = y_k^{1/2} + 1$, $y_{k+1} = (1 - x_k^2)$.

- 1.6** Suppose that $x \in \mathbf{R}^n$, where n is divisible by 3, and that the Jacobian is a square *tridiagonal* matrix.

- (a) Design an algorithm, where all the elements of the Jacobian are found by four evaluations of $g(x)$, when the uncentered difference approximation is used.
- (b) You may obtain the elements packed in three vectors. How do you unpack them into an $n \times n$ matrix? How many function evaluations do you need with the centered difference approximation?
- (c) Generalize to the case of an arbitrary *banded Jacobian*.

Comment: This idea was first published by Curtis, Powell, and Reid [128]

- 1.7** Apply two iterations of Newton's method to the equations of Problem 9.1.5, using the initial approximations $x_0 = 0.1$, and $y_0 = 1.1$.

- 1.8** If some of the equations in the system $f(x) = 0$ are linear, Newton's method will take this into account. Show that if (say) $f_i(x)$ is linear, then the Newton iterates x_k , $k \geq 1$, will satisfy $f_i(x_k) = 0$.

Figure 9.1.1. A rotating double pendulum.

- 1.9** A double pendulum rotates with angular velocity ω around a vertical axis (like a centrifugal regulator). At equilibrium the two pendulums make the angles x_1 and x_2 to the vertical axis, see Figure 9.1.1. It can be shown that the angles are determined by the equations

$$\begin{aligned}\tan x_1 - k(2 \sin x_1 + \sin x_2) &= 0, \\ \tan x_2 - 2k(\sin x_1 + \sin x_2) &= 0.\end{aligned}$$

where $k = l\omega^2/(2g)$.

(a) Solve by Newton's method the system for $k = 0.3$, with initial guesses $x_1 = 0.18$, $x_2 = 0.25$. How many iterations are needed to obtain four correct decimals?

(b) Determine the solutions with four correct decimals and plot the results for

$$k = 0.30, 0.31, \dots, 0.35, 0.4, 0.5, \dots, 0.9, 1, 2, 3, 4, 5, 10, 15, 20, \infty.$$

Use the result obtained for the previous k as initial guess for the new k . Record also how many iterations are needed for each value of k .

(c) Verify that the Jacobian is singular for $x_1 = x_2 = 0$, when $k = 1 - 1/\sqrt{2} \approx 0.2929$. A somewhat sloppy theory suggests that

$$x_1 \approx x_2 \approx \sqrt{k - (1 - 1/\sqrt{2})}, \quad 0 \leq k - (1 - 1/\sqrt{2}) \ll 1.$$

Do your results support this theory?

- 1.10** Describe how to apply the Newton idea to the solution of the steady state of a Matrix Riccati equation, i.e., to the solution of a matrix equation of the form,

$$A + BX + XC + XDX = 0,$$

where A, B, C, D are rectangular matrices of appropriate size. Assume that an algorithm for equations of the form $PX + XQ = R$ is given. Under what condition does such a linear equation have a unique solution? You don't need to discuss how to find the first approximation.

- 1.11** (a) Derive the formula for $\min B(\eta)$ and the optimal choice of η for the *uncentered* difference approximation to $g'(x)v$, also the simplified error estimate (for $\xi = \frac{1}{2}\|g\|/\|g'\|$).
(b) Work out the details of the study of the directional second derivative.
1.12 Investigate, for various functions f, g , the ratio of values of $B(\eta)$, obtained with the optimal η and with the value of η derived from the simplified estimate of ξ . Take, for example, $g(x) = e^{\alpha x}$, $g(x) = x^{-k}$.

9.2 Nonlinear Least Squares Problems

Consider the **nonlinear least squares problem**

$$\min_{x \in \mathbf{R}^n} \phi(x), \quad \phi(x) = \frac{1}{2}\|f(x)\|_2^2 = \frac{1}{2}f(x)^T f(x), \quad (9.2.1)$$

where $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$, $m \geq n$. Such problems are encountered in many applications such as operations research, control theory, chemical engineering, and all kinds of curve fitting and mathematical model fitting. Note that when $m = n$ and a consistent system this problem is equivalent to the nonlinear system $f(x) = 0$.

The nonlinear least squares problem is a special case of the nonlinear **optimization problem**

$$\min_x \phi(x), \quad \phi \in \mathbf{R}, \quad x \in \mathbf{R}^n, \quad (9.2.2)$$

where ϕ is the **objective function**. (Maximization problems can, of course, be transformed into a minimization problem by reversing the sign of ϕ .)

The problems (9.2.1) and (9.2.2) are said to be **unconstrained**. In practice the solution is often restricted to lie in some region $\mathcal{B} \subset \mathbf{R}^n$. This region is usually defined by inequality and equality constraints of the form

$$c_i(x) = 0, \quad i = 1 : m_1, \quad c_i(x) \geq 0, \quad i = m_1 + 1 : m. \quad (9.2.3)$$

In the simplest case the constraint functions $c_i(x)$ are linear. There may also be constraints in the form of lower and upper bounds $l_i \leq c_i(x) \leq u_i$. In this chapter we mainly consider only methods for unconstrained problems.

The important special case when both $\phi(x)$ and the constraints (9.2.3) are linear. This **linear programming** problem has been extensively studied. In Section 9.3 we survey some of the very efficient algorithms that have been developed for their solution.

9.2.1 Methods for Unconstrained Optimization

Consider an unconstrained optimization problem (9.2.2, where the objective function ϕ is a mapping $\mathbf{R}^n \rightarrow \mathbf{R}$. Often one would like to find a **global minimum**, i.e., a point where $\phi(x)$ assumes its least value in some subset $x \in \mathcal{B} \subset \mathbf{R}^n$. However, this is only possible in rather special cases and most numerical methods try to find local minima of $\phi(x)$.

Most numerical methods for unconstrained optimization try to find a **local minimum** of $\phi(x)$, i.e., a point x^* such that $\phi(x^*) \leq \phi(y)$ for all y in a neighborhood of x^* . If the objective function ϕ is continuously differentiable at a point x then any local minimum point x of ϕ must satisfy

$$g(x) = \nabla \phi(x) = 0, \quad (9.2.4)$$

where $g(x)$ is the gradient vector. This shows the close relationship between solving optimization problems and nonlinear systems of equations.

Definition 9.2.1.

A point x^ is said to be a **local minimum** of ϕ if $\phi(x^*) \leq \phi(y)$ for all y in a sufficiently small neighborhood of x^* . If $\phi(x^*) < \phi(y)$ then x^* is a **strong local minimum**.*

Assume that the objective function ϕ is continuously differentiable at a point x . The gradient vector $g(x) = \nabla \phi(x)$ is the normal to the tangent hyperplane of the multivariate function $\phi(x)$; see (9.2.4). A point x^* which satisfies $g(x) = \nabla \phi(x) = 0$ is called a **stationary point**. As in the scalar case, a *necessary* condition for x^* to be optimal is that it is a stationary point.

Note that information about the Hessian is needed to determine if a stationary point corresponds to a minimum of the objective function. We have the following fundamental result.

Theorem 9.2.2.

Necessary conditions for x^* to be a local minimum 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 $H(x^*)$ positive definite then x^* is a strong local minimum.

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

It is possible for a stationary point to be neither a maximum nor a minimum. Such a point is called a **saddle point** and is illustrated in two dimensions in Figure 9.3.1.

Many iterative methods for minimizing a function $\phi(x) : \mathbf{R}^n \rightarrow \mathbf{R}$ generate a sequence of points $\{x_k\}$, $k = 0, 1, 2, \dots$ from

$$x_{(k+1)} = x_k + \lambda_k d_k, \quad (9.2.5)$$

where d_k is a **search direction** and λ_k a **step length**. If we put

$$f(\lambda) = \phi(x_k + \lambda d_k), \quad (9.2.6)$$

then $f'(0) = (d_k)^T g(x_k)$, where $g(x_k)$ is the gradient of ϕ at x_k . The search direction d_k is said to be a **descent direction** if the directional derivative satisfies $(d_k)^T g(x_k) < 0$.

We assume in the following that d_k is normalized so that $\|d_k\|_2 = 1$. Then by the Schwarz inequality $f'(0)$ is minimized when

$$d_k = -g(x_k)/\|g(x_k)\|_2, \quad (9.2.7)$$

i.e., the negative gradient direction is a *direction of steepest descent*. With this choice the method (9.2.5) is the **steepest descent method** (Cauchy 1847). If combined with a suitable step length criteria this method will always converge to a stationary point.

In the steepest descent method the Hessian is not needed. Because of this the rate of convergence can, in general, only be *linear*. Convergence can be very slow and this method should usually be used only as a starting step, or when other search directions fail.

Example 9.2.1.

If the steepest descent method is applied to a quadratic function

$$\phi(x) = b^T x + \frac{1}{2} x^T G x,$$

where G is a symmetric positive definite matrix. Then it can be shown that (see Section 11.2.2)

$$\phi(x_{k+1}) - \phi(x^*) \approx \rho^2(\phi(x_k) - \phi(x^*)), \quad \rho = \frac{\kappa - 1}{\kappa + 1},$$

where $\kappa = \kappa_2(G)$ is the condition number of G . For example, if $\kappa = 1000$, then $\rho^2 = (999/1001)^2 \approx 0.996$, and about 575 iterations would be needed to gain one decimal digit of accuracy.

Faster convergence can be achieved by making use also of the second derivative Hessian matrix of the objective function $\phi(x)$. In Newton's method the new iterate x_{k+1} is determined by minimizing the **quadratic model**

$$Q_k(s_k) = \phi(x_k) + g(x_k)^T s_k + \frac{1}{2} s_k^T H(x_k) s_k, \quad (9.2.8)$$

of the function $\phi(x)$ at the current iterate x_k . When the Hessian matrix $H(x_k)$ is positive definite, Q_k has a unique minimizer s_k . We then set $x_{k+1} = x_k + s_k$, which is the **Newton step**. Clearly s_k is the solution of the symmetric linear system

$$H(x_k) s_k = -g(x_k). \quad (9.2.9)$$

A step along the Newton direction will not necessarily reduce ϕ even though it is the step that minimizes the model function. Therefore, Newton's method needs to be modified, unless the point x_K close to a minimizer. Either a line search can be included or a trust region technique used. In a line search the new iterate is taken to be

$$x_{k+1} = x_k + \lambda_k d_k,$$

where d_k is a search direction and $\lambda_k > 0$ chosen so that $\phi(x_{k+1}) < \phi(x_k)$. The algorithms described in Section 6.4 for minimizing the univariate function $\phi(x_k + \lambda d_k)$ can be used to determine λ_k . Usually it is not worthwhile to determine the minimizer accurately. It suffices to require that λ_k satisfies the following two conditions:

(a) The first requirement is that

$$|g(x_k + \lambda_k d_k)^T d_k| \leq \eta |g(x_k)^T d_k|, \quad (9.2.10)$$

where typically $\eta = 0.9$. This ensures that the magnitude of the directional derivative at $x_k + \lambda_k d_k$ is sufficiently reduced from that at x_k . Taking $\eta = 0$ corresponds to an exact line search.

(b) The second requirement is that the step produces a sufficient decrease in ϕ , or more precisely that

$$\phi(x_k + \lambda_k d_k) \leq \phi(x_k) + \mu \lambda_k g(x_k)^T d_k, \quad (9.2.11)$$

where μ is a constant satisfying $0 < \mu < \eta < 1$. Typically $\mu = 0.001$ is used.

It should be emphasized that quadratic convergence will only be achieved only if the step lengths converge sufficiently fast towards unity.

The Newton step is not a descent direction if $g_k^T H(x_k)^{-1} g_k \leq 0$. This situation is not likely to occur in the vicinity of a local optimum x^* , because of the positive (or at least nonnegative) definiteness of $H(x^*)$. Far away from an optimal point, however, this can happen. This is the reason for admitting the gradient as an alternative search direction—especially since there is a danger that the Newton direction will lead to a saddle point.

In the quadratic model the term $s_k^T H(x_k) s_k$ can be interpreted as the curvature of the surface $\phi(x)$ at x_k along s_k . Often $H(x_k)$ is expensive to compute, and we want to approximate this term. Expanding the gradient function in a Taylor series about x_k along a direction s_k we have

$$g(x_k + s_k) = g_k + H(x_k)s_k + \dots \quad (9.2.12)$$

Hence the curvature can be approximated from the gradient using a forward difference approximation

$$s_k^T G_k s_k \approx (g(x_k + s_k) - g(x_k))^T s_k.$$

In **quasi-Newton**, or variable metric methods an approximate Hessian is built up as the iterations proceed. Denote by B_k the approximate Hessian at the k th step. It is then required that B_{k+1} approximates the curvature of ϕ along $s_k = x_{k+1} - x_k$, i.e.,

$$B_{k+1}s_k = \gamma_k, \quad \gamma_k = g(x_{k+1}) - g(x_k), \quad (9.2.13)$$

where γ_k is the change in the gradient. The first equation in (9.2.13) is the analog of the secant equation (9.2.12) and is called the **quasi-Newton condition**.

Since the Hessian matrix is symmetric, it seems natural to require also that each approximate Hessian is symmetric. The quasi-Newton condition can be satisfied by making a simple update to B_k . The Powell–Symmetric–Broyden (PSB) update is

$$B_{k+1} = B_k + \frac{r_k s_k^T + s_k r_k^T}{s_k^T s_k} - \frac{(r_k^T s_k) s_k s_k^T}{(s_k^T s_k)^2}, \quad (9.2.14)$$

where $r_k = \gamma_k - B_k s_k$. The update matrix $B_{k+1} - B_k$ is here of rank two. It can be shown that it is the unique symmetric matrix which minimizes $\|B_{k+1} - B_k\|_F$, subject to (9.2.13).

When line searches are used, practical experience has shown the Broyden–Fletcher–Goldfarb–Shanno (BFGS) update, given by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\gamma_k \gamma_k^T}{\gamma_k^T s_k},$$

to be the best update.

If the choice of step length parameter λ_k is such that $\gamma_k^T s_k > 0$, then B_{k+1} will inherit positive definiteness from B_k . Therefore it is usual to combine the BFGS formula with $B_0 = I$. The most widely used algorithms for unconstrained

optimization use these techniques, when it is reasonable to store B_k as a dense matrix. Note that since the search direction will be computed from

$$B_k d_k = -g_k, \quad k \geq 1, \quad (9.2.15)$$

this means that the first iteration of a quasi-Newton method is a steepest descent step.

If B_k is positive definite then the local quadratic model has a unique local minimum, and the search direction d_k computed from (9.2.15) is a descent direction. Therefore it is usually required that the update formula generates a positive definite approximation B_{k+1} when B_k is positive definite.

To compute a new search direction we must solve the linear system (9.2.15). In general this would require $O(n^3)$ operations. Since the approximate Hessian B_k is a rank two modification of B_{k-1} , it is possible to solve this system more efficiently. One possibility would be to maintain an approximation to the *inverse* Hessian, using the Sherman–Morrison formula (7.1.26). Then only $O(n^2)$ operations would be needed. If the Cholesky factorization $B_k = L_k D_k L_k^T$ is available, then the system (9.2.15) can be solved in order n^2 operations. Furthermore, the factors L_{k+1} and D_{k+1} of the updated Hessian approximation B_{k+1} can be computed in about the same number of operations that would be needed to generate B_{k+1}^{-1} . An important advantage of using the Cholesky factorization is that the positive definiteness of the approximate Hessian cannot be lost through round-off errors.

An algorithm for modifying the Cholesky factors of a symmetric positive definite matrix B is given in [251]. Let $B = LDL^T$ be the Cholesky factorization of B , where $L = (l_{ij})$ is unit lower triangular and $D = \text{diag}(d_j) > 0$ diagonal. Let $\bar{B} = B \pm vv^T$ be a rank-one modification of B . Then we can write

$$\bar{B} = LDL^T \pm vv^T = L(D \pm pp^T)L^T,$$

where p is the solution of the triangular system $Lp = v$. The Cholesky factorization $D \pm pp^T = \hat{L}\hat{D}\hat{L}^T$ can be computed by a simple recursion, and then we have $\bar{L} = L\hat{L}$. In case of a positive correction $B = B + vv^T$, the vector p and the elements of \bar{L} and \bar{D} can be computed in a numerical stable way using only $3n^2/2$ flops.

9.2.2 Nonlinear Least Squares Problems

We emphasize in the following mainly those aspects of the problem (9.2.1), which derive from the special form of $\phi(x)$.

An important source of nonlinear least squares problems is fitting data to a mathematical model. Given data (y_i, t_i) , $i = 1 : m$, one want to fit a model function $y = h(x, t)$. If we let $r_i(x)$ represent the error in the model prediction for the i :th observation,

$$r_i(x) = y_i - h(x, t_i), \quad i = 1 : m,$$

we want to minimize some norm of the vector $r(x)$. The choice of the least squares measure is justified here, as for the linear case, by statistical considerations. If the observations have equal weight, this leads to the minimization problem in (9.2.1) with $f(x) = r(x)$.

The standard methods for the nonlinear least squares problem require derivative information about the component functions of $f(x)$. We assume here that $f(x)$ is twice continuously differentiable. It is easily shown that the gradient of $\phi(x) = \frac{1}{2}f^T(x)f(x)$ is

$$g(x) = \nabla\phi(x) = J(x)^T f(x), \quad (9.2.16)$$

where

$$J(x)_{ij} = \frac{\partial f_i(x)}{\partial x_j} \in \mathbf{R}^{m \times n}, \quad i = 1 : m, \quad j = 1 : n.$$

is the Jacobian matrix of $f(x)$. The Hessian matrix is

$$H(x) = \nabla^2\phi(x) = J(x)^T J(x) + Q(x), \quad Q(x) = \sum_{i=1}^m f_i(x)G_i(x), \quad (9.2.17)$$

where $G_i(x) \in \mathbf{R}^{n \times n}$, is the Hessian matrix of $f_i(x)$ with elements

$$G_i(x)_{jk} = \frac{\partial^2 f_i(x)}{\partial x_j \partial x_k}, \quad i = 1 : m, \quad j, k = 1 : n. \quad (9.2.18)$$

The special forms of the gradient $g(x)$ and Hessian $H(x)$ can be exploited by methods for the nonlinear least squares problem.

A necessary condition for x^* to be a local minimum of $\phi(x)$ is that x^* is a stationary point, i.e., satisfies

$$g(x^*) = J(x^*)^T f(x^*) = 0. \quad (9.2.19)$$

A necessary condition for a stationary point x^* to be a local *minimum* of $\phi(x)$ is that the Hessian matrix $H(x)$ is positive definite at x^* .

There are basically two different ways to view problem (9.2.1). One could think of this problem as arising from an overdetermined system of nonlinear equations $f(x) = 0$. It is then natural to approximate $f(x)$ by a linear model around a given point x_k

$$\tilde{f}(x) = f(x_k) + J(x_k)(x - x_k), \quad (9.2.20)$$

and use the solution p_k to the linear least squares problem

$$\min_p \|f(x_k) + J(x_k)p\|_2. \quad (9.2.21)$$

to derive an new (hopefully improved) improved approximate solution $x_{k+1} = x_k + p_k$. This approach, which only uses first order derivative information about $f(x)$, leads to a class of methods called **Gauss–Newton** type methods. This method, which in general only have linear rate of convergence, will be discussed in Section 9.2.3.

In the second approach (9.2.1) is viewed as a special case of unconstrained optimization in \mathbf{R}^n . A quadratic model at a point x_k is used,

$$\tilde{\phi}_c(x) = \phi(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k), \quad (9.2.22)$$

where the gradient and Hessian of $\phi(x) = \frac{1}{2}f^T(x)f(x)$ are given by (9.2.16) and (9.2.17). The minimizer of $\tilde{\phi}_c(x)$ is given by $x_{k+1} = x_k + p_k$, where

$$p_k = -H(x_k)^{-1}J(x_k)^T f(x_k). \quad (9.2.23)$$

This method is equivalent to Newton's method applied to (9.2.1), which usually is locally quadratically convergent.

The Gauss–Newton method can be thought of as arising from neglecting the second derivative term

$$Q(x) = \sum_{i=1}^m f_i(x)G_i(x),$$

in the Hessian $H(x_k)$. Note that $Q(x_k)$ will be small close to the solution x^* if either the residual norm $\|f(x^*)\|$ is small or if $f(x)$ 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 $f(x^*) = 0$ the local convergence will be the same for both methods. However, 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.

The cost of computing and storing the mn^2 second derivatives (9.2.18) in $Q(x)$ can be prohibitively high. However, for curve fitting problems the function values $f_i(x) = y_i - h(x, t_i)$ and the derivatives $\partial^2 f_i(x)/\partial x_j \partial x_k$, 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 sometimes be computed cheaply. Another case when it may be feasible to store approximations to all $G_i(x)$, $i = 1 : m$, is when every function $f_i(x)$ only depends on a small subset of the n variables. Then both the Jacobian $J(x)$ and the Hessian matrices $G_i(x)$ will be *sparse* and special methods, such as those discussed in Section 7.7 may be applied.

9.2.3 Gauss–Newton and Newton Methods

The Gauss–Newton method for problem (9.2.1) is based on a sequence of linear approximations of $f(x)$ of the form (9.2.20). (As the name implies the Gauss–Newton method was used already by Gauss.) If x_k denotes the current approximation then the Gauss–Newton step d_k is a solution to the linear least squares problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \quad d_k \in \mathbf{R}^n. \quad (9.2.24)$$

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 $d_k = -J(x_k)^\dagger f(x_k)$ 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 $J(x_k)^T f(x_k) \neq 0$ then d_k is a descent direction for $\phi(x) = \frac{1}{2} f^T(x) f(x)$,

The first property is easily verified. To prove the second property we note that

$$d_k^T g(x_k) = -f(x_k)^T J^\dagger(x_k)^T J(x_k)^T f(x_k) = -\|P_{J_k} f(x_k)\|_2^2, \quad (9.2.25)$$

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 $J(x_k)^T f(x_k) \neq 0$ then $f(x_k)$ is not in the nullspace of $J(x_k)^T$ and it follows that $P_{J_k} f(x_k) \neq 0$. This proves (ii).

The Gauss–Newton method can fail at an intermediate point where the Jacobian is rank deficient or illconditioned. Formally we can take d_k to be the minimum norm solution

$$d_k = -J(x_k)^\dagger f(x_k).$$

In practice it is necessary to include some strategy to estimate the numerical rank of $J(x_k)$, cf. Section 7.1.6 and 8.1.5. That the assigned rank can have a decisive influence is illustrated by the following example.

Example 9.2.2. (Gill, Murray and Wright [254, p. 136])

Let $J = J(x_k)$ and $f(x_k)$ be defined by

$$J = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix},$$

where $\epsilon \ll 1$ and f_1 and f_2 are of order unity. If J is considered to be of rank two then the search direction $d_k = s_1$, whereas if the assigned rank is one $d_k = s_2$, where

$$s_1 = -\begin{pmatrix} f_1 \\ f_2/\epsilon \end{pmatrix}, \quad s_2 = -\begin{pmatrix} f_1 \\ 0 \end{pmatrix}.$$

Clearly the two directions s_1 and s_2 are almost orthogonal and s_1 is almost orthogonal to the gradient vector $J^T f$.

Usually it is preferable to *underestimate* the rank except when $\phi(x)$ is actually close to an ill-conditioned quadratic function. One could also switch to a search direction along the negative gradient $-g_k = -J(x_k)^T f(x_k)$, or use a linear combination

$$d_k = \mu_k g_k, \quad \mu_k = \|g_k\|_2^2 / \|J(x_k) g_k\|_2^2.$$

as in Powell's method.

The Gauss–Newton method as described above has several advantages. It solves linear problems in just one iteration and has fast convergence on small residual and mildly nonlinear problems. However, it may not be locally convergent on problems that are very nonlinear or have large residuals.

To analyze the rate of convergence of Gauss–Newton type methods let $J^\dagger(x)$ denote the pseudoinverse of $J(x)$, and assume that $\text{rank}(J(x)) = n$. Then $I = J^\dagger(x) J(x)$, and (9.2.17) can be written in the form

$$H(x) = J(x)^T (I - \gamma K(x)) J(x), \quad K(x) = J^\dagger(x)^T G_w(x) J^\dagger(x). \quad (9.2.26)$$

where $\gamma = \|f(x)\|_2 \neq 0$, and

$$G_w(x) = \sum_{i=1}^m w_i G_i(x), \quad w(x) = -\frac{1}{\gamma} f(x). \quad (9.2.27)$$

The matrix $K(x)$ is symmetric, and has a geometric interpretation. It is called the **normal curvature matrix** of the n -dimensional surface $z = f(x)$ in \mathbf{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 eigenvalues of $K(x)$, are called the **principal radii of curvature** of the surface.

The Hessian matrix $H(x^*)$ is positive definite and x^* a local minimum if and only if $u^T H(x^*) u > 0$, for all $u \in \mathbf{R}^n \neq 0$. If $\text{rank}(J(x^*)) = n$, 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

$$1 - \gamma \kappa_1 > 0. \quad (9.2.28)$$

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^* .

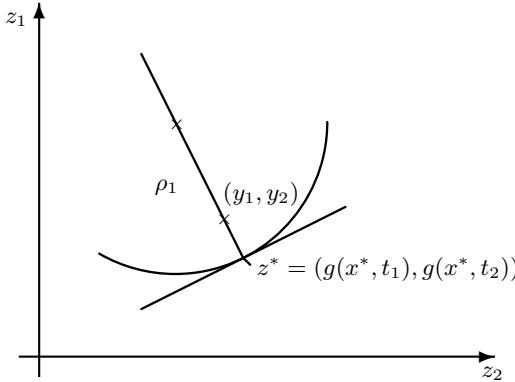


Figure 9.2.1. Geometry of the data fitting problem for $m = 2, n = 1$.

Example 9.2.3.

The geometrical interpretation of the nonlinear least squares problem (9.2.1) is to find a point on the surface $\{f(x) \mid x \in \mathbf{R}^n\}$ in \mathbf{R}^m closest to the origin. In case of data fitting $f_i(x) = y_i - h(x, t_i)$, and it is more illustrative to consider the surface

$$z(x) = (h(x, t_1), \dots, h(x, t_m))^T \in \mathbf{R}^m.$$

The problem is then to find the point $z(x^*)$ on this surface closest to the observation vector $y \in \mathbf{R}^m$. This is illustrated in Figure 9.2.1 for the simple case of $m = 2$

observations and a scalar parameter x . Since in the figure we have $\gamma = \|y - z(x^*)\|_2 < \rho$, it follows that $1 - \gamma\kappa_1 > 0$, which is consistent with the fact that x^* is a local minimum. In general the solution (if it exists) is given by an orthogonal projection of y onto the surface $z(x)$. Compare the geometrical interpretation in Figure 8.1.1 for the linear case $z(x) = Ax$.

It can be shown that the asymptotic rate of convergence of the Gauss–Newton method in the neighborhood of a critical point x^* is equal to

$$\rho = \gamma \max(\kappa_1, -\kappa_n),$$

where κ_i are the eigenvalues of the of the normal curvature matrix $K(x)$ in (9.2.26) evaluated at x^* and $\gamma = \|f(x^*)\|_2 = 0$. In general convergence is linear, but if $\gamma = 0$ then convergence becomes superlinear. Hence the asymptotic rate of convergence of the undamped Gauss–Newton method is fast when either

- (i) the residual norm $\gamma = \|r(x^*)\|_2$ is small, or
- (ii) $f(x)$ is mildly nonlinear, i.e. $|\kappa_i|, i = 1 : n$ are small.

If x^* is a saddle point then $\gamma\kappa_1 \geq 1$, i.e., using undamped Gauss–Newton one is repelled from a saddle point. This is an excellent property since saddle points are not at all uncommon for nonlinear least squares problems.

The Gauss–Newton method can be modified for global convergence. in a similar way as described in Section 9.1.3 Newton’s method. If the Gauss–Newton direction d_k is used as a search direction we consider the one-dimensional minimization problem

$$\min_{\lambda} \|f(x_k + \lambda d_k)\|_2^2.$$

As remarked above it is in general not worthwhile to solve this minimization accurately. Instead we can take λ_k to be the largest number in the sequence $1, \frac{1}{2}, \frac{1}{4}, \dots$ for which

$$\|f(x_k)\|_2^2 - \|f(x_k + \lambda_k d_k)\|_2^2 \geq \frac{1}{2} \lambda_k \|P_{J_k} f(x_k)\|_2^2.$$

Here $\lambda = 1$ corresponds to the full Gauss–Newton step. Since d_k is a descent direction, this damped Gauss–Newton method is locally convergent on almost all nonlinear least squares problems. In fact is 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 minimum. This is in contrast to the undamped Gauss–Newton method, which may fail to converge to a local minimum.

The rate of convergence for the undamped Gauss–Newton method can be estimated during the iterations from

$$\rho_{\text{est}} = \|P_J(x_{k+1})r_{k+1}\|_2/\|P_J(x_k)r_k\|_2 = \rho + O(\|x_k - x^*\|_2^2). \quad (9.2.29)$$

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) then one should consider switching to a method using second derivative information, or perhaps evaluate the quality of the underlying model.

Even the damped Gauss–Newton method can have difficulties to get around an intermediate point where the Jacobian matrix rank deficient. This can be avoided either by taking second derivatives into account (see Section 9.2.3) or by further stabilizing the damped Gauss–Newton method to overcome this possibility of failure. Methods using the latter approach were first suggested by Levenberg [426] and Marquardt [405]. Here a search direction d_k is computed by solving the problem

$$\min_{d_k} \{\|f(x_k) + J(x_k)d_k\|_2^2 + \mu_k \|d_k\|_2^2\}, \quad (9.2.30)$$

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

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \quad \text{subject to } \|d_k\|_2 \leq \delta_k, \quad (9.2.31)$$

where $\mu_k = 0$ if the constraint in (9.2.31) 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 region of trust for the linear model $f(x) \approx f(x_k) + J(x_k)(x - x_k)$.

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 $f(x_k)$, $J(x_k)$, and determine d_k as a solution to the subproblem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \quad \text{subject to } \|D_k d_k\|_2 \leq \delta_k,$$

where D_k is a diagonal scaling matrix.

- (b) Compute the ratio $\rho_k = (\|f(x_k)\|_2^2 - \|f(x_k + d_k)\|_2^2)/\psi_k(d_k)$, where

$$\psi_k(d_k) = \|f(x_k)\|_2^2 - \|f(x_k) + J(x_k)d_k\|_2^2$$

is the model prediction of the decrease in $\|f(x_k)\|_2^2$.

- (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 $f(x)$ is continuously differentiable, $f'(x)$ 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 very nonlinear problems. Methods using second derivative information , see Section 9.2.3 are somewhat more efficient but also more complex than the Levenberg–Marquardt methods.

The analysis in the Section 9.2.3 showed that for large residual problems and strongly nonlinear problems, methods of Gauss–Newton type may converge slowly. Also, these methods can have problems at points where the Jacobian is rank deficient. When second derivatives of $f(x)$ are available Newton’s method, which uses the quadratic model (9.2.22), can be used to overcome these problems. The optimal point d_k of this quadratic model, satisfies the linear system

$$H(x_k)d_k = -J(x_k)^T f(x_k), \quad (9.2.32)$$

where $H(x_k)$ is the Hessian matrix at x_k , and $x_k + d_k$ is chosen as the next approximation.

It can be shown, see Dennis and Schnabel [156, p. 229], that Newton’s method is quadratically convergent to a local minimum x^* as long as $H(x)$ is Lipschitz continuous around x_k and $H(x^*)$ is positive definite. To get global convergence a line search algorithm is used, where the search direction d_k is taken as the Newton direction. Note that the Hessian matrix $H(x_k)$ must be positive definite in order for the Newton direction d_k to be a descent direction.

Newton’s method is not often used since the second derivative term $Q(x_k)$ in the Hessian is rarely available at a reasonable cost. However, a number of methods have been suggested that partly takes 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 successively builds up approximations B_k to the Hessian matrices $H(x_k)$. The search directions are computed from

$$B_k d_k = -J(x_k)^T f(x_k),$$

where B_k satisfies the quasi-Newton conditions

$$B_k s_k = y_k, \quad s_k = x_k - x_{k-1}, \quad y_k = g(x_k) - g(x_{k-1}), \quad (9.2.33)$$

where $g(x_k) = J(x_k)^T f(x_k)$. As starting value $B_0 = J(x_0)^T J(x_0)$ is recommended.

The direct application of quasi-Newton methods to the nonlinear least squares problem outlined above has not been so efficient in practice. One reason is that these methods disregard the information in $J(x_k)$, and often $J(x_k)^T J(x_k)$ is the dominant part of $H(x_k)$. A more successful approach is to approximate $H(x_k)$ by $J(x_k)^T J(x_k) + S_k$, where S_k is a quasi-Newton approximation of the term $Q(x_k)$. Initially one takes $S_0 = 0$. The quasi-Newton relations (9.2.33) can now be written

$$S_k s_k = z_k, \quad z_k = (J(x_k) - J(x_{k-1}))^T f(x_k), \quad (9.2.34)$$

where S_k is required to be symmetric. It can be shown that a solution to (9.2.34) which minimizes the change from S_{k-1} in a certain weighted Frobenius norm is

given by the update formula

$$B_k = B_{k-1} + \frac{w_k y_k^T + y_k w_k^T}{y_k^T s_k} - \frac{w_k^T s_k y_k y_k^T}{y_k^T s_k^2}, \quad (9.2.35)$$

where $s_k = x_k - x_{k-1}$, and $w_k = z_k - B_{k-1}s_k$.

In some cases the updating (9.2.35) gives inadequate results. This motivates the inclusion of "sizing" in which the matrix B_k is replaced by $\tau_k B_k$, where

$$\tau_k = \min\{|s_k^T z_k|/|s_k^T B_k s_k|, 1\}.$$

This heuristic choice ensures that S_k converges to zero for zero residual problems, which improves the convergence behavior.

In another approach, due to Gill and Murray [252], $J(x_k)^T J(x_k)$ is regarded as a good estimate of the Hessian in the right invariant subspace corresponding to the large singular values of $J(x_k)$. In the complementary subspace the second derivative term $Q(x_k)$ is taken into account. Let the SVD of $J(x_k)$ be

$$J(x_k) = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n),$$

where the singular values are ordered so that $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$. Then putting $Q_k = Q(x_k)$ the equations for the Newton direction $d_k = Vq$ can be written

$$(\Sigma^2 + V^T Q_k V)q = -\Sigma r_1, \quad r_1 = (I_n \ 0) U^T f(x_k). \quad (9.2.36)$$

We now split the singular values into two groups, $\Sigma = \text{diag}(\Sigma_1, \Sigma_2)$, where $\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_r)$ are the "large" singular values. If we partition V, q and r conformally, then the first r equations in (9.2.36) can be written.

$$(\Sigma_1^2 + V_1^T Q_k V_2)q_1 + V_1^T Q_k V_2 q_2 = -\Sigma_1 \bar{r}_1.$$

If the terms involving Q_k are neglected compared to $\Sigma_1^2 q_1$, we get $q_1 = -\Sigma_1^{-1} \bar{r}_1$. If this is substituted into the last $(n - r)$ equations we can solve for q_2 from

$$(\Sigma_2^2 + V_2^T Q_k V_2)q_2 = -\Sigma_2 \bar{r}_2 - V_2^T Q_k V_1 q_1.$$

The approximate Newton direction is then given by $d_k = Vq = V_1 q_1 + V_2 q_2$. The splitting of the singular values is updated at each iteration, the idea being to maintain r close to n as long as adequate progress is made.

There are several alternative ways to implement the method by Gill and Murray. If Q_k is not available explicitly, then a finite difference approximation to $V_2^T Q_k V_2$ can be obtained as follows. Let v_j be a column of V_2 and h a small positive scalar. Then

$$(\nabla r_i(x_k + hv_j) - \nabla r_i(x_k))/h = v_j^T G_i(x_k) + O(h),$$

where the vector on the left hand side is the i th row of $(J(x_k + hv_j) - J(x_k))/h$. Multiplying with $r_i(x_k)$ and adding we obtain

$$\begin{aligned} r(x_k)^T (J(x_k + hv_j) - J(x_k))/h &= v_j^T \sum_{i=1}^m r_i(x_k) G_i(x_k) + O(h) \\ &= v_j^T Q_k + O(h). \end{aligned}$$

Repeating this for all columns in V_2 we obtain an approximation for $V_2^T Q_k$ and we finally form $(V_2^T Q_k) V_2$.

9.2.4 Separable 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 minimize the nonlinear functional

$$\|r(y, z)\|^2, \quad r_i(y, z) = s_i - \sum_{j=1}^p y_j \phi_j(z; t_i), \quad i = 1 : m. \quad (9.2.37)$$

where (s_i, t_i) , $i = 1 : m$, is the data to be fitted. Here $y \in \mathbf{R}^p$ are linear parameters and $z \in \mathbf{R}^q$ are nonlinear parameters. Simple examples of nonlinear functions $\phi_j(z; t_i)$ are exponential or rational functions $\phi_j = e^{z_j t}$, $\phi_j = 1/(t - z_j)$. The full least squares problem can be written in the form

$$\min_{y, z} \|g(z) - \Phi(z)y\|^2, \quad (9.2.38)$$

where $\Phi(z)$ is a matrix whose j th column equals $\phi_j(z; t_i)$. For any fixed values of z the problem of finding the corresponding optimal values of y is a *linear* least squares problem that can be easily solved. A particularly simple case is when $r(y, z)$ is linear in both y and z so that we also have

$$r(y, z) = h(y) - \Psi(y)z, \quad \Psi(y) \in \mathbf{R}^{m \times q}.$$

An important particular example is the exponential fitting problem

$$\min_{a, z} \sum_{i=1}^m (q(t_i) - y_i)^2, \quad q(t) = \sum_{k=1}^n a_k e^{z_k t}. \quad (9.2.39)$$

This arises in many applications, e.g., where we have reactions with different time constant. The model is nonlinear only in the parameters z_k , $k = 1 : n$. Given values of z_k the linear subproblem (9.2.36) for a_k , $k = 1 : n$ is easily solved. Problems of this form are often ill-conditioned because the same data can be well approximated by different exponential sums.

A standard method for solving the problem (9.2.39), when data (t_j, y_j) , $i = j : m$, is given in equidistant points $t_j = t_1 + jh$ is **Prony's method**. Introducing the new variables $v_k = e^{hz_k}$ and setting $c_k = a_k e^{z_k t_1}$ fitting the data to the exponential leads the overdetermined linear system $Mc = y$, where

$$M = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_1 & v_2 & \cdots & v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_1^m & v_2^m & \cdots & v_n^m \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}. \quad (9.2.40)$$

For given v_k , the minimizing problem (9.2.39) is equivalent to the linear least squares problem of minimizing $\|Mc - y\|_2^2$. Now assume that the unknown v_1, \dots, v_n are roots to the polynomial

$$\phi(v) = (v - v_1)(v - v_2) \cdots (v - v_n) = v^n + s_1 v^{n-1} + \cdots + s_m.$$

Multiplying the equations in (9.2.40) in turn by $s_n, s_{n-1}, \dots, s_1, s_0 = 1$, and adding, we obtain

$$\sum_{k=1}^n \phi(v_k) c_k = \sum_{k=0}^n s_{n-k} y_k = 0$$

since $\phi(v_k) = 0$, $k = 1 : n$. By shifting the origin with h we get a new equation. Repeating this we get a (usually overdetermined) system. Thus we have $m - n + 1$ equations for determining the unknowns s_n, s_{n-1}, \dots, s_1 . This can be solved by the method of least squares. Determining the roots of the polynomial $\phi(v)$ we obtain v_k and $\lambda_k = \ln v_k/h$. Finally we get c_k from the linear system (9.2.40) and $a_k = c_k e^{-\lambda_k x_1}$.

We now consider a method for solving general separable least squares problems. We first observe that the solution of (9.2.38) can be expressed as

$$y(z) = \Phi^\dagger(z)g(z), \quad (9.2.41)$$

where $\Phi^\dagger(z)$ is the pseudoinverse of $\Phi(z)$. If the linear parameters are eliminated in (9.2.38) the original minimization problem can be cast in the form

$$\min_z \| (I - P_{\Phi(z)})g \|^2, \quad P_{\Phi(z)} = \Phi(z)\Phi(z)^\dagger, \quad (9.2.42)$$

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** consists of solving (9.2.42), by a Gauss–Newton–Marquardt 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 [271], holds for any symmetric generalized inverse Φ^- . Its proof is a good exercise in differentiating matrix expressions.

Lemma 9.2.3.

Let $\Phi = \Phi(\zeta) \in \mathbf{R}^{m \times n}$ be a matrix of local constant rank and Φ^\dagger its pseudoinverse. Then

$$\frac{d}{d\zeta} (\Phi\Phi^\dagger) = P_{\mathcal{N}(\Phi^T)} \frac{d\Phi}{d\zeta} \Phi^\dagger + (\Phi^\dagger)^T \frac{d\Phi^T}{d\zeta} P_{\mathcal{N}(\Phi^T)}, \quad (9.2.43)$$

where $P_{\mathcal{R}(\Phi)} = I - \Phi\Phi^\dagger$.

Proof. Since $P_{\mathcal{R}(\Phi)}\Phi = \Phi$,

$$\frac{d}{d\zeta} (P_{\mathcal{R}(\Phi)}\Phi) = \frac{d}{d\zeta} (P_{\mathcal{R}(\Phi)})\Phi + P_{\mathcal{R}(\Phi)} \frac{d\Phi}{d\zeta} = \frac{d\Phi}{d\zeta},$$

and hence

$$\frac{d}{d\zeta}(P_{\mathcal{R}(\Phi)})\Phi = \frac{d\Phi}{d\zeta} - P_{\mathcal{R}(\Phi)}\frac{d\Phi}{d\zeta} = (P_{\mathcal{N}(\Phi^T)})\frac{d\Phi}{d\zeta}.$$

Thus, since $P_{\mathcal{R}(\Phi)} = \Phi\Phi^\dagger$,

$$\frac{d}{d\zeta}(P_{\mathcal{R}(\Phi)})P_{\mathcal{R}(\Phi)} = (P_{\mathcal{N}(\Phi^T)})\frac{d\Phi}{d\zeta}\Phi^\dagger. \quad (9.2.44)$$

Since an orthogonal projector is symmetric we have

$$\left(\frac{d}{d\zeta}(P_{\mathcal{R}(\Phi)})P_{\mathcal{R}(\Phi)} \right)^T = P_{\mathcal{R}(\Phi)}\frac{d}{d\zeta}(P_{\mathcal{R}(\Phi)}) \quad (9.2.45)$$

we finally obtain from (9.2.44) and (9.2.45)

$$\begin{aligned} \frac{d}{d\zeta}(P_{\mathcal{R}(\Phi)}) &= \frac{d}{d\zeta}(P_{\mathcal{R}(\Phi)}^2) = \frac{d}{d\zeta}(P_{\mathcal{R}(\Phi)})P_{\mathcal{R}(\Phi)} + P_{\mathcal{R}(\Phi)}\frac{d}{d\zeta}(P_{\mathcal{R}(\Phi)}) \\ &= (P_{\mathcal{N}(\Phi^T)})\frac{d\Phi}{d\zeta}\Phi^\dagger + (\Phi^\dagger)^T\frac{d\Phi^T}{d\zeta}P_{\mathcal{N}(\Phi^T)}, \end{aligned}$$

which completes the proof. \square

We here describe a variable projection algorithm due to Kaufman [372], which uses a Gauss–Newton method applied to the problem (9.2.42). The j th column of the Jacobian of the reduced problem can be written

$$J = - \left[P_{\mathcal{N}(\Phi^T)} \frac{d\Phi}{dz_j} \Phi^\dagger + (\Phi^\dagger)^T \frac{d\Phi^T}{dz_j} P_{\mathcal{N}(\Phi^T)} \right] y.$$

Kaufman's simplification consists of 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 approximation. The next approximation is determined as follows:

- (i) Compute the solution δy_k to the linear subproblem

$$\min_{\delta y_k} \|f(z_k)\delta y_k - (g(z_k) - f(z_k)y_k)\|_2, \quad (9.2.46)$$

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 $x_{k+1/2}$, i.e., d_k is the solution to

$$\min_{d_k} \|C(x_{k+1/2})d_k + r(y_{k+1/2}, z_k)\|_2, \quad (9.2.47)$$

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 (9.2.47) we have used that by (9.2.38) 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 given by

$$r_z(y_{k+1/2}, z_k) = B(z_k)y_{k+1/2} - g'(z_k), \quad B(z)y = \left(\frac{\partial F}{\partial z_1}y, \dots, \frac{\partial F}{\partial z_q}y \right),$$

where $B(z)y \in \mathbf{R}^{m \times q}$. Note that in case $r(y, z)$ is linear also in y it follows from (9.2.18) 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 Section 9.2.3 and 9.2.3.

The variable projection approach not only reduces the dimension of the parameter space but also leads to a better conditioned problem. One advantage of the Kaufman 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 (9.2.46). This seems to make a difference in the first steps of the iterations, and the variable projection algorithm can solve problems for which methods not using separability fail.

It can be shown that the Gauss–Newton algorithm applied to (9.2.42) has the same asymptotic convergence rate as the ordinary Gauss–Newton’s method. In particular, both converge quadratically for zero residual problem. This is in contrast to the naive algorithm for separable problems of alternatively minimizing $\|r(y, z)\|_2$ over y and z , which *always* converges linearly.

A carefully written and documented program VARPO implements the variable projection algorithm, including the modification of Kaufman and calculation of the covariance matrix. A version called VARP2 by LeVeque handles multiple right hand sides. Both VARPO and VARP2 are available in the public domain in the Port Library, by David Gay⁴⁰ see also A. J. Miller⁴¹

9.2.5 Nonlinear Orthogonal Regression

In Section 8.6.5 we considered the linear orthogonal regression problem of fitting a hyperplane M to a set of given points in $P_i \in \mathbf{R}^n$, $i = 1 : m$. The solution to this problem could be obtained from the SVD of a related matrix. In this section we consider the orthogonal regression where the mathematical model to be fitted is nonlinear.

Suppose we have observations (y_i, t_i) , $i = 1 : m$, that are to be fitted to the model

$$y = f(p, t). \quad (9.2.48)$$

where f is a scalar nonlinear function, t a scalar variable and $p \in \mathbf{R}^n$ parameters to be determined. In the classical regression model the values t_i of the independent variable are assumed to be exact and only y_i are subject to random errors. In this case it is natural to minimize the sum of squares of the deviations $y_i - f(p, t_i)$. In the more general situation, when also the values t_i contain errors may be better to

⁴⁰<http://netlib.bell.labs.com/netlib/master/readme.html>;

⁴¹<http://users.igp.ornl.gov/amiller/>.

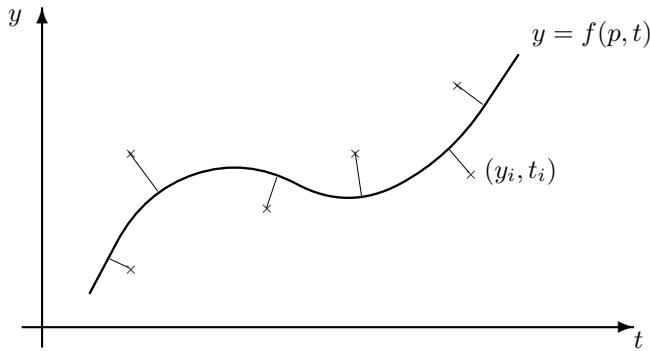


Figure 9.2.2. *Orthogonal distance regression.*

minimize the *sum of squares of the orthogonal distance* from the observations (y_i, t_i) to the curve; see Figure 9.2.2. 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 (9.2.48) is minimized, cf. Figure 9.2.2. Hence the parameters p should be chosen as the solution to

$$\min_{p, \epsilon, \delta} \sum_{i=1}^m (\epsilon_i^2 + \delta_i^2), \quad \text{subject to } y_i + \epsilon_i = f(p, t_i + \delta_i), \quad i = 1 : m.$$

Eliminating ϵ_i using the constraints we arrive at the **orthogonal distance problem**

$$\min_{p, \delta} \sum_{i=1}^m (f(p, t_i + \delta_i) - y_i)^2 + \delta_i^2. \quad (9.2.49)$$

Note that (9.2.49) is a nonlinear least squares problem even if $f(p, t)$ is linear in p .

The problem (9.2.49) 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

$$r_1^T(\delta, p)_i = f(p, t_i + \delta_i) - y_i, \quad r_2^T(\delta) = \delta_i, \quad i = 1 : m,$$

the Jacobian matrix for problem (9.2.49) can be written in block form as

$$\tilde{J} = \begin{pmatrix} D_1 & J \\ \underbrace{I_m}_m & \underbrace{0}_n \end{pmatrix} \}_{m+n}^{2m} \in \mathbf{R}^{2m \times (m+n)}, \quad (9.2.50)$$

where

$$D_1 = \text{diag}(d_1, \dots, d_m), \quad d_i = \left(\frac{\partial f}{\partial t} \right)_{t=t_i+\delta_i},$$

$$J_{ij} = \left(\frac{\partial f}{\partial p_j} \right)_{t=t_i+\delta_i}, \quad i = 1 : m, \quad j = 1 : n.$$

Note that \tilde{J} is sparse and highly structured. In the Gauss–Newton method we compute corrections $\Delta\delta_k$ and Δp_k to the current approximations which solve the linear least squares problem

$$\min_{\Delta\delta, \Delta p} \left\| \tilde{J} \begin{pmatrix} \Delta\delta \\ \Delta p \end{pmatrix} - \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \right\|_2, \quad (9.2.51)$$

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 (9.2.51) now decouples, and Δp_k is determined as the solution to

$$\min_{\Delta p} \|L\Delta p - s_2\|_2.$$

Here $L \in \mathbf{R}^{m \times n}$, so this is a problem of the same size as that which defines the Gauss–Newton correction in the classical nonlinear least squares problem. We then have

$$\Delta\delta_k = D_2^{-1}(s_2 - K\Delta p_k).$$

So far we have assumed that y and t are scalar variables. More generally, if $y \in \mathbf{R}^{n_y}$ and $t \in \mathbf{R}^{n_t}$ the problem becomes

$$\min_{p, \delta} \sum_{i=1}^m \left(\|f(p, t_i + \delta_i) - y_i\|_2^2 + \|\delta_i\|_2^2 \right).$$

In a similar manner, the structure in this more general problem can be taken advantage of

The algorithm by Boggs, Byrd and Schnabel[67, 68] incorporates a full trust region strategy. The nonlinear case is also treated using a stabilized Gauss–Newton methods. Schwetlik and Tiller [527] use a partial Marquardt type regularization where only the Δx part of \tilde{J} is regularized.

9.2.6 Fitting of Circles and Ellipses.

A special nonlinear least squares problem that arises in many areas of 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 the first, called **algebraic fitting**, a least squares functional is used, which directly involves the function $f(x, y, p) = 0$ to be fitted. If (x_i, y_i) , $i = 1 : n$ are given data points we minimize the functional

$$\Phi(p) = \sum_{i=1}^m f^2(x_i, y_i, p).$$

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

$$f(x, y, p) = a (x \ y) \begin{pmatrix} x \\ y \end{pmatrix} + (b_1 \ b_2) \begin{pmatrix} x \\ y \end{pmatrix} + c = 0.$$

We define a parameter vector p and an $m \times 4$ matrix S with rows s_i^T by

$$p = (a, b_1, b_2, c)^T, \quad s_i^T = (x_i^2 + y_i^2, x_i, y_i, 1). \quad (9.2.52)$$

The problem can now be formulated as

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|p\|_2 = 1.$$

Note that the p is defined only up to a constant multiple, which is why the constraint is required. The solution equals 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

$$z = -\frac{1}{2a} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad \rho = \frac{1}{2a} \sqrt{\|b\|_2^2 - 4ac}. \quad (9.2.53)$$

We now discuss the algebraic fitting of ellipses. An ellipse in the (x, y) -plane can be represented algebraically by

$$f(x, y, p) = (x \ y) \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + (b_1 \ b_2) \begin{pmatrix} x \\ y \end{pmatrix} + c = 0. \quad (9.2.54)$$

If we define

$$p = (a_{11}, a_{12}, a_{22}, b_1, b_2, c)^T, \quad s_i^T = (x_i^2, 2x_iy_i, y_i^2, x_i, y_i, 1), \quad (9.2.55)$$

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:**

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|p\|_2 = 1. \quad (9.2.56)$$

The solution of this constrained problem equals the right singular vector corresponding to the smallest singular value of S .

(b) **Linear constraint:**

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad p^T b = 1, \quad (9.2.57)$$

where b is a fixed vector. Assuming $\|b\|_2 = 1$, which is no restriction, and let H be an orthogonal matrix such that $Hb = e_1$. Then the constraint becomes $(Hp)^T e_1 = 1$ so we can write $Sp = (SH^T)(Hp)$, where $Hp = (1q^T)^T$. Now if we partition $SH^T = [sS_2]$ we arrive at the unconstrained problem

$$\min_q \|S_2 q + s\|_2^2, \quad (9.2.58)$$

which is a standard linear least squares problem.

(c) **Quadratic constraint:**

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|Bp\|_2 = 1. \quad (9.2.59)$$

Of particular interest is the choice $B = (0 \ I)$. In this case, if we let $p^T = (p_1, p_2)$ the constraint can be written $\|p_2\|_2^2 = 1$, and is equivalent to a generalized total least squares problem. The solution can then be obtained as follows. First form the QR factorization of S ,

$$S = QR = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}.$$

We can now determine p_2 from the SVD of S and then p_1 from back-substitution in $R_{11}p_1 = -R_{12}p_2$.

It should be stressed that the different constraints above can lead to very different solutions, unless the errors in the fit are small. One desirable property of the fitting algorithm is that when the data is 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 .

The disadvantage of the SVD constraint is its non-invariance under translation and rotations. In case of a linear constraint the choice $b^T = (1 \ 0 \ 1 \ 0 \ 0 \ 0)$, which corresponds to

$$\text{trace}(A) = a_{11} + a_{22} = \lambda_1 + \lambda_2 = 1. \quad (9.2.60)$$

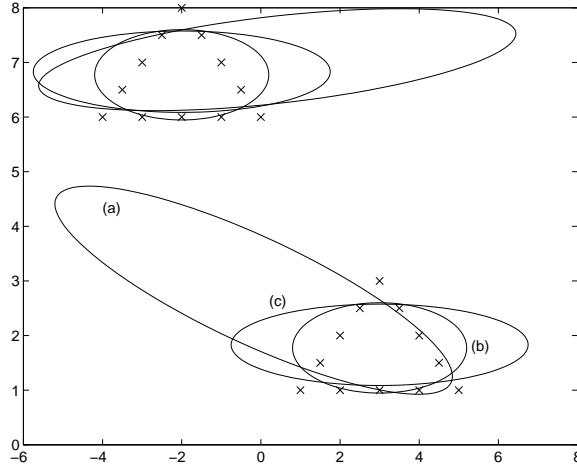


Figure 9.2.3. Ellipse fits for triangle and shifted triangle data: (a) SVD constraint; (b) Linear constraint $\lambda_1 + \lambda_2 = 1$; (c) Bookstein constraint $\lambda_1^2 + \lambda_2^2 = 1$.

gives the desired invariance. This constraint, attributed to Bookstein,

$$\|A\|_F^2 = a_{11}^2 + 2a_{12}^2 + a_{22}^2 = \lambda_1^2 + \lambda_2^2 = 1. \quad (9.2.61)$$

also leads to this kind of invariance. Note that the Bookstein constraint can be put in the form $(0 \ I)$ by permuting the variables and scaling by $\sqrt{2}$.

To construct and plot the ellipse it is convenient to convert the algebraic form (9.2.54) to the parametric form

$$\begin{pmatrix} x(\theta) \\ y(\theta) \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + Q(\alpha) \begin{pmatrix} a \cos(\theta) \\ b \sin(\theta) \end{pmatrix}, \quad Q(\alpha) = \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix}. \quad (9.2.62)$$

The new parameters (x_c, y_c, a, b, α) can be obtained from the algebraic parameters p . The eigendecomposition $A = Q\Lambda Q^T$, where A is the 2×2 matrix in (9.2.54) can be obtained by a Jacobi rotation, see Section 10.4.1. We assume that $a_{12} = 0$ since otherwise $Q = I$ and $\Lambda = A$ is the solution. To determine Λ and Q we first compute

$$\tau = (a_{22} - a_{11})/(2a_{12}), \quad \tan \alpha = t = \text{sign}(\tau)/(|\tau| + \sqrt{1 + \tau^2}).$$

The elements in Q and Λ are then given by

$$\begin{aligned} \cos \alpha &= 1/\sqrt{1 + t^2}, & \sin \alpha &= t \cos \alpha, \\ \lambda_1 &= a_{11} - t a_{12}, & \lambda_2 &= a_{22} + t a_{12}. \end{aligned}$$

If we introduce the new coordinates $z = Q\tilde{z} + s$ in the algebraic form (9.2.54) 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 equals

$$s = \begin{pmatrix} x_c \\ y_c \end{pmatrix} = -\frac{1}{2} A^{-1} b = -\frac{1}{2} A^{-1} Q \Lambda^{-1} (Q^T b), \quad (9.2.63)$$

and the axis (a, b) of the ellipse are given by

$$\begin{pmatrix} a \\ b \end{pmatrix} = \sqrt{-\tilde{c}} \operatorname{diag} \Lambda^{-1/2}, \quad \tilde{c} = c + \frac{1}{2} b^T s = -\frac{1}{2} \tilde{b}^T \Lambda^{-1} \tilde{b}. \quad (9.2.64)$$

In **geometric fitting** of data (x_i, y_i) , $i = 1 : m$ to a curve of the form $f(x, y, p) = 0$, where the orthogonal distance $d_i(p)$ is first measured from each data point to the curve, where

$$d_i^2(p) = \min_{f(x,y,p)=0} ((x - x_i)^2 + (y - y_i)^2).$$

Then the problem

$$\min_p \sum_{i=1}^m d_i^2(p)$$

is solved. This is similar to orthogonal distance regression described for an explicitly defined function $y = f(x, \beta)$ in Section 9.2.5.

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

$$f(x, y, p) = \begin{pmatrix} x - x_c - r \cos \phi \\ y - y_c - r \sin \phi \end{pmatrix} = 0, \quad (9.2.65)$$

where $p = (x_c, y_c, r)^T$. The problem can be written as a nonlinear least squares problem

$$\min_{p, \phi_i} \|r(p, \phi)\|_2^2, \quad \phi = (\phi_1, \dots, \phi_m), \quad (9.2.66)$$

where

$$r = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} \in \mathbf{R}^{2m}, \quad r_i = \begin{pmatrix} x_i - x_c - r \cos \phi_i \\ y_i - y_c - r \sin \phi_i \end{pmatrix}.$$

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 matrix, 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 the rows the Jacobian associated with this problem has the form

$$J = \begin{pmatrix} rS & A \\ -rC & B \end{pmatrix} \underbrace{\}_{m \times m}},$$

where

$$S = \text{diag}(\sin \phi_i), \quad C = \text{diag}(\cos \phi_i), \quad (9.2.67)$$

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 with an orthogonal matrix we obtain

$$Q^T J = \begin{pmatrix} rI & SA - CB \\ 0 & CA + SB \end{pmatrix}, \quad Q = \begin{pmatrix} S & C \\ -C & S \end{pmatrix}.$$

To obtain the QR factorization of J we only need to compute the QR factorization of the $m \times 3$ matrix $CA + SB$.

A Gauss–Newton type method with a trust region strategy can be implemented using this QR factorization of the Jacobian. Good starting values for the parameters may often be obtained using an algebraic fit as described in the previous section. Experience shows that the amount of computation involved in a geometric fit is at least an order of magnitude more than for an algebraic fit.

For the geometric fit of an ellipse we use the parametric form

$$f(x, y, p) = \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} - Q(\alpha) \begin{pmatrix} a \cos \phi \\ b \sin \phi \end{pmatrix} = 0. \quad (9.2.68)$$

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 written as a nonlinear least squares of the form (9.2.66), where

$$r_i = \begin{pmatrix} x_i - x_c \\ y_i - y_c \end{pmatrix} - Q(\alpha) \begin{pmatrix} a \cos \phi_i \\ b \sin \phi_i \end{pmatrix}.$$

We thus have $2m$ nonlinear equations for $m+5$ unknowns ϕ_1, \dots, ϕ_m and x_c, y_c, a, b, α . To construct the Jacobian we need the partial derivatives

$$\frac{\partial r_i}{\partial \phi_i} = Q(\alpha) \begin{pmatrix} -a \sin \phi_i \\ b \cos \phi_i \end{pmatrix}, \quad \frac{\partial r_i}{\partial \alpha} = -\frac{d}{d\alpha} Q(\alpha) \begin{pmatrix} a \cos \phi_i \\ b \sin \phi_i \end{pmatrix},$$

and

$$\frac{\partial r_i}{\partial a} = -Q(\alpha) \begin{pmatrix} \cos \phi_i \\ 0 \end{pmatrix}, \quad \frac{\partial r_i}{\partial b} = -Q(\alpha) \begin{pmatrix} 0 \\ \sin \phi_i \end{pmatrix}.$$

Note that

$$\frac{d}{d\alpha} Q(\alpha) = \begin{pmatrix} -\sin \alpha & \cos \alpha \\ -\cos \alpha & -\sin \alpha \end{pmatrix} = Q \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

After a reordering of the rows the Jacobian associated with this problem has the form

$$J = U \begin{pmatrix} -aS & A \\ bC & B \\ m & 3 \end{pmatrix} \}_{m \times 2m}, \quad S = \text{diag}(\sin \phi_i), \quad C = \text{diag}(\cos \phi_i).$$

where $U = -\text{diag}(Q, \dots, Q) \in \mathbf{R}^{2m \times 2m}$ is a block diagonal orthogonal matrix and S and C given by (9.2.67). The i th row of the matrices $A \in \mathbf{R}^{m \times 5}$ and $B \in \mathbf{R}^{m \times 5}$ are

$$\begin{aligned} a_i^T &= (-b \sin \phi_i \quad \cos \phi_i \quad 0 \quad \cos \alpha \quad \sin \alpha), \\ b_i^T &= (a \cos \phi_i \quad 0 \quad \sin \phi_i \quad -\sin \alpha \quad \cos \alpha). \end{aligned}$$

The first m columns of $U^T J$ can be diagonalized using a sequence of Givens rotations, where the i th rotation zeros the second component in the vector

$$\begin{pmatrix} -a \sin \phi_i \\ b \cos \phi_i \end{pmatrix}, \quad i = 1 : m.$$

The fitting of a sphere or an ellipsoid can be treated analogously. The sphere can be represented in parametric form as

$$f(x, y, z, p) = \begin{pmatrix} x - x_c - r \cos \theta \cos \phi \\ y - y_c - r \cos \theta \sin \phi \\ z - z_c - r \sin \theta \end{pmatrix} = 0, \quad (9.2.69)$$

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 matrix can simply be brought into upper triangular form; cf. Computer Exercise 2.

When the data covers 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. Also the fitting of a sphere or an ellipsoid to near planar data gives rise to ill-conditioned problems.

Review Questions

- 2.1.** Consider the unconstrained optimization problem $\min_x \phi(x)$, $x \in \mathbf{R}^n$. Give necessary conditions for x^* to be a local minimum. ($\phi(x) : \mathbf{R}^n \rightarrow \mathbf{R}$ is assumed to be twice continuously differentiable.)

- 2.2.** (a) In many iterative methods for minimizing a function $\phi(x)$, a sequence of points are generated from $x_{k+1} = x_k + \lambda_k d_k$, $k = 0, 1, 2, \dots$, where d_k is a search direction. When is d_k a descent direction? Describe some strategies to choose the step length λ_k .
 (b) Define the Newton direction. When is the Newton direction a descent direction?
- 2.3.** In quasi-Newton, or variable metric methods an approximate Hessian is built up as the iterations proceed. Denote by B_k the approximate Hessian at the k th step. What quasi-Newton condition does B_k satisfy, and what is the geometrical significance of this condition?
- 2.4.** (a) What property should the function $f(x)$ have to be unimodal in $[a, b]$?
 (b) Describe an interval reduction method for finding the minimum of a unimodal function in $[a, b]$, which can be thought of as being an analogue of the bisection method. What is its rate of convergence?
- 2.5.** Describe the damped Gauss–Newton method with a recommended step length procedure.
- 2.6.** How does the Gauss–Newton method differ from the full Newton method? When can the behavior of the Gauss–Newton method be expected to be similar to that of Newton’s method?
- 2.7.** What is meant by a separable nonlinear least squares problem? Describe a recommended method. Give an important example.
- 2.8.** Consider fitting observations (y_i, t_i) , $i = 1 : m$ to the model $y = g(p, t)$, where y and t are scalar variables and $p \in \mathbf{R}^n$ are parameters to be determined. Formulate the method of orthogonal distance regression for this problem.

Problems

- 2.1.** (a) The general form for a quadratic function is

$$\phi(x) = \frac{1}{2}x^T G x - b^T x + c,$$

where $G \in \mathbf{R}^{n \times n}$ is a symmetric matrix and $b \in \mathbf{R}^n$ a column vector. 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 from (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 minimum point?

2.2. Let $\psi(x)$ be quadratic with Hessian matrix G , which need not be positive definite.

(a) Let $\psi(\lambda) = \phi(x_0 - \lambda d)$. Show using Taylor's formula that

$$\psi(\lambda) = \psi(0) - \lambda g^T d + \frac{1}{2} \lambda^2 d^T G d.$$

Conclude that if $d^T G d > 0$ for a certain vector d then $\psi(\lambda)$ is minimized when $\lambda = g^T d / d^T G d$, and

$$\min_{\lambda} \psi(\lambda) = \psi(0) - \frac{1}{2} \frac{(d^T g)^2}{d^T G d}.$$

(b) Using the result from (a) show that if $g^T G g > 0$ 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 $g^T G^{-1} g > (g^T g)^2 / g^T G g$. (The conclusion holds also if $\phi(x_0 - \lambda d)$ can be approximated by a quadratic function of λ reasonably well in the relevant intervals.)

(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 minimum point?

2.3. Show that Lemma 9.2.3 holds for any generalized inverse Φ^- which satisfies $\Phi \Phi^- \Phi = \Phi$ and $(\Phi \Phi^-)^T = \Phi \Phi^-$.

2.4. 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 . The problem

$$\min_{x_0, y_0, r} \sum_{i=1}^m d_i^2(x_0, y_0, r)$$

is thus a nonlinear least squares problem. An approximative 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 down 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 $m < 3$.

- 2.5.** Generalize the algorithm described in Section 11.3.9 to fit a sphere to three-dimensional data (x_i, y_i, z_i) , $i = 1 : m$.

9.3 Linear Programming

9.3.1 Optimality for Linear Inequality Constraints

A **linear programming** (LP) problem is an optimization problem with a linear objective function, where the domain of the variables are restricted by a system of linear equations and inequalities, the problem. Often the following formulation of an LP problem is used:

$$\begin{aligned} \min_{x \in \mathbf{R}^n} & c^T x \\ \text{subject to } & Ax \geq b. \end{aligned} \tag{9.3.1}$$

Here $x \in \mathbf{R}^n$ is the vector of unknowns, $c \in \mathbf{R}^n$ is the **cost vector**, and $A \in \mathbf{R}^{m \times n}$ the constraint matrix. The function $c^T x$ to be minimized is called the **objective** function. (Note that the problem of minimizing $c^T x$ is equivalent to maximizing $-c^T x$.)

The LP problem arose from the need of planning logistics support during World War II, and has since been studied extensively LP problems also come up in economics, strategic planning, transportation and production 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 today have become much larger than first envisioned. Problems with several million variables are now routinely solved!

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 inequality constraint Any point x , which has the form $a_i^T x \geq b_i$. The corresponding equality $a_i^T x = b_i$ defines a hyperplane in \mathbf{R}^n . The inequality restricts x to lie on the feasible side of this hyperplane. A point which satisfies all the inequality constraints, is said to be a **feasible point**. The set of all feasible points is called the **feasible region**. For the linear programming problem (9.3.1) the feasible region is

$$\mathcal{F} = \{x \in \mathbf{R}^n \mid Ax \geq b\}. \tag{9.3.2}$$

A solution to the LP (9.3.1) can exist only if the feasible region \mathcal{F} is not empty. In general, it is not possible to determine directly if this is the case. Indeed, the problem of finding a feasible point to (9.3.1) is usually posed and solved as a linear programming problem.

When \mathcal{F} is not empty, it has the important property of being a **convex set**. That is, 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 simple to verify that \mathcal{F} defined by (9.3.2) has this property, since

$$Az = (1 - \alpha)Ax + \alpha Ay \geq (1 - \alpha)b + \alpha b = b.$$

when $0 \leq \alpha \leq 1$. If at a feasible point x the inequality constraint $a_i^T x \geq b_i$ is satisfied with equality, i.e.,

$$r_i(x) = a_i^T x - b_i = 0,$$

then it is said to be **active**. The **active set** is the subset of the inequality constraints $Ax \geq b$ which are active at x .

Figure 9.3.1. Geometric illustration of a linear programming problem.

Example 9.3.1.

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 5 minutes, M_2 3 minutes, and M_3 4 minutes. The corresponding figures for one unit of P_2 are: M_1 1 minute, M_2 4 minutes, and M_3 3 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?

Suppose that 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

$$\begin{aligned} 5x_1 + x_2 &\leq 60 && \text{for } M_1, \\ 3x_1 + 4x_2 &\leq 60 && \text{for } M_2, \\ 4x_1 + 3x_2 &\leq 60 && \text{for } M_3. \end{aligned} \tag{9.3.3}$$

The problem is illustrated geometrically in Figure 9.4.1. The first of the inequalities (9.3.4) can be interpreted that the solution (x_1, x_2) must lie 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 (x_1, x_2) 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 and the dashed line $f = 0$; it clearly takes on its largest value at the vertex B . Since every vertex is the intersection of two lines, we must have equality in (at least) two of the inequalities. At the solution x^* equality holds in the inequalities for M_1 and M_3 . These two constraints are called **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_2 continuously, while M_2 is used only $600/11 = 54.55$ minutes per hour.

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 active constraints matrix has rank n . Thus at least n constraints are active at a 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 9.3.1 there are five vertices O, A, B, C , and D , all of which are nondegenerate. The vertices form a polyhedron, or **simplex** in \mathbf{R}^n .

Vertices are of central importance in linear programming since many LP have the property that a minimizer lies at a vertex. The following theorem states the conditions under which this is true.

Theorem 9.3.1.

Consider the linear program of

$$\min_{x \in \mathbf{R}^n} c^T x \quad \text{subject to} \quad Ax \geq b,$$

where $A \in \mathbf{R}^{m \times n}$. If $\text{rank}(A) = n$ and the optimal value of $c^T x$ is finite, a vertex minimizer exists.

Note that by convexity an infinity of non-vertex solutions will exist if the minimizer is not unique. For example, in a problem like Example 9.3.1, one could have an objective function $f = c^T x$ such that the line $f = 0$ were parallel to one of the sides of the pentagon. Then all points on the line segment between two optimal vertices in the polyhedron are also optimal points. If the linear program also includes the constraints $x \geq 0$, then the constraint matrix has the form

$$\begin{pmatrix} A \\ I_n \end{pmatrix} \in \mathbf{R}^{(m+n) \times n}.$$

Since the rows include the identity matrix I_n this matrix always has rank n . Hence a feasible vertex must exist if any feasible point exists.

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 $a_i^T p \geq 0$. It is not difficult to see that the feasible directions

p are not affected by the inactive constraints at x . Hence p is a feasible directions 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 to consider the set of inactive constraints i for which $a_i^T p < 0$. For these constraints, which are called decreasing constraints, $a_i^T(x + \alpha p) = a_i^T x + \alpha a_i^T p = b_i$, and thus the constraint i becomes active when

$$\alpha = \alpha_i = \frac{a_i^T x - b_i}{-a_i^T p}.$$

Hence the largest step we can take along p is $\max \alpha_i$ where we maximize over all decreasing constraints.

For a linear program there are three possibilities: There may be no feasible points, in which case the LP has no solution; there may be a feasible point x^* at which the objective function is minimized; Finally, the feasible region may be unbounded and the objective function unbounded below in the feasible region.

We first derive conditions under which a feasible point x^* to an inequality constrained linear program is *not* a minimizer. We say that $p \neq 0$ is a **descent direction** if $c^T p < 0$, i.e., if the objective functions decreases if we move in the direction of p . Further, p is a feasible descent direction at x^* if for some $\tau > 0$ we have

$$A(x^* + \alpha p) \geq b \quad \forall \quad 0 < \alpha \leq \tau.$$

Clearly the feasibility of p depends only on the constraints active at x^* . Let A_A be the matrix composed of rows of A which correspond to these active constraints. Then p is a feasible descent direction if

$$A_A p \geq 0 \quad \text{and} \quad c^T p < 0.$$

We conclude that a feasible point x is a minimizer for the linear program

$$\min_{x \in \mathbf{R}^n} c^T x \quad \text{subject to} \quad Ax \geq b,$$

if and only if $c^T p \geq 0$ for all p such that $A_A p \geq 0$.

To relate these necessary and sufficient conditions to properties of A and c we need to use **Farkas' Lemma**, 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. Therefore we refer to [255, Sec. 7.7] for a proof.

Lemma 9.3.2 (Farkas' Lemma).

Let $A \in \mathbf{R}^{m \times n}$ be a matrix and $g \in \mathbf{R}^n$ be a nonzero vector. Then either the primal system

$$Ax \geq 0 \quad \text{and} \quad g^T x < 0$$

has a solution $x \in \mathbf{R}^n$, or the dual system

$$A^T y = g \quad \text{and} \quad y \geq 0$$

has a solution $y \in \mathbf{R}^m$, but neither both.

Using Farkas' lemma we can now obtain the following main result:

Theorem 9.3.3.

Assume that x^* is a feasible point to the linear program

$$\min_{x \in \mathbf{R}^n} c^T x \quad \text{subject to} \quad Ax \geq b,$$

Then the following holds:

- (a) If x^* satisfies the conditions

$$c = A_A^T \pi^* + \lambda^*, \quad \lambda^* \geq 0,$$

where A_A 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.

Remark 9.3.1.

A new and more elementary proof of optimality conditions for linear programming has been given by Forsgren [210]. This is built on perturbation of the constraints and uses neither Farkas lemma nor the Simplex method.

9.3.2 Standard Form for LP

It is convenient to adopt the following slightly different **standard form** of a linear programming problem:

$$\min_{x \in \mathbf{R}^n} c^T x \quad \text{subject to} \quad Ax = b, \quad x \geq 0. \quad (9.3.4)$$

where $A \in \mathbf{R}^{m \times n}$. Here the constraints $x \geq 0$ (or **simple bounds**) are the only inequality constraints. The set \mathcal{F} of feasible points consists of points x that satisfy $Ax = b$ and $x \geq 0$. If $\text{rank}(A) = n$ this set contains just one point if $A^{-1}b \geq 0$; otherwise it is empty. Hence in general we have $\text{rank}(A) < n$.

It is simple to convert a linear programming problem to the standard form (9.3.4). Many LP software packages apply an automatic internal conversion to this standard form. The change of form involves modification of the dimensions, variables and constraints. 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 problems with inequality constraints is converted, the number of variables will increase. If the original constraints are $Ax \leq b$, $A \in \mathbf{R}^{m \times n}$, 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 9.3.2.

The problem in Example 9.3.1 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 Figure 9.4.1) 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.

For completeness we note that, although this is seldom used in practice, equality constraints can be converted to inequality constraints. For example, $a_i^T x = b_i$ is equivalent to the two inequality constraints $a_i^T x > b_i$ and $-a_i^T x \geq -b_i$.

For $p \neq 0$ to be a feasible descent direction with respect to the compatible linear equality constraints $Ax = b$ we must require that

$$c^T p < 0 \quad \text{and} \quad Ap = 0.$$

The second condition says that p belongs to 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 , the 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 9.3.4.

A feasible point x^ for the standard linear program*

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{subject to} \quad Ax = b, \quad x \geq 0,$$

is minimizer if and only if

$$c = A^T \lambda^*, \quad \lambda^* \geq 0, \quad \lambda_i^* x_1^* = 0, \quad i = 1 : n. \quad (9.3.5)$$

The last condition in the theorem is known as a **complementarity condition**. Note that since both x^* and η^* are nonnegative the condition implies that if $x_1^* > 0$ then $\eta_1^* = 0$ and vice versa.

A vertex for a standard form problem is also called a **basic feasible point**.

In case more than $n - m$ coordinates are zero at a feasible point we say that the point is a **degenerate** feasible point. A feasible vertex must exist if any feasible point exists. Since the m equality constraints are active at all feasible points, 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.

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 which 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 9.3.5.

For a linear programming problem in standard form some optimal feasible point is also a basic feasible point, i.e., at least $n - m$ of its coordinates are zero; equivalently at most m coordinates are strictly positive.

The standard form given above has the drawback that when variables are subject to lower and upper bounds, these bounds have to be entered as general constraints in the matrix A . Since lower and upper bounds on x can be handled much more easily, a more efficient formulation is often used where inequalities $l \leq x \leq u$ are substituted for $x \geq 0$. Then it is convenient to allow $l_i = -\infty$ and $u_i = \infty$ for some of the variables x_i . If for some j , $l_j = -\infty$ and $u_j = \infty$, x_j is said to be free, and if for some j , $l_j = u_j$, x_j is said to be fixed. For simplicity, we consider in the following mainly the first standard form.

Example 9.3.3.

As a nontrivial example of the use of Theorem 9.3.1 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 which 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 equals c_{ij} . The problem is to determine the quantities x_{ij} transported so that the total cost is minimized. This problem 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, since

$$\sum_{i=1}^I \sum_{j=1}^J x_{ij} - \sum_{j=1}^J \sum_{i=1}^I x_{ij} = 0.$$

The number of linearly independent equations is thus (at most) equal to $m = I + J - 1$. From Theorem 9.3.1 it follows that *there exist 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 which make use of the special structure of the equations.

Many other 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.

9.3.3 The Simplex Method

The **simplex method** was devised in 1947 by George Danzig, who was working at Rand Corporation. Until the late 1980s it was the only effective method for solving large linear programming problems. The simplex method is now rivaled by so called interior-point methods (see Section 9.3.5), but it is still competitive for many classes of problems.

The simplex method is based on generating a path through the set of feasible vertices that are adjacent. From Theorem 9.3.1 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 $\binom{n}{n-m}$ possible ways of setting $n - m$ variables equal to zero, sorting out those combinations which do not give feasible points. The rest are vertices of the polyhedron, and one can look among these to find a vertex at which f is minimized. However, since the number of vertices increases exponentially with $n - m$ this is intractable even for small values of m and n .

The simplex method starts at a vertex (basic feasible point) and recursively proceeds from one vertex to an adjacent vertex with a lower value of the objective function $c^T x$. The first phase in the simplex method is to determine an initial basic feasible point (vertex). In some cases an initial vertex can be trivially found (see, e.g., Example 9.3.4 below). A systematic method which can be used in more difficult situations will be described later in Section 9.3.3.

When an initial feasible point has been found, the following steps are repeated until convergence:

- I. Check if the current vertex is an optimal solution. If so, then stop, else continue.
- II. Proceed from the current vertex to a neighboring vertex at which the value of f if possible is smaller.

Consider the standard form linear programming problem (9.3.4). At a vertex $n - m$ variables are zero. We divide the index set $I = \{1 : m\}$ into two disjoint sets

$$I = B \cup N, \quad B = \{j_1, \dots, j_n\}, \quad N = \{i_1, \dots, i_{n-m}\}, \quad (9.3.6)$$

such that N corresponds to the zero variables. We call x_B **basic variables** and x_N **nonbasic variables**. If the vector x and the columns of the matrix A are split in a corresponding way, we can write the system $Ax = b$ as

$$A_B x_B = b - A_N x_N. \quad (9.3.7)$$

We start by illustrating the simplex method on the small example from the introduction.

Example 9.3.4.

In Example 9.3.2 we get an initial feasible point 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

$$A_B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad A_N = \begin{pmatrix} 5 & 1 \\ 3 & 4 \\ 4 & 3 \end{pmatrix},$$

Putting $x_N = 0$ gives $\hat{x}_B = b = (60, 60, 60)^T$. Since $x_B \geq 0$ this corresponds to a vertex (the vertex O in Figure 9.4.1) for which $f = 0$. The optimality criterion is not fulfilled since $c_B = 0$ and $\hat{c}_N^T = c_N^T = (-30, -20) < 0$. If we choose $x_r = x_1 = \theta > 0$, then using \hat{x}_B and the first column of $A_B^{-1} A_N = A_N$, we find

$$\theta_{max} = 60 \min_i \{1/5, 1/3, 1/4\} = 12.$$

Clearly a further increase in x_1 is inhibited by x_3 . We now exchange these variables to get $x_B = (x_1, x_4, x_5)^T$, and $x_N = (x_3, x_2)^T$. (Geometrically this means that one goes from O to A in Figure 9.4.1.)

The new sets of basic and non-basic variables are $x_B = (x_1, x_4, x_5)^T$, and $x_N = (x_3, x_2)$. Taking $x_N = 0$ we have

$$\hat{x}_B = (12, 24, 12)^T, \quad f = 0 + 12 \cdot 30 = 360.$$

The new splitting of A is

$$A_B = \begin{pmatrix} 5 & 0 & 0 \\ 3 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix}, \quad A_N = \begin{pmatrix} 1 & 1 \\ 0 & 4 \\ 0 & 3 \end{pmatrix}.$$

The reduced costs $\hat{c}_N^T = (6 \ -14)$ for non-basic variables are easily computed from (9.3.10). The optimality criterion is not satisfied. We take $x_r = x_2$ and solve $A_B b_2 = a_2$ to get $b_2 = (1/5)(1, 17, 11)^T$. We find $\theta = 5 \min(12/1, 24/17, 12/11) = 60/11$. Exchanging x_2 and x_5 we go from A to B in Figure 9.4.1. The new basic variables are

$$\hat{x}_B = (x_1, x_4, x_2) = (5/11)(24, 12, 12)^T, \quad f = 4,800/11.$$

The non-basic variables are $x_N = (x_3, x_5)^T$, and to compute the reduced costs we must solve

$$\begin{pmatrix} 5 & 3 & 4 \\ 0 & 1 & 0 \\ 1 & 4 & 3 \end{pmatrix} d = \begin{pmatrix} -30 \\ 0 \\ -20 \end{pmatrix}.$$

We have $c_N^T = (0, 0)$ and get $\hat{c}_N^T = (d_1, d_3) = \frac{10}{11}(1 \ 7)$. The optimality criterion is now satisfied, so we have found the optimal solution.

Each step of the Simplex method requires the solution of three systems of linear equations involving a common coefficient matrix $B_N = A_B^{-1}A_N$. It is common practice to solve these systems by updating the inverse in each step and then applying it to the right hand sides. These calculations can be presented in form of a **tableau**; see Problem 9.4.3. These implementations are costly and potentially unstable, since they do not allow for pivoting for size.

We now give a general description of the steps in the simplex method which is closer to what is used in current simplex codes. We assume that the matrix A_B in (9.3.7) is nonsingular. (This will always be the case if $\text{rank}(A) = n$.) We can then express the basic variables in terms of the nonbasic

$$x_B = \hat{x}_B - A_B^{-1}A_N x_N, \quad \hat{x}_B = A_B^{-1}b, \quad (9.3.8)$$

where \hat{x}_B is obtained by solving the linear system $A_B \hat{x}_B = b$. If $\hat{x}_B \geq 0$ then $x_N = 0$ corresponds to a basic feasible point (vertex). The vector c is also split in two subvectors c_B and c_N , and using (9.3.8) we have

$$f = c^T x = c_B^T(\hat{x}_B - A_B^{-1}A_N x_N) + c_N^T x_N = c_B^T \hat{x}_B + \hat{c}_N^T x_N,$$

where

$$\hat{c}_N = c_N - A_N^T d, \quad d = A_B^{-T} c_B. \quad (9.3.9)$$

Here d can be computed by solving the linear system

$$A_B^T d = c_B. \quad (9.3.10)$$

The components of \hat{c}_N are known as the **reduced costs** for the nonbasic variables, and the process of computing them known as **pricing**. If $\hat{c}_N \geq 0$ then $x_N = 0$ corresponds to an optimal point, since f cannot decrease when one gives one (or more) nonbasic variables positive values (negative values are not permitted). Hence if the **optimality criterion** $\hat{c}_N \geq 0$ is satisfied, then the solution $x_B = \hat{x}_B$, $x_N = 0$ is optimal, and we can stop.

If the optimality criterion is *not* satisfied, then there is at least one non-basic variable x_r whose coefficient \hat{c}_r in \hat{c}_N is negative. We now determine the largest positive increment one can give x_r without making any of the basic variables negative, while holding the other non-basic variables equal to zero. Consider equation (9.3.8), and let b_r be the corresponding column of the matrix $A_B^{-1}A_N$. This column can be determined by solving the linear system

$$A_B b_r = a_r, \quad (9.3.11)$$

where a_r is the column in A_N corresponding to x_r . If we take $x_r = \theta > 0$, then $x_B = \hat{x}_B - \theta b_r$, and for any basic variable x_i we have $x_i = \hat{x}_i - \theta b_{ir}$. Hence if $b_{ir} > 0$, then x_i remains positive for $\theta = \theta_i \leq \hat{x}_i/b_{ir}$. The largest θ for which no basic variable becomes negative is given by

$$\theta = \min_i \theta_i, \quad \theta_i = \begin{cases} \hat{x}_i/b_{ir} & \text{if } b_{ir} > 0; \\ +\infty & \text{if } b_{ir} \leq 0; \end{cases} \quad (9.3.12)$$

If $\theta = +\infty$, the object function is unbounded in the feasible region, and we stop. Otherwise there is at least one basic variable x_l that becomes zero for this value of θ . Such a variable is now interchanged with x_r , so x_r becomes a basic variable and x_l a non-basic variable. (Geometrically this corresponds to going to a neighboring vertex.) Note that the new values of the basic variables can easily be found by updating the old values using $x_i = x_i - \theta b_{ir}$, and $x_r = \theta$.

In case several components of the vector \hat{c}_N are negative 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 object 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 scalings of the matrix A . A scaling invariant strategy called the **steepest edge strategy** can lead to great gains in efficiency, see Gill, Murray, and Wright [255, Chap. 8].

It is possible that even at a vertex which is not an optimal solution one cannot increase 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 non-basic variables are zero. As mentioned previously, such a point is called a **degenerate vertex**. In such a case one has to exchange a non-basic variable with one of the basic variables which is zero at the vertex, and a step with $\theta = 0$ occurs. In more difficult cases, it may even be possible to make several such exchanges.

Figure 9.3.2. Feasible points in a degenerate case.

Example 9.3.5.

Suppose we want to maximize $f = 2x_1 + 2x_2 + 3x_3$ subject to the constraints

$$x_1 + x_3 \leq 1, \quad x_2 + x_3 \leq 1, \quad x_i \geq 0, \quad i = 1, 2, 3.$$

The feasible points form a four-sided pyramid in (x_1, x_2, x_3) -space; see Figure 9.4.2. Introduce slack variables x_4 and x_5 , and take $\{x_1, x_2, x_3\}$ as non-basic variables.

This gives a feasible point since $x_1 = x_2 = x_3 = 0$ (the point O in Figure 9.4.2) satisfies the constraints. Suppose at the next step we move to point A , by exchanging x_3 and x_4 . At this point the non-basic variables $\{x_1, x_2, x_4\}$ are zero but also x_5 , and A is a degenerate vertex, and we have

$$\begin{aligned}x_3 &= 1 - x_1 - x_4, \\x_5 &= x_1 - x_2 + x_4, \\f &= 3 - x_1 + 2x_2 - 3x_4.\end{aligned}$$

The optimality condition is not satisfied, and at the next step we have to exchange x_2 and x_5 , and remain at point A . In the final step we can now exchange x_1 and x_2 to get to the point B , at which

$$\begin{aligned}x_1 &= 1 - x_3 - x_4, \\x_2 &= 1 - x_3 - x_5, \\f &= 4 - x_3 - x_4 - 2x_5.\end{aligned}$$

The optimality criterion is fulfilled, and so B is the optimal point.

Traditionally, degeneracy has been a major problem with the Simplex method. A proof that the simplex algorithm converges after a finite number of steps relies on a strict increase of the objective function in each step. When steps in which f does not increase 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 which prevent cycling by allowing slightly infeasible points, see Gill, Murray, and Wright [255, Sec. 8.3.3]. By perturbing each bound by a small random 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 with the solution of the two systems of equations $A_B^T d = c_B$ and $A_B b_r = a_r$. We note that both the matrix A_B and the right hand sides c_B and a_r are often very sparse. In the original simplex method these systems were solved by recurring the inverse A_B^{-1} of the basis matrix. This is in general inadvisable, because of lack of numerical stability.

Stable methods have been devised which store and update LU factorizations

$$P_B A_B = LU \tag{9.3.13}$$

where P_B is a permutation matrix: see [36]. The initial factorization (9.3.13) is computed by Gaussian elimination using partial pivoting. The new basis matrix which results from dropping the column a_r and inserting the column a_s in the last position is a Hessenberg matrix. Special methods can therefore be used to generate the factors of the subsequent basis matrices as columns enter or leave.

From the above it is clear that the major computational effort in a simplex step is the solution of the two linear systems

$$A_B^T \hat{d} = c_B, \quad A_B b_r = a_r, \tag{9.3.14}$$

to compute reduced costs and update the basic solution. These systems can be solved cheaply by computing a LU factorization of the matrix A_B if available. For large problems it is essential to take advantage of sparsity in A_B . In particular the initial basis should be chosen such that A_B has a structure close to diagonal or triangular. Therefore row and column permutations are used to bring A_B into such a form. Assume that a LU factorization has been computed for the initial basis. Since in each step only *one column* in A_B is changed, techniques for updating a (sparse) LU factorization play a central role in modern implementation of the simplex method.

Although the worst case behaviour 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 methods tends to give the exact result after about $2m-3m$ steps, and essentially independent 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 optimal feasible point. In some cases it may be possible to start from the optimal solution of a nearby problem. (This is sometimes called “a warm start”.)

It may not be trivial to decide if a feasible point exists, and if so, how to find one. By adding a sufficient number of new **artificial variables** to the constraints in (9.3.4), the problem can be modified so that an initial bases matrix A_B can be found,

$$x_B = A_B^{-1}b \geq 0, \quad x_N = 0, \quad x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}.$$

The artificial variables are driven towards zero by introducing large positive costs associated with them in the initial phase of the Simplex algorithm. If a feasible point exists, then eventually all artificial variables will become non-basic variables and can be dropped. This is often called the **phase 1** in the solution of the original linear program. The following example illustrates this technique.

Example 9.3.6.

Maximize $f = x_1 - x_2$, subject to the constraints $x_i \geq 0$, $i = 1 : 5$, and

$$\begin{aligned} x_3 &= -2 + 2x_1 - x_2, \\ x_4 &= 2 - x_1 + 2x_2, \\ x_5 &= 5 - x_1 - x_2. \end{aligned}$$

If $x_1 = x_2 = 0$, then x_3 is negative, and hence x_1, x_2 cannot be used as non-basic variables. It is not immediately obvious which pair of variables suffice as non-basic variables. If we introduce a new artificial variable $x_6 \geq 0$, defined by

$$x_6 = 2 - 2x_1 + x_2 + x_3,$$

then we can take x_4, x_5, x_6 as basic variables. We thus have found a feasible point for an extended problem with six variables. This problem has the same solution as the original if we can ensure that the artificial variable x_6 is zero at the solution. To accomplish this we modify the objective function to become

$$\bar{f} = x_1 - x_2 - Mx_6 = -2M + (1 + 2M)x_1 - (1 + M)x_2 - Mx_3.$$

Here M is assumed to be a large positive number, much larger than other numbers in the computation. Then a positive value of x_6 will tend to make the function to be maximized quite small, which forces the artificial variable to become zero at the solution. Indeed, as soon as x_6 appears as a nonbasic variable, (this will happen if x_1 and x_6 are exchanged here) it is no longer needed in the computation, and can be deleted, since we have found an initial feasible point for the original problem.

The technique sketched above 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 A_B has a diagonal or triangular structure. Several such basis selection algorithms, named basis crashes, have been developed, see Bixby [54].

9.3.4 Duality

Consider the linear programming problem in standard form

$$\begin{aligned} & \min_{x \in \mathbf{R}^n} c^T x \\ & \text{subject to } Ax = b, \quad x \geq 0. \end{aligned}$$

When this problem has a bounded optimal minimizer x^* The optimality conditions of Theorem 9.3.4 imply the existence of Lagrange multipliers y^* such that

$$c = A^T y^* + \eta^*, \quad \eta^* \geq 0, \quad \eta_i^* x_i^* = 0, \quad i = 1, \dots, n.$$

It follows that y^* satisfies the inequality constraints $y^T A \leq c^T$. This leads us to define the **dual problem** to the standard form problem as follows:

$$\begin{aligned} & \max_{y \in \mathbf{R}^m} g = y^T b \\ & \text{subject to } y^T A \leq c^T. \end{aligned} \tag{9.3.15}$$

Here y are the **dual variables**. The initial problem will be called the **primal problem** and x the **primal variables**. If y satisfies the inequality in (9.3.15) y is called a feasible point of the dual problem. Note that the constraint matrix f of the dual problem is the transposed constraint matrix of the primal, the right-hand side in the dual is the normal vector of the primal objective, and the normal vector of the dual objective is the right-hand side of the primal.

Note that the dual to a standard form linear programming problem is in all inequality form. However, the dual problem may also be written in standard form

$$\begin{aligned} & \max_{y \in \mathbf{R}^m} g = y^T b \\ & \text{subject to } A^T y + z = c, \quad z \geq 0, \end{aligned} \tag{9.3.16}$$

where z are the dual slack variables. The solution y^* to the dual problem is the Lagrange multiplier for the m linear equality constraints in the primal problem. The

primal solution x^* is the Lagrange multiplier for the n linear equality constraints of the standard-form dual problem.

Let x and y be arbitrary feasible vectors for the primal and dual problems, respectively. Then

$$g(y) = y^T b = y^T A x \leq c^T x = f(x). \quad (9.3.17)$$

The nonnegative quantity

$$c^T x - y^T b = x^T z$$

is called the **duality gap**. We will show it is zero if and only if x and y are optimal for the primal and dual.

Theorem 9.3.6.

The optimal values of the primal and dual problem are equal, i.e.,

$$\max g(y) = \min f(x). \quad (9.3.18)$$

The minimum value is obtained at a point \hat{y} which is the solution of the m simultaneous equations

$$\hat{y}^T a_i = c_i, \quad i \in S, \quad (9.3.19)$$

where the set S is the set of integers defined previously.

Proof. By (9.3.17) it holds that

$$\max g(y) \leq \min f(x). \quad (9.3.20)$$

We shall show that \hat{y} as defined by (9.3.19) is a feasible vector. Since $Ax = b$, we may write

$$f(x) = c^T x - \hat{y}^T (Ax - b) = \hat{y}^T b + (c^T - \hat{y}^T A)x.$$

Hence by (9.3.19)

$$f(x) = y^T b + \sum_{j \notin S} (c_j - \hat{y}^T a_j) x_j. \quad (9.3.21)$$

Now $f(x)$ is expressed in terms of the nonbasic variables corresponding to the optimal solution of the primal. It then follows from the optimality criterion (see Section 9.3.3) that $c_j - \hat{y}^T a_j \geq 0$, $j \notin S$. This together with (9.3.19), shows that \hat{y} is a feasible point for the dual. Moreover, since $\hat{x}_j = 0$, $j \notin S$, then by (9.3.21) $f(\hat{x}) = \hat{y}^T b = g(\hat{y})$. This is consistent with (9.3.20) only if $\max g(y) = g(\hat{y})$. Hence $\max g(y) = \min f(x)$, and the theorem is proved. \square

A linear program initially given in the inequality form (9.3.15)–(9.3.17) can be converted to standard form by adding n slack variables. If the simplex method is used to solve this standard problem, each step involves solution of a linear system of sizes $n \times n$. If n is large it may be advantageous to switch instead to the primal problem, which is already in standard form. A simplex step for this problem involves solving linear systems of size $m \times m$, which may be much smaller size!

9.3.5 Barrier Functions and Interior Point Methods

Interior point methods for nonlinear optimization problems work by augmenting the minimization objective function with a logarithmic term

$$-\mu \log c(x)$$

for each constraint $c(x) \geq 0$. Whatever value of μ the **barrier function**, i.e. the objective function plus the logarithmic terms goes to ∞ as any constraint $c(x)$ goes to zero. This makes sequence of approximation stay strictly inside the feasible region. As the parameter μ goes to zero, the minimizer generally converges to a minimizer of the original objective function, normally on the boundary of the feasible region.

A problem with the barrier function approach is that it seems to require the solution of a sequence of unconstrained problems which become increasingly more ill-conditioned. This can be avoided by following the barrier trajectory and exploiting duality properties.

Interest in interior point methods for linear programming did not arise until later, since solving a *linear* problem by techniques from *nonlinear* optimization was not believed to be a competitive approach. The simplex method, which is fast in practice, is still much used. However, it is generally accepted that for solving really huge linear programs a so called primal-dual interior point path-following method is the most efficient. This approximates what is now called the **central path**, which is what was previously known as the barrier trajectory.

Adding a logarithmic barrier to the dual linear programming problem (9.3.15) we consider the problem

$$\begin{aligned} & \text{maximize} \quad g = y^T b + \mu \sum_{j=1}^n \log z_j, \\ & \text{subject to} \quad y^T A \leq c^T. \end{aligned} \tag{9.3.22}$$

The first order optimality conditions for (9.3.22) can be shown to be

$$\begin{aligned} XZe &= \mu e, \\ Ax &= b \\ A^T y + z &= c, \end{aligned} \tag{9.3.23}$$

where $e = (1, 1, \dots, 1)^T$, and $X = \text{diag}(x)$, $Z = \text{diag}(z)$. Let $\mu > 0$ be a parameter (which we will let tend to zero). Note that the last two sets of equations are the primal and dual feasibility equations, and in the limit $\mu \rightarrow 0$ the first set of equations expresses the complementarity condition $y^T x = 0$.

We then have a set of (partly nonlinear) equations for the unknown variables x, y, z . If we apply Newton's method the corrections will satisfy the following system of linear equations

$$\begin{aligned} Z\delta x + X\delta z &= \mu e - XZe, \\ A\delta x &= b - Ax, \\ A^T \delta y + \delta z &= c - A^T y - z. \end{aligned} \tag{9.3.24}$$

If $Z > 0$ we can solve to get

$$\begin{aligned} AZ^{-1}XA^T\delta y &= -AZ^{-1}(\mu e - XZe) + AZ^{-1}r_D + r_P, \\ \delta z &= -A^T\delta y + r_D, \\ \delta x &= Z^{-1}(\mu e - XZe) - Z^{-1}X\delta z. \end{aligned}$$

A sparse Cholesky factorization of ADA^T , where $D = Z^{-1}X$ is a positive diagonal matrix, is the main computational cost for the solution. Note that we need not worry about feasibility. The idea is to follow a **central path** $y(\mu)$ when $\mu \rightarrow 0$.

Review Questions

- 3.1.** Give the standard form for a linear programming problem. Define the terms feasible point, basic feasible point, and slack variable.
- 3.2.** State the basic theorem of linear optimization (Theorem 9.3.1). Can there be more than one optimal solution? Is every optimal solution a basic feasible point?
- 3.3.** Describe the simplex method. What does one do in the case of a degenerate feasible vector?
- 3.4.** Give the dual problem to $\min c^T x$ subject $Ax = b$, $x \geq 0$. How are the solutions to the dual and primal problems related?

Problems

- 3.1.** (a) Find the maximum of $f = x_1 + 3x_2$ subject to the constraints $x_1 \geq 0$, $x_2 \geq 0$,

$$x_1 + x_2 \leq 2, \quad x_1 + 2x_2 \leq 2.$$

First solve the problem graphically, and then use the simplex method with $x_1 = x_2 = 0$ as initial point. In the following variants, begin at the optimal vertex found in problem (a).

- (b) Find the maximum of $f = 2x_1 + 5x_2$ under the same constraints as in (a).
- (c) Find the maximum of $f = x_1 + x_2$ under the same constraints as in (a).
- (d) Find the maximum of $f = x_1 + 3x_2$ after changing the second constraint in (a) to $2x_1 + 2x_2 \leq 3$.

- 3.2.** Suppose that there is a set of programs LP that solves linear programming problems in standard form. One wants to treat the problem to minimize $f = d^T x$, $d^T = (1, 2, 3, 4, 5, 1, 1)$, where $x_i \geq 0$, $i = 1 : 7$,

$$\begin{aligned} |x_1 + x_2 + x_3 - 4| &\leq 12 \\ 3x_1 + x_2 + 5x_4 &\leq 6 \\ x_1 + x_2 + 3x_3 &\geq 3 \end{aligned}$$

$$|x_1 - x_2 + 5x_7| \geq 1$$

Give A , b , and c in the standard form formulation in this case.

- 3.3.** At each stage in the simplex method a basic variable x_l is exchanged with a certain nonbasic variable x_r . Before the change we have for each basic variable x_i a linear relation

$$x_i = b_{ir}x_r + \sum b_{ik}x_k, \quad i \in L,$$

where the sum is taken over all nonbasic variables except x_r . If the equation for $i = l$ is used to solve for x_r we get

$$x_r = \frac{1}{b_{lr}}x_l - \sum \frac{b_{lk}}{b_{lr}}x_k.$$

If this expression is substituted in the rest of the equations we obtain after the exchange a relation of the form

$$x_i = \hat{b}_{il}x_l + \sum \hat{b}_{ik}x_k, \quad i \neq l,$$

(even for $i = r$), where the sum is now taken over all the nonbasic variables except x_l . Express the coefficients in the new relation in terms of the old coefficients.

- 3.4.** (a) Put the dual problem in normal form, defined in Section 9.3.2. (Note that there is no non-negativity condition on y .)
(b) Show that the dual problem of the dual problem is the primal problem.

9.4 Least Squares Problems with Inequality Constraints

9.4.1 Classification of Problems.

This section is concerned with methods for linear least squares problems subject to different types of inequality constraints.

PROBLEM LSI.

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad l \leq Cx \leq u, \quad (9.4.1)$$

where $A \in \mathbf{R}^{m \times n}$ and $C \in \mathbf{R}^{p \times n}$. The inequalities are to be interpreted componentwise and we assume that $l \leq u$.

If c_i^T denotes the i th row of the constraint matrix C then the constraints can also be written

$$l_i \leq c_i^T x \leq u_i, \quad i = 1, \dots, p.$$

The cases where lower or upper bounds are not present can be handled by allowing the elements $l_i = -\infty$ and $u_i = \infty$.

Note that if linear equality constraints are present, then these can be eliminated using one of the methods given in Section 5.1. Both the direct elimination method

and the nullspace method can be used to reduce the problem to a lower-dimensional problem without equality constraints. An equality constraint can also be specified by setting the corresponding bounds equal, i.e., $l_i = u_i$, but this is generally not efficient.

For questions of existence, uniqueness and boundedness of solutions to Problem LSI, see, e.g., [207]. These questions have also been studied by Lötstedt [415]. It is convenient to split the solution into two mutually orthogonal components $x = x_R + x_N$, $x_R \in \mathcal{R}(A^T)$, and $x_N \in \mathcal{N}(A)$.

Theorem 9.4.1.

If the set

$$\mathcal{M} = \{l \leq Cx \leq u\}$$

is not empty, then there exists a bounded solution x^* to (9.4.1). Further Ax and $x_R = A^\dagger Ax$ are uniquely determined.

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 $l \leq Cx \leq u$ convex and polyhedral. \square

In particular when $\text{rank}(A) = n$ then $\mathcal{N}(A)$ is empty. In this case the solution x to Problem BLS is unique.

An important special case of Problem LSI is when the inequalities are simple bounds.

PROBLEM BLS.

$$\min_x \|Ax - b\|_2, \quad \text{subject to } l \leq x \leq u. \quad (9.4.2)$$

Bound-constrained least squares problems arise in many practical applications, e.g., reconstruction problems in geodesy and tomography, contact problems for mechanical systems, and the 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 it is essential that such constraints be considered separately from more general constraints in (9.4.1).

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. Then we have the following **nonnegative least squares** problem.

PROBLEM NNLS.

$$\min_x \|Ax - b\|_2 \quad \text{subject to } x \geq 0. \quad (9.4.3)$$

Nonnegativity constraints are natural for problems where the variables, by definition, can only take on nonnegative values.

Another special case of problem LSI is the least distance problem.

PROBLEM LSD.

$$\min_x \|x\|_2, \quad \text{subject to } g \leq Gx \leq h, \quad (9.4.4)$$

or more generally,

$$\min_x \|x_1\|_2, \text{ subject to } g \leq G_1 x_1 + G_2 x_2 \leq h, \quad (9.4.5)$$

where

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad x_1 \in \mathbf{R}^k, \quad x_2 \in \mathbf{R}^{n-k}.$$

If the matrix A has full column rank, Problem BLS is a strictly convex optimization problem, and then problem BLS has a unique solution for any vector b . When $\text{rank}(A) < n$ the solution to Problem BLS is not unique. This may happen, e.g., when $A \in \mathbf{R}^{m \times n}$, with $m < n$. We denote by \mathcal{M} be the set of all solutions to the problem (9.4.2), i.e.,

$$\mathcal{M} = \left\{ x \mid \min_{l \leq x \leq u} \|Ax - b\|_2 \right\}. \quad (9.4.6)$$

In order to determine a unique solution x we may look for a solution of minimum norm in \mathcal{M} satisfying

$$\min_{x \in \mathcal{M}} \|x\|_2. \quad (9.4.7)$$

In a problem without constraints the solution to (9.4.6)–(9.4.7) is the pseudoinverse solution to the least squares problem (9.4.2).

Lötstedt [416] developed a two-stage algorithm to solve problem (9.4.6)–(9.4.7). In the first stage a particular solution x to (9.4.2) is determined which is split into two mutually orthogonal components

$$x = x_R + x_N, \quad x_R \in \mathcal{R}(A^T), \quad x_N \in \mathcal{N}(A). \quad (9.4.8)$$

From Theorem 9.4.1 we have that x_R is uniquely determined, but any x_N such that x remains feasible is admissible. Since $\|x\|_2^2 = \|x_R\|_2^2 + \|x_N\|_2^2$, in the second stage we need to solve

$$\min_{l \leq x \leq u} \|x_N\|_2, \quad x_N \in \mathcal{N}(A). \quad (9.4.9)$$

Using a full orthogonal decomposition of A^T (this could be the SVD of A^T)

$$A^T = (U_1 \ U_2) \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix},$$

with S nonsingular, then $\mathcal{R}(A^T)$ is spanned by U_1 and $\mathcal{N}(A)$ by U_2 . We have $x_N = U_1(U_1^T x)$ and $x_N = U_2(U_2^T x)$. Further, for any $x_N \in \mathcal{N}(A)$ there is a z such that $x_N = U_2 z$. Since U_2 has orthonormal columns $\|x_N\|_2^2 = \|z\|_2^2$. Hence Problem (9.4.9) is equivalent to

$$\min \|z\|_2, \quad l - x_R \leq U_2 z \leq u - x_R. \quad (9.4.10)$$

This is a least distance problem for z .

9.4.2 Basic Transformations of Problem LSI.

We first remark that (9.4.1) is equivalent to the quadratic programming problem

$$\min_x (x^T B x + c^T x), \quad \text{subject to } l \leq Cx \leq u, \quad (9.4.11)$$

where $B = A^T A$, $c = -2A^T b$. The problem (9.4.11) also arises as a subproblem in general nonlinear programming algorithms. Therefore, it has been studied extensively, and many algorithms have been proposed for it. In the application to problem LSI the matrix B is positive semidefinite, and hence (9.4.11) is a convex program. In general, to use algorithms for quadratic programming for solving (9.4.1) is not a numerically safe approach, since working with the explicit cross-product matrix B should be avoided. However, methods for quadratic programming can often be adapted to work directly with A .

It is often advantageous to perform an initial transformation of A in (9.4.1) to triangular form. Using standard column pivoting (see Section 2.7.3), we compute a rank revealing QR decomposition

$$Q^T AP = R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \}_{m-r}^r, \quad (9.4.12)$$

where Q is orthogonal, P a permutation matrix, and R_{11} upper triangular and nonsingular. We assume that the numerical rank r of A is determined using some specified tolerance, as discussed in Section 2.7. The objective function in (9.4.1) then becomes

$$\|(R_{11}, R_{12}) x - c_1\|_2, \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = Q^T b,$$

since we can delete the last $(m-r)$ rows in R and c .

By a further transformation, discussed by Cline [113], problem LSI can be brought into a least distance problem. We perform orthogonal transformations from the right in (9.4.12) to obtain a complete orthogonal decomposition (see Section 2.7.5)

$$Q^T APV = \begin{pmatrix} T & 0 \\ 0 & 0 \end{pmatrix} \}_{m-r}^r,$$

where T is triangular and nonsingular. Then problem LSI (9.4.1) can be written

$$\min_y \|Ty_1 - c_1\|_2, \quad \text{subject to } l \leq Ey \leq u,$$

where

$$E = (E_1, E_2) = CPV, \quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = V^T Px,$$

E and y are conformally partitioned, and $y_1 \in \mathbf{R}^r$. We now make the further change of variables

$$z_1 = Ty_1 - c_1, \quad z_2 = y_2.$$

Substituting $y_1 = T^{-1}(z_1 + c_1)$ in the constraints we arrive at an equivalent least distance problem:

$$\min_z \|z_1\|_2, \text{ subject to } \tilde{l} \leq G_1 z_1 + G_2 z_2 \leq \tilde{u}, \quad (9.4.13)$$

where

$$\begin{aligned} G_1 &= E_1 T^{-1}, & G_2 &= E_2, \\ \tilde{l} &= l - G_1 c_1, & \tilde{u} &= u - G_1 c_1. \end{aligned}$$

We 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 (9.4.4).

The dual approach is based on the following equivalence between a least distance problem and a nonnegativity-constrained problem.

Theorem 9.4.2.

Consider the least distance problem with lower bounds

$$\min_x \|x\|_2, \quad \text{subject to } g \leq Gx. \quad (9.4.14)$$

Let $u \in \mathbf{R}^{m+1}$ be the solution to the nonnegativity constrained problem

$$\min_u \|Eu - f\|_2, \quad \text{subject to } u \geq 0, \quad (9.4.15)$$

where

$$E = \begin{pmatrix} G^T \\ g^T \end{pmatrix}, \quad f = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \}_{1:n}^n.$$

Let the residual corresponding to the solution be

$$r = (r_1, \dots, r_{n+1})^T = Eu - f,$$

and put $\sigma = \|r\|_2$. If $\sigma = 0$, then the constraints $g \leq Gx$ are inconsistent and (9.4.14) has no solution. If $\sigma \neq 0$, then the vector x defined by

$$x = (x_1, \dots, x_n)^T, \quad x_j = -r_j/r_{n+1}, \quad j = 1, \dots, n \quad (9.4.16)$$

is the unique solution to (9.4.14).

Proof. See Lawson and Hanson [399, pp. 165–167]. \square

Cline [113, 1975] describes how the modified LSD problem (9.4.5) with only upper bounds can be solved in two steps, each of which requires a solution of a problem of type NNLS, the first of these having additional linear equality constraints.

9.4.3 Active Set Algorithms for Problem LSI.

We now consider methods for solving problem LSI, which do not use a transformation into a least distance problem.

In general, methods for problems with linear inequality constraints are iterative in nature. We consider here so-called **active set algorithms**, which are based on the following observation. At the solution to (9.4.1) a certain subset of constraints $l \leq Cx \leq u$ will be active, i.e., satisfied with equality. If this subset was known a priori, the solution to the LSI problem would also be the solution to a problem with equality constraints only, for which efficient solution methods are known; see Section 5.1.

In active set algorithms 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 which are satisfied at the current approximation, but not necessarily all such constraints. In each iteration the value of the objective function is decreased and the optimum is reached in finitely many steps. A general description of active set algorithms for linear inequality-constrained optimization is given in Gill, Murray, and Wright [254, 1981, Chap. 5.2].

Any point x which satisfies all constraints in (9.4.1) is called a **feasible point**. In general, a feasible point from which to start the active set algorithm is not known. (A trivial exception is the case when all constraints are simple bounds, as in (9.4.2) and (9.4.3).) Therefore, the active set algorithm consists of two phases, where in the first phase a feasible point is determined as follows. For any point x denote by $I = I(x)$ the set of indices of constraints violated at x . Introduce an artificial objective function as the sum of all infeasibilities,

$$\phi(x) = \sum_{i \in I} \max\{(c_i^T x - u_i), (l_i - c_i^T x)\}.$$

In the first phase $\phi(x)$ is minimized subject to the constraints

$$l_i \leq c_i^T x \leq u_i, \quad i \notin I.$$

If the minimum of $\phi(x)$ is positive, then the inequalities are inconsistent, and the problem has no solution. Otherwise, when a feasible point has been found, the objective function is changed to $\|Ax - b\|_2$. Except for that, the computations in phases one and two use the same basic algorithm.

We now briefly outline an active set algorithm for solving the LSI problem. In the case when A has full column rank the algorithm described below is essentially equivalent to the algorithm given by Stoer [556, 1971].

Let x_k , the iterate at the k th step, satisfy the working set of n_k linearly independent constraints with the associated matrix C_k . We take

$$x_{k+1} = x_k + \alpha_k p_k,$$

where p_k is a search direction and α_k a nonnegative step length. The search direction is constructed so that the working set of constraints remains satisfied for all values of α_k . This will be the case if $C_k p_k = 0$. In order to satisfy this condition we compute a decomposition

$$C_k Q_k = (0, T_k), \quad T_k \in \mathbf{R}^{n_k \times n_k}, \quad (9.4.17)$$

where T_k is triangular and nonsingular, and Q_k is a product of orthogonal transformations. (This is essentially the QR decomposition of C_k^T .) If we partition Q_k conformally,

$$Q_k = \left(\underbrace{Z_k}_{n-n_k}, \underbrace{Y_k}_{n_k} \right), \quad (9.4.18)$$

then the $n - n_k$ columns of Z_k form a basis for the nullspace of C_k . Hence the condition $C_k p_k = 0$ is satisfied if we take

$$p_k = Z_k q_k, \quad q_k \in \mathbf{R}^{n-n_k}. \quad (9.4.19)$$

We now determine q_k so that $x_k + Z_k q_k$ minimizes the objective function, i.e., in phase two q_k solves the unconstrained least squares problem

$$\min_{q_k} \|AZ_k q_k - r_k\|_2, \quad r_k = b - Ax_k. \quad (9.4.20)$$

To simplify the discussion we assume in the following that the matrix AZ_k is of full rank so that (9.4.20) has a unique solution. To compute this solution we need the QR decomposition of the matrix AZ_k . This is obtained from the QR decomposition of the matrix AQ_k , where

$$P_k^T A Q_k = P_k^T (A Z_k, A Y_k) = \begin{pmatrix} R_k & S_k \\ 0 & U_k \\ 0 & 0 \end{pmatrix} \begin{matrix} \} n - n_k \\ \} n_k \end{matrix}. \quad (9.4.21)$$

The advantage of computing this larger decomposition is that then the orthogonal matrix P_k need not be saved and can be discarded after being applied also to the residual vector r_k . The solution q_k to (9.4.20) can now be computed from the triangular system

$$R_k q_k = c_k, \quad P_k^T r_k = \begin{pmatrix} c_k \\ d_k \end{pmatrix} \begin{matrix} \} n - n_k \\ \} n_k \end{matrix}.$$

The next approximate solution is then taken to be $x_{k+1} = x_k + \alpha_k p_k$, where α_k is a step length to be chosen.

We now determine $\bar{\alpha}$, the maximum nonnegative step length along p_k for which x_{k+1} remains feasible with respect to the constraints not in the working set. If $\bar{\alpha} \leq 1$ we take $\alpha_k = \bar{\alpha}$, and then add the constraints which are hit to the working set for the next iteration.

If $\bar{\alpha} > 1$ we take $\alpha_k = 1$. In this case x_{k+1} will minimize the objective function when the constraints in the working set are treated as equalities, and the orthogonal projection of the gradient onto the subspace of feasible directions will be zero:

$$Z_k^T g_{k+1} = 0, \quad g_{k+1} = -A^T r_{k+1}.$$

In this case we check the optimality of x_{k+1} by computing Lagrange multipliers for the constraints in the working set. At x_{k+1} these are defined by the equation

$$C_k^T \lambda = g_{k+1} = -A^T r_{k+1}. \quad (9.4.22)$$

The residual vector to the new unconstrained problem r_{k+1} satisfies

$$P_k^T r_{k+1} = \begin{pmatrix} 0 \\ d_k \end{pmatrix}.$$

Hence, multiplying (9.4.22) by Q_k^T and using (9.4.17) we obtain

$$Q_k^T C_k^T \lambda = \begin{pmatrix} 0 \\ T_k^T \end{pmatrix} \lambda = -Q_k^T A^T P_k \begin{pmatrix} 0 \\ d_k \end{pmatrix},$$

so from (9.4.21)

$$T_k^T \lambda = -(U_k^T, 0) d_k.$$

The Lagrange multiplier λ_i for the constraint $l_i \leq c_i^T x \leq u_i$ in the working set is said to be optimal if $\lambda_i \leq 0$ at an upper bound and $\lambda_i \geq 0$ at a lower bound. If all multipliers are optimal then we have found an optimal point and are finished. If a multiplier is not optimal then the objective function can be decreased by deleting the corresponding constraint from the working set. If more than one multiplier is not optimal, then it is usual to delete that constraint whose multiplier deviates most from optimality.

At each iteration step the working set of constraints is changed, which leads to a change in the matrix C_k . If a constraint is dropped, a row in C_k is deleted; if a constraint is added, a new row in C_k is introduced. An important feature of an active set algorithm is the efficient solution of the sequence of unconstrained problems (9.4.20). Using techniques described in Section 3.2 methods can be developed to update the matrix decompositions (9.4.17) and (9.4.20). In (9.4.17) the matrix Q_k is modified by a sequence of orthogonal transformations from the right. These transformations are then applied to Q_k in (9.4.21) and this decomposition, together with the vector $P_k^T r_{k+1}$, is similarly updated. Since these updatings are quite intricate, they will not be described in detail here.

For the case when A has full rank the problem LSI always has a unique solution. If A is rank deficient there will still be a unique solution if all active constraints at the solution have nonzero Lagrange multipliers. Otherwise there is 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 $\min \|x\|_2$, $x \in M$. This is a least distance problem.

In the rank deficient case it can happen that the matrix AZ_k in (9.4.20) is rank deficient, and hence R_k is singular. Note that if some R_k is nonsingular it can become singular during later iterations only when a constraint is deleted from the working set, in which case only its last diagonal element can become zero. This simplifies the treatment of the rank deficient case. To make the initial R_k nonsingular one can add artificial constraints to ensure that the matrix AZ_k has full rank.

A possible further complication is that the working set of constraints can become linearly dependent. This can cause possible cycling in the algorithm, so that its convergence cannot be ensured. A simple remedy that is often used is to enlarge the feasible region of the offending constraint by a small quantity; see also Gill, Murray, and Wright [254, 1981, Chap. 5.8.2].

9.4.4 An Active Set Algorithm for BLS.

As remarked earlier the problem LSI simplifies considerably when the only constraints are simple bounds. This problem is important in its own right and also serves as a good illustration of the general algorithm. Hence we now consider the algorithm for problem BLS in more detail.

We first note that feasibility of the bounds is resolved by simply checking whether $l_i \leq u_i, i = 1, \dots, p$. Further, the specification of the working set is equivalent to a partitioning of x into free and fixed variables. During an iteration the fixed variables will not change and can be effectively removed from the problem.

We divide the index set of x according to

$$\{1, 2, \dots, n\} = \mathcal{F} \cup \mathcal{B},$$

where $i \in \mathcal{F}$ if x_i is a free variable and $i \in \mathcal{B}$ if x_i is fixed at its lower or upper bound. The matrix C_k will now consist of the rows $e_i, i \in \mathcal{B}$, of the unit matrix I_n . We let $C_k = E_{\mathcal{B}}^T$, and if $E_{\mathcal{F}}$ is similarly defined we can write

$$Q_k = (E_{\mathcal{F}}, E_{\mathcal{B}}), \quad T_k = I_{n_k}.$$

This shows that Q_k is simply a permutation matrix, and the product

$$AQ_k = (AE_{\mathcal{F}}, AE_{\mathcal{B}}) = (A_{\mathcal{F}}, A_{\mathcal{B}})$$

corresponds to a permutation of the columns of A . Assume now that the bound corresponding to x_q is to be dropped. This can be achieved by

$$AQ_{k+1} = AQ_k P_R(k, q),$$

where $P_R(k, q)$, $q > k + 1$, is a permutation matrix which performs a right circular shift in which the columns are permuted:

$$k + 1, \dots, q - 1, q \Rightarrow q, k + 1, \dots, q - 1.$$

Similarly, if the bound corresponding to x_q becomes active it can be added to the working set by

$$AQ_{k+1} = AQ_k P_L(q, k),$$

where $P_L(q, k)$, $q < k - 1$, is a permutation matrix, which performs a left circular shift in which the columns are permuted:

$$q, q + 1, \dots, k \Rightarrow q + 1, \dots, k, q.$$

Subroutines for updating the QR decomposition (QRD) after right or left circular shifts are included in LINPACK and are described in Dongarra et al. [164, Chap. 10].

For problem BLS the equation (9.4.22) for the Lagrange multipliers simplifies to

$$\lambda = E_{\mathcal{B}}^T g_{k+1} = -E_{\mathcal{B}}^T A^T r_{k+1} = -(U_k^T, 0) d_k.$$

Hence the Lagrange multipliers are simply equal to the corresponding components of the gradient vector $-A^T r_{k+1}$.

Several implementations of varying generality of active set methods for problem BLS have been developed. Lawson and Hanson [399] give a Fortran implementation of an algorithm for problem NNLS. They also give a Fortran subroutine based on this algorithm for problem LSD with lower bounds only.

The active set algorithms usually restrict the change in dimension of the working set by dropping or adding only one constraint at each iteration. For large scale problems this can force many iterations to be taken when, e.g., many variables have to leave their bound. Hence an active set algorithm can be slow to converge when the set of active constraints cannot be guessed well by the user.

Review Questions

4.1. To be written.

Problems

4.1. To be written.

Notes and References

Section 9.1

The Newton–Kantorovich theorem is due to Kantorovich, see [366, 367]. The numerical solution of nonlinear equations by the methods of Newton, Brown and Brent is discussed by Moré and Cosnard [444]. An evaluation of numerical software that solves systems of nonlinear equations is given by Hiebert [322]. Here eight different available Fortran codes are compared on a set of test problems. Of these one uses a quasi-Newton two Brown’s method, one Brent’s method, and the remaining four Powell’s hybrid method, see Powell [491]. A standard treatment of continuation methods is Allgower and Georg [6].

Different aspects of automatic differentiation are discussed by Rall [495], Corliss et al. [119], and Griewank and Walther [282]. Derivative free methods in optimization are discussed in Conn, Scheinberg, and Vicente [117]. Quasi-Newton methods are discussed by Eriksson [193].

Section 9.2

A standard textbook on numerical methods for unconstrained optimization, nonlinear systems, and nonlinear least squares is Dennis and Schnabel [156]. A review of developments and applications of the variable projection approach for separable nonlinear least squares problems is given by Golub and Pereyra [272]. Methods for solving constrained nonlinear least squares problems are discussed by Gulliksson, Söderkvist, and Wedin [293].

Trust region methods are discussed by Conn, Gould and Toint [116]. Golub and LeVeque [267] extended 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 [373]. Ruhe and Wedin [508] analyze several different algorithms for a more general class of separable problems.

Algorithms for geometric fitting of circles and ellipses are described in Gander, Golub, and Strelbel [225].

Section 9.3

A classical reference on linear optimization is Danzig [135]. For nonlinear optimization the book by Luenberger [419] is a good introduction. Excellent textbooks on optimization are Gill, Murray and Wright [254], and Fletcher [207]. A very useful reference on software packages for large scale optimization is Moré and Wright [445]. See also Nocedal and Wright [452].

For difficult optimization problems where the gradient is not available direct search methods may be the only option. A review of this class of methods is given in [382]. Two recent books on interior point methods for linear programming are Vanderbei [591] and Wright [619]. Applications, theory and algorithms for linear and nonlinear optimization are treated in Grive, Nash, and Sofer [284].

Interior point methods for nonlinear optimization problems were introduced by Fiacco and McCormick [201]. In 1984 Karmarkar [369] published a projective method, which passes through the interior of the polygon, for solving linear programming problems. It was soon realized, see Margaret Wright [618], that Karmarkar's method was closely related to logarithmic barrier methods.

Section 9.3

A robust and general set of Fortran subroutines for problem LSI and convex quadratic programming is given by Gill et al. [250, 1986]. The method is a two-phase active set method. It also allows a linear term in the objective function and handles a mixture of bounds and general linear constraints.

Chapter 10

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

10.1 Basic Properties

10.1.1 Introduction

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

$$(A - \lambda I)x = 0, \quad x \neq 0, \tag{10.1.1}$$

and are called the **eigenvectors** of A . The scalar λ in (10.1.1) are the corresponding **eigenvalues** of A . Thus, λ is an eigenvalue of A only if $A - \lambda I$ is a singular matrix and can be found by solving the characteristic equation

$$p_n(\lambda) = \det(\lambda I - A) = c_0 + c_1\lambda + \cdots + c_{n-1}\lambda^{n-1} + \lambda^n, \tag{10.1.2}$$

where $p_n(\lambda)$ is the characteristic polynomial of degree n . When an eigenvalue λ is known, the corresponding eigenvectors can be obtained by solving the homogeneous linear system (10.1.1). Clearly, an eigenvector x is only determined up to a multiplicative constant.

Eigenvalues and eigenvectors are a standard tool in the mathematical sciences and in scientific computing. Eigenvalues give information about the behavior of evolving systems governed by a matrix or operator. The problem of computing eigenvalues and eigenvectors of a matrix occurs in many settings in physics and engineering. Eigenvalues are useful in analyzing resonance, instability, and rates of growth or decay with applications to, e.g., vibrating systems, airplane wings, ships,

buildings, bridges and molecules. Eigenvalues can be used to determine whether a building or a bridge may collapse or whether the flow over a wing is laminar or turbulent. Eigenvalue decompositions also play an important part in the analysis of many numerical methods. Singular values and vectors are closely related to an eigenproblem for a special symmetric matrix.

The matrix eigenproblem is inherently a nonlinear problem, and this leads to several challenging computational problems. Clearly any general method for determining eigenvalues must involve an iteration, since a fundamental theorem by Abel says that no finite algorithm exists for the solution of an algebraic equation of degree greater than 4. In the early days of computing the classical textbook approach to solve an eigenvalue problem was to first compute the coefficients c_0, \dots, c_{n-1} of the characteristic polynomial and then determine the zeros of $p_n(\lambda)$. The effort at that time was on developing reliable solvers for polynomial equations. However, a crucial discovery, made in the early 1960th, was that the zeros of polynomials could be extremely sensitive to small perturbations in the coefficients. Therefore, the explicit computation of the coefficients of $p(\lambda)$ should in general be avoided.

In Section 10.1 we give a brief account of basic facts of the matrix eigenvalue problem and the theory of canonical forms. Section 10.2 is devoted to the methods for localization of eigenvalues and perturbation results. There are many aspects to consider when solving a particular problem. The matrix may have some special property such as being Hermitian or unitary. It may have some special structure, such as band structure, that can be taken advantage of. One also has to consider if all or only part of the eigenvalues are wanted and whether the corresponding eigenvectors are also required.

The power method and some of its modifications are treated in Section 10.3. The QR algorithm, which is the method of choice for computing eigenvalues and eigenvectors of small to medium size matrices, is the topic of Section 10.4. The Hermitian case and a QR algorithm for computing singular values and singular vectors are treated in Section 10.5. Some alternative algorithms for Hermitian eigenvalue problems are surveyed in Section 10.6. These methods have advantages for parallel implementation and are potentially very accurate. Section 10.8 deals with matrix series and matrix functions such as the square root, exponential and logarithm of a matrix. Finally, in Section 10.7 some generalized and structured eigenvalue problems are studied.

10.1.2 Basic Theory

It follows from (10.1.1) that λ is an eigenvalue of A if and only if the system $(A - \lambda I)x = 0$ has a nontrivial solution $x \neq 0$, or equivalently if and only if the matrix $A - \lambda I$ is singular. Hence, the eigenvalues satisfy the **characteristic equation**

$$p_n(\lambda) = \det(\lambda I - A) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0 \quad (10.1.3)$$

The polynomial $p_n(\lambda) = \det(A - \lambda I)$ is called the **characteristic polynomial** of the matrix A . Expanding the determinant in (10.1.3) it follows that $p(\lambda)$ has the form

$$p_n(\lambda) = (\lambda - a_{11})(\lambda - a_{22}) \cdots (\lambda - a_{nn}) + q(\lambda), \quad (10.1.4)$$

$$= c_0 + c_1\lambda + \cdots + c_{n-1}\lambda^{n-1} + \lambda^n, \quad (10.1.5)$$

where $q(\lambda)$ has degree at most $n - 2$. Thus, 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, and we can write

$$p(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_n).$$

The set $\{\lambda_1, \dots, \lambda_n\}$ of all eigenvalues of A is called the **spectrum**⁴² of A and is denoted by $\lambda(A)$. Using the relation between roots and coefficients of an algebraic equation, we obtain the relations

$$\lambda_1\lambda_2 \cdots \lambda_n = \det(A), \quad (10.1.6)$$

$$\lambda_1 + \lambda_2 + \cdots + \lambda_n = \text{trace}(A), \quad (10.1.7)$$

where $\text{trace}(A) = a_{11} + a_{22} + \cdots + a_{nn}$. The relation (10.1.7) is useful for checking the accuracy of a computed eigenvalues.

For any nonzero vector $v_0 = v \in \mathbf{C}^n$, define a sequence of vectors by

$$v_k = Av_{k-1} = A^k v, \quad k = 1, 2, \dots \quad (10.1.8)$$

Let v_m be the first of these vectors that can be expressed as a linear combination of the preceding ones

$$c_0v_0 + c_1v_1 + \cdots + v_m = (c_0I + c_1A + \cdots + A^m)v = p(A)v = 0.$$

where p is a polynomial of degree m . Since there are at most n linearly independent vectors in \mathbf{C}^n , we must have $m \leq n$. The polynomial p is the polynomial of minimal degree that annihilates v and therefore it is called the minimal polynomial of v . The degree m of p is the **grade** of v with respect to A . Of all vectors $v \in \mathbf{C}^n$ there is at least one for which the degree is maximal, since for any vector $m \leq n$. If v is such a vector and q its minimal polynomial, then it can be shown that $q(A)x = 0$ for any vector $x \in \mathbf{C}^n$, and hence

$$q(A) = \gamma_0I + \gamma_1A + \cdots + \gamma_{s-1}A^{s-1} + A^s = 0.$$

This polynomial q is the **minimal polynomial** of A ; see Section 10.1.3.

⁴²From Latin verb *specere* meaning “to look”.

Theorem 10.1.1.

Let $A \in \mathbf{C}^{n \times n}$. Then

$$\lambda(A^T) = \lambda(A), \quad \lambda(A^H) = \bar{\lambda}(A).$$

Proof. Since $\det(\lambda I - A^T)^T = \det(\lambda I - A)^T = \det(\lambda I - A)$ it follows that A^T and A have the same characteristic polynomial and thus the same set of eigenvalues. For the second part note that $\det(\bar{\lambda}I - A^H) = \det(\lambda I - A)^H$ is zero if and only if $\det(\lambda I - A)$ is zero. \square

By the above theorem, if λ is an eigenvalue of A then $\bar{\lambda}$ is an eigenvalue of A^H , i.e., $A^H y = \bar{\lambda} y$ for some vector $y \neq 0$, or equivalently

$$y^H A = \lambda y^H, \quad y \neq 0. \quad (10.1.9)$$

Here y is called a **left eigenvector** of A , and consequently if $Ax = \lambda x$, x is also called a **right eigenvector** of A . The eigenvectors are only determined up to a constant multiplicative factor. Usually they are 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.

Theorem 10.1.2.

Let λ_i and λ_j be two distinct eigenvalues of $A \in \mathbf{C}^{n \times n}$, 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 with x_j from the right and the second with 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 . Assume that X is nonsingular, and set $Y^H = X^{-1}$. Then $Y^H X = I$. and $Y^H A = \Lambda y^H$, which shows that the rows of Y^H are left eigenvectors $y_i^H A = \lambda_i y_i^H$, $i = 1 : n$. We can also write

$$A = X\Lambda Y^H = \sum_{i=1}^n \lambda_i P_i, \quad P_i = x_i y_i^H. \quad (10.1.10)$$

which is the **spectral decomposition** of A . Here, P_i is the **spectral projector** corresponding to λ_i . It is called so because $(x_i y_i^H)z$ projects z onto the range of x_i along y_i . Note that $P_i^2 = x_i(y_i^H x_i) y_i^H = P_i$ because of the normalization $y_i^H x_i = 1$.

If all the eigenvalues λ_{i_1} are simple then the null space of $A - \lambda_{i_1}I$ has dimension $(n - 1)$ for all $i = 1 : n$. Then the decomposition (10.1.10) is essentially unique. If there are eigenvalues of multiplicity $p > 1$, then the corresponding eigenvectors can be chosen as any basis for the set of solutions to $(A - \lambda_i I)x = 0$.

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

Definition 10.1.3.

The **spectral radius** of a matrix $A \in \mathbf{C}^{n \times n}$ is the maximal absolute value of the eigenvalues of A

$$\rho(A) = \max_i |\lambda_i|. \quad (10.1.11)$$

The **spectral abscissa** is the maximal real part of the eigenvalues of A

$$\alpha(A) = \max_i \Re(\lambda_i). \quad (10.1.12)$$

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

Lemma 10.1.4.

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

$$\rho(A) \leq \|A\|, \quad (10.1.13)$$

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 $\|x\|$ the result follows. \square

Two matrices $A \in \mathbf{C}^{n \times n}$ and $\tilde{A} \in \mathbf{C}^{n \times n}$ are said to be **similar** if there is a square nonsingular matrix $S \in \mathbf{C}^{n \times n}$ such that

$$\tilde{A} = S^{-1}AS, \quad (10.1.14)$$

The transformation (10.1.14) is called a **similarity transformation** of A . Similarity of matrices is an equivalence transformation, i.e., if A is similar to B and B is similar to C then A is similar to C .

Theorem 10.1.5.

If A and B are similar, then A and B have the same characteristic polynomial, and hence the same eigenvalues. Further, if $B = S^{-1}AS$ and y is an eigenvector of B corresponding to λ then Sy is an eigenvector of A corresponding to λ .

Proof. We have

$$\begin{aligned} \det(\lambda I - B) &= \det(\lambda I - S^{-1}AS) = \det(S^{-1}(\lambda I - A)S) \\ &= \det(S^{-1}) \det(\lambda I - A) \det(S) = \det(\lambda I - A). \end{aligned}$$

Further, from $AS = SB$ it follows that $ASy = SBy = \lambda Sy$. \square

Similarity transformations corresponds to a change in the coordinate system. Similar matrices represent the same linear transformation in different coordinate systems.

Let the matrix $A \in \mathbf{R}^{n \times n}$ have the factorization $A = BC$, where $B \in \mathbf{R}^{n \times n}$ is invertible, and set $\tilde{A} = CB$. Then \tilde{A} is similar to A since

$$A = BC = B^{-1}(BC)B = CB = \tilde{A}. \quad (10.1.15)$$

A slightly more general result is the following:

Lemma 10.1.6.

Let $A = BC \in \mathbf{C}^{n \times n}$, where $B \in \mathbf{C}^{n \times p}$ and $C \in \mathbf{C}^{p \times m}$, and set $\tilde{A} = CB \in \mathbf{R}^{p \times p}$. Then the nonzero eigenvalues of A and \tilde{A} are the same.

Proof. The result follows from the identity

$$S^{-1} \begin{pmatrix} BC & 0 \\ C & 0 \end{pmatrix} S = \begin{pmatrix} 0 & 0 \\ C & CB \end{pmatrix}, \quad S = \begin{pmatrix} I & B \\ 0 & I \end{pmatrix}.$$

This shows that the two block triangular matrices are similar and therefore have the same eigenvalues. \square

Many important algorithms for computing eigenvalues and eigenvectors use a sequence of similarity transformations

$$A_k = P_k^{-1} A_{k-1} P_k, \quad k = 1, 2, \dots,$$

where $A_0 = A$, to transform the matrix A into a matrix of simpler form. The matrix A_k is similar to A and the eigenvectors x of A and y of A_k are related by $x = P_1 P_2 \cdots P_k y$. for example, if the matrix A can be transformed to triangular form, then its eigenvalues equal the diagonal elements.

The following theorem can be proved by induction.

Theorem 10.1.7.

Assume that the matrix A can be reduced by a similarity transformation to block upper triangular form

$$\tilde{A} = S^{-1} AS = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ 0 & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A_{NN} \end{pmatrix}, \quad (10.1.16)$$

where each diagonal block A_{ii} is square. Then

$$\lambda(A) = \bigcup_{i=1}^N \lambda(A_{ii}),$$

where $\lambda(A)$ denotes the spectrum of A . In particular, the eigenvalues of a triangular matrix are its diagonal elements.

The equation $Ax = \lambda x$ shows that the subspace spanned by an eigenvalue is invariant under multiplication with A . This concept can be generalized to subspaces. The importance of this concept is that eigenspaces can be used to bring together eigenvectors which are not well defined individually. They also play a central role in eigenproblems for large matrices, where only a subset of the eigenvalues and eigenvectors can be computed.

Definition 10.1.8.

Let $A \in \mathbf{C}^{n \times n}$ and let \mathcal{X} be a subspace of \mathbf{C}^n . Then \mathcal{X} is an **invariant subspace** or **eigenspace** of A if

$$A\mathcal{X} \equiv \{Ax \mid x \in \mathcal{X}\} \subseteq \mathcal{X}.$$

Clearly, any set of right eigenvectors spans an invariant subspace. Let $X \in \mathbf{C}^{n \times k}$, be a basis for an eigenspace \mathcal{X} of dimension k of A . Then

$$AX = XB, \quad B = Y^H AX \in \mathbf{C}^{k \times k} \quad (10.1.17)$$

where $Y^H X = I_k$, i.e., Y^H is the unique left inverse of X . We say that the matrix B is a representation of A on the eigenspace \mathcal{X} .

Conversely, if (10.1.17) holds, then $AXz = XBz \in \mathcal{R}(X)$ for any $z \in \mathbf{C}^k$. Hence, the columns of X is a basis for a right invariant subspace $\mathcal{R}(X)$. If $Bz = \lambda z$ then

$$AXz = XBz = \lambda Xz,$$

and so any eigenvalue λ of B is also an eigenvalue of A and the vector Xz an eigenvector of A .

The basis in \mathcal{X} can be chosen arbitrarily. Let X be a basis and $S \in \mathbf{C}^{k \times k}$ nonsingular. Then $\tilde{X} = XS$ is a basis, $\tilde{Y}^H = S^{-1}Y^H$ and

$$\tilde{B} = S^{-1}Y^H A XS = S^{-1}BS$$

is a similarity transformation of B . In particular, if $X = Q_X R$ is the QR factorization of X , then Q_X is a unitary basis for \mathcal{X} and Q_X^H is its left inverse.

Similarly, if

$$Y^H A = BY^H, \quad Y \in \mathbf{C}^{n \times k},$$

$\text{rank}(Y) = k \leq n$, then $\mathcal{R}(Y)$ is a left invariant subspace and $A^H \mathcal{Y} \subseteq \mathcal{Y}$. If v is a left eigenvectors of B , then $v^H B = \lambda v^H$ it follows that

$$v^H Y^H A = v^H B Y^H = \lambda v^H Y^H.$$

Hence λ is an eigenvalue of A and Yv is a left eigenvector.

Suppose we change the basis in \mathcal{X} by setting $\tilde{X} = XS$, where S is nonsingular. It is left as an exercise to verify that then Y and B are transformed according to $\tilde{Y}^H = S^{-1}Y^H$ and $\tilde{B} = S^{-1}BS$,

10.1.3 The Jordan Canonical Form

If the eigenvectors x_1, x_2, \dots, x_n of a matrix $A \in \mathbf{C}^{n \times n}$ are linearly independent, then the matrix of eigenvectors $X = (x_1, x_2, \dots, x_n)$ is nonsingular and it holds that

$$Y^H A X = \Lambda; \quad Y^H = X^{-1}, \quad (10.1.18)$$

that is, a similarity transformation (10.1.18) transforms A to diagonal form. In this case the matrix is said to be **diagonalizable** and it has the spectral decomposition

$$A = X \Lambda Y^H = \sum_{i=1}^n \lambda_i x_i y_i^H. \quad (10.1.19)$$

The following theorem says that a matrix is diagonalizable if all its eigenvalues are simple.

Theorem 10.1.9.

Let x_1, \dots, x_k be eigenvectors of $A \in \mathbf{C}^{n \times n}$ 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.

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, we must have $p = k$ linearly independent vectors. \square

A matrix A may not have a full set of n linearly independent eigenvectors. Let $\lambda_1, \dots, \lambda_k$ be the distinct zeros of $p(\lambda)$ and let σ_i be the multiplicity of λ_i , $i = 1 : k$. The integer σ_i is called the **algebraic multiplicity** of the eigenvalue λ_i and

$$\sigma_1 + \sigma_2 + \dots + \sigma_k = n.$$

To every distinct eigenvalue corresponds at least one eigenvector. All the eigenvectors corresponding to the eigenvalue λ_i form a linear subspace $L(\lambda_i)$ of \mathbf{C}^n of dimension

$$\rho_i = n - \text{rank}(A - \lambda_i I). \quad (10.1.20)$$

The integer ρ_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 in general uniquely determined.

Theorem 10.1.10.

The geometric and algebraic multiplicity of an eigenvalue satisfy the inequality

$$\rho(\lambda) \leq \sigma(\lambda). \quad (10.1.21)$$

Proof. Let $\bar{\lambda}$ be an eigenvalue with geometric multiplicity $\rho = \rho(\bar{\lambda})$ and let x_1, \dots, x_ρ be linearly independent eigenvectors associated with $\bar{\lambda}$. If we put $X_1 = (x_1, \dots, x_\rho)$ then we have $AX_1 = \bar{\lambda}X_1$. We now let $X_2 = (x_{\rho+1}, \dots, x_n)$ consist of $n - \rho$ more vectors such that the matrix $X = (X_1, X_2)$ is nonsingular. Then it follows that the matrix $X^{-1}AX$ must have the form

$$X^{-1}AX = \begin{pmatrix} \bar{\lambda}I & B \\ 0 & C \end{pmatrix}$$

and hence the characteristic polynomial of A , or $X^{-1}AX$ is

$$p(\lambda) = (\lambda - \bar{\lambda})^\rho \det(\lambda I - C).$$

Thus, the algebraic multiplicity of $\bar{\lambda}$ is at least equal to ρ . \square

If $\rho(\lambda) = \sigma(\lambda)$ then the eigenvalue λ is said to be **semisimple**; if $\rho(\lambda) < \sigma(\lambda)$ then λ is **defective**. A matrix with at least one defective eigenvalue is **defective**, otherwise it is **nondefective**. The eigenvectors of a nondefective matrix A span the space \mathbf{C}^n and A is said to have a complete set of eigenvectors. A matrix is nondefective if and only if it is semisimple. We now state without proof the following fundamental **Jordan canonical form**.⁴³

Theorem 10.1.11 (The Jordan Canonical Form).

If $A \in \mathbf{C}^{n \times n}$, then there is a nonsingular matrix $X \in \mathbf{C}^{n \times n}$, such that

$$X^{-1}AX = J = \text{diag}(J_{m_1}(\lambda_1), \dots, J_{m_t}(\lambda_t)), \quad (10.1.22)$$

where $m_i \geq 1$, $i = 1 : t$, and

$$J_{m_i}(\lambda_i) = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix} = \lambda_i I + S \in \mathbf{C}^{m_i \times m_i}, \quad (10.1.23)$$

are **Jordan blocks**. The numbers m_1, \dots, m_t are unique and $\sum_{i=1}^t m_i = n$. The form (10.1.22) is called the **Jordan canonical form** of A and is unique up to the ordering of the Jordan blocks.

⁴³Marie 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.

A full proof of this fundamental theorem would be quite long and is omitted. It starts with a block diagonal decomposition of A given in Theorem 10.1.17. The more difficult part is the reduction of the individual blocks to Jordan block form. For this part we refer to Fletcher and Sorensen [208, 1983].

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} , which satisfy a chain

$$Ax_1 = \lambda_1 x_1, \quad Ax_{i+1} = \lambda_1 x_{i+1} + x_i, \quad i = 1 : m_1 - 1.$$

are called **principal vectors** of A .

Note that the same eigenvalue may appear in several different Jordan blocks. A matrix for which this occurs is called **derogatory**. The Jordan canonical form has the advantage that it displays all eigenvalues and eigenvectors of A explicitly.

Example 10.1.1.

Consider a system of linear differential equations with constant coefficients defined by

$$\frac{dy}{dt} = Ay, \quad y \in \mathbf{R}^n, \quad (10.1.24)$$

where $A = XJX^{-1}$ is the Jordan canonical form of A . Then after the linear transformation $z = Xy$ the system becomes

$$\frac{dz}{dt} = XAX^{-1}z = Jz,$$

If A is not defective, then J is strictly diagonal and the system decouples into n scalar equations

$$\frac{dz_i}{dt} = \lambda_i z_i, \quad i = 1 : n,$$

with solutions $z_i = z_i^{(0)} e^{\lambda_i t}$. When A is defective, there is not a complete decoupling, but the components associated with each separate Jordan block are decoupled from each other. The general solution can also be expressed using the exponential of a matrix; see Section 10.8.3.

A serious disadvantage is that the Jordan canonical form is not in general a continuous function of the elements of A . For this reason the Jordan canonical form of a nondiagonalizable matrix may be very difficult to determine numerically.

Example 10.1.2.

Consider a perturbed Jordan block of the form

$$J_m(\lambda, \epsilon) = \begin{pmatrix} \lambda & 1 & & \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & \lambda \end{pmatrix} \in \mathbf{C}^{m \times m}.$$

The matrix $J_m(\lambda, 0)$ has an eigenvalue equal to λ of multiplicity m , and is in Jordan canonical form. For any $\epsilon > 0$ the matrix $J_m(\lambda, \epsilon)$ has m distinct eigenvalues μ_i , $i = 1 : m$, which are the roots of the equation $(\lambda - \mu)^m - (-1)^m \epsilon = 0$. Hence, $J_m(\lambda, \epsilon)$ is diagonalizable for any $\epsilon \neq 0$, and its eigenvalues λ_i satisfy $|\lambda_i - \lambda| = |\epsilon|^{1/m}$. For example, if $m = 10$ and $\epsilon = 10^{-10}$, then the perturbation is of size 0.1. Note that since the trace of the perturbed matrix does not change, the mean value of the eigenvalues is not perturbed.

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 we have $q(J_m(\lambda)) = N^j = 0$ for $j \geq m$. The minimal polynomial of a matrix A with the *distinct* eigenvalues $\lambda_1, \dots, \lambda_k$ then has the form

$$q(z) = (z - \lambda_1)^{m_1} (z - \lambda_2)^{m_2} \cdots (z - \lambda_k)^{m_k}, \quad (10.1.25)$$

where m_j is the highest dimension of any Jordan box corresponding to the eigenvalue λ_j , $j = 1 : k$.

As a corollary we obtain the **Cayley–Hamilton theorem**, which states that if $p_A(z) = \det(zI - A)$ is the characteristic polynomial of a matrix A then $p_A(A) = 0$, i.e., the matrix satisfies its own characteristic polynomial.

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.

We end with an approximation theorem due to Bellman, which sometimes makes it possible to avoid the complication of the Jordan canonical form.

Theorem 10.1.12.

Let $A \in \mathbf{C}^{n \times n}$ be a given matrix. Then for any $\epsilon > 0$ there exists a matrix B with $\|A - B\|_2 \leq \epsilon$, such that B has n distinct eigenvalues. Hence, the class of diagonalizable matrices is dense in $\mathbf{C}^{n \times n}$.

Proof. Let $X^{-1}AX = J$ be the Jordan canonical form of A . Then, by a slight extension of Example 10.1.2 it follows that for any $\delta > 0$ there is a matrix $J(\delta)$ with distinct eigenvalues such that $\|J - J(\delta)\|_2 = \delta$. (Show this!) If we take $B = XJ(\delta)X^{-1}$, then

$$\|A - B\|_2 \leq \epsilon, \quad \epsilon = \delta \|X\|_2 \|X^{-1}\|_2.$$

This proves the theorem. \square

10.1.4 The Schur Decomposition

The Jordan canonical form of a matrix reveals its eigenvalues and gives information about the eigenvectors. It is a useful theoretical tool and can be used for extending

analytical functions to matrix arguments; see Section 10.8. It is less useful for computational purposes, since it can be very sensitive to perturbations to A . A more computationally useful decomposition is the **Schur decomposition**; Schur [525, 1909].

Theorem 10.1.13 (The Schur Decomposition).

Given $A \in \mathbf{C}^{n \times n}$ there exists a unitary matrix $U \in \mathbf{C}^{n \times n}$ such that

$$U^H AU = T = D + N, \quad (10.1.26)$$

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 the matrix 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 $A \in \mathbf{C}^{n \times n}$.

Let λ be an arbitrary eigenvalue of A . Then, $Ax = \lambda x$, for some $x \neq 0$ and we let $u_1 = x/\|x\|_2$. Then we can always find $U_2 \in \mathbf{C}^{n \times n-1}$ 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

$$U^H AU = \begin{pmatrix} u_1^H \\ U_2^H \end{pmatrix} AU = \begin{pmatrix} \lambda u_1^H u_1 & u_1^H AU_2 \\ \lambda U_2^H u_1 & U_2^H AU_2 \end{pmatrix} = \begin{pmatrix} \lambda & w^H \\ 0 & B \end{pmatrix}.$$

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

$$\bar{U}^H A \bar{U} = T = \begin{pmatrix} \lambda & w^H \tilde{U} \\ 0 & \tilde{T} \end{pmatrix}, \quad \bar{U} = U \begin{pmatrix} 1 & 0 \\ 0 & \tilde{U} \end{pmatrix},$$

where \bar{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

The Schur decomposition is not unique, because it depends on the ordering of the eigenvalues in T . Multiple eigenvalues can also cause the decomposition to be not to be unique.

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 eigenvector and eigenvalue are removed. This is an important technique in the solution of eigenvalue problems and is known as **deflation**. The advantage of the Schur decomposition is that it can be obtained using a numerically stable unitary transformation. The eigenvalues of A are displayed on the diagonal. The columns in $U = (u_1, u_2, \dots, u_n)$ are called **Schur vectors**. It is easy to verify that the nested sequence of subspaces

$$\mathcal{S}_k = \text{span}[u_1, \dots, u_k], \quad k = 1 : n,$$

are invariant subspaces, i.e., if $z \in \mathcal{S}_k$ implies that $Az \in \mathcal{S}_k$. Of the Schur vectors in general only the first u_1 is an eigenvector. However, since the Schur basis is orthogonal it is often preferable in many applications to the eigenvector basis.

For any Hermitian matrix A there exists a unitary matrix such that $A = U\Lambda U^H$, where Λ is real and diagonal. We now consider the extension of this class of matrices when we permit Λ to be complex. A matrix $A \in \mathbf{C}^{n \times n}$ that satisfies

$$A^H A = AA^H. \quad (10.1.27)$$

is said to be **normal**. Important classes of normal matrices in $\mathbf{C}^{n \times n}$ are:

- A is Hermitian, $A^H = A$.
- A is skew-Hermitian, $A^H = -A$.
- A is unitary $A^H = A^{-1}$.

For matrices $A \in \mathbf{R}^{n \times n}$ the corresponding terms are symmetric ($A^T = A$), skew-symmetric ($A^T = -A$), and orthogonal ($A^T = A^{-1}$).

Theorem 10.1.14.

A matrix $A \in \mathbf{C}^{n \times n}$ is normal if and only if A can be unitarily diagonalized, i.e., there exists a unitary matrix $U \in \mathbf{C}^{n \times n}$ such that

$$U^H AU = D = \text{diag}(\lambda_1, \dots, \lambda_n).$$

Proof. If A is unitarily diagonalizable then

$$A^H A = UD^H DU^H = UDD^H U^H = AA^H,$$

and A is normal. On the other hand, if A is normal, then for unitary U

$$(U^H AU)^H U^H AU = U^H (A^H A) U = U^H (AA^H) U = U^H AU (U^H AU)^H.$$

so $U^H AU$ 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. $T^H T = TT^H$. Equating the $(1,1)$ -element on both sides of the equation $T^H T = TT^H$ 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 offdiagonal elements in T vanishes, and so T is diagonal. \square

If A is Hermitian, $A^H = A$ it follows that $\lambda^H = \lambda$, i.e. the eigenvalues of a Hermitian matrix are real. For a skew-Hermitian matrices $\lambda^H = -\lambda$, which implies that the eigenvalues are zero or purely imaginary.

From Theorem 10.1.14 it follows that any Hermitian matrix may be decomposed into

$$A = U \Lambda U^H = \sum_{i=1}^n \lambda_i u_i u_i^H. \quad (10.1.28)$$

with λ_i real. In the special case that A is real and symmetric we can take U to be real and orthogonal, $U = Q = (q_1, \dots, q_n)$, where q_i are orthonormal eigenvectors.

For a unitary matrices (or real orthogonal) matrix $A^{-1} = A^H$ and hence $\lambda^{-1} = \bar{\lambda}$ or $|\lambda|^2 = 1$. Thus unitary matrices and real orthogonal matrices have eigenvalues on the unit circle. For a real orthogonal matrix the complex eigenvalues must occur in complex conjugate pairs. Hence, if n odd a real orthogonal matrix must have a real eigenvalue equal to 1 or -1 .

Note that although U in the Schur decomposition (10.1.26) is not unique, $\|N\|_F$ is independent of the choice of U , and

$$\Delta_F^2(A) \equiv \|N\|_F^2 = \|A\|_F^2 - \sum_{i=1}^n |\lambda_i|^2.$$

The quantity $\Delta_F(A)$ is called the **departure from normality** of A .

The following relationship between unitary and Hermitian matrices is called the Cayley parameterization.

Theorem 10.1.15.

If U is unitary and does not have -1 as an eigenvalue, then

$$U = (I + iH)(I - iH)^{-1},$$

where H is Hermitian and uniquely determined by the formula

$$H = i(I - U)(I + U)^{-1},$$

It is often desired to sort the diagonal elements of the matrix T in the Schur decomposition. For example, one may want close eigenvalues to appear in adjacent positions. Another application is when one wants to find the invariant subspace corresponding to a certain subset of the eigenvalues of T . These then have to moved to the top of the Schur form. Any such reordering can be achieved by a sequence of unitary transformations, each of which swaps two adjacent diagonal elements. Consider the 2×2 example

$$A = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix}, \quad a_{22} \neq a_{11}.$$

Then we seek a unitary 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 $|s|^2 + |\bar{c}|^2 = 1$, It can be verified that this holds if the elements in U are chosen as

$$\begin{aligned}\rho &= (|a_{11} - a_{22}|^2 + |a_{12}|^2)^{1/2}, \\ \bar{c} &= a_{12}/\rho, \quad s = (a_{11} - a_{22})\rho.\end{aligned}$$

This reordering algorithm can also be used to move a set of arbitrary eigenvalues to the upper left corner of T .

If the matrix A is real, we would like to restrict ourselves to real similarity transformations, since otherwise we introduce complex elements in $U^{-1}AU$. If A has complex eigenvalues, then A obviously cannot be reduced to triangular form by a real orthogonal transformation. For a real matrix A the eigenvalues occur in complex conjugate pairs, and it is possible to reduce A to block triangular form T , with 1×1 and 2×2 diagonal blocks, in which the 2×2 blocks correspond to pairs of complex conjugate eigenvalues. T is then said to be in **quasi-triangular** form, and we call the decomposition the real quasi-Schur form.

Theorem 10.1.16 (*The real Schur form*).

Given $A \in \mathbf{R}^{n \times n}$ there exists a real orthogonal matrix $Q \in \mathbf{R}^{n \times n}$ such that

$$Q^T A Q = T = D + N, \quad (10.1.29)$$

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 A have the complex eigenvalue $\lambda \neq \bar{\lambda}$ corresponding to the eigenvector x . Then, since $A\bar{x} = \bar{\lambda}\bar{x}$, $\bar{\lambda}$ is also an eigenvalue with eigenvector $\bar{x} \neq x$, and $\mathcal{R}(x, \bar{x})$ is an invariant subspace of dimension two. Let

$$X_1 = (x_1, x_2), \quad x_1 = x + \bar{x}, \quad x_2 = i(x - \bar{x})$$

be a real basis for this invariant subspace. Then $AX_1 = X_1M$ where $M \in \mathbf{R}^{2 \times 2}$ has eigenvalues λ and $\bar{\lambda}$. Let $X_1 = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1R$ be the QR decomposition of X_1 . Then $AQ_1R = Q_1RM$ or $AQ_1 = Q_1P$, where $P = RMR^{-1} \in \mathbf{R}^{2 \times 2}$ is similar to M . Using (10.2.40) 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 we need to be able to swap two adjacent diagonal blocks of size 1×1 or 2×2 using a real orthogonal similarity transformation. The scheme given for swapping diagonal elements can be generalized to handle these cases; see Bai, Demmel and McKenney [25].

If A is not normal, then the matrix T in its Schur decomposition cannot be diagonal. To transform T to a form closer to a diagonal matrix we have to use *non-unitary similarities*.

By Theorem 10.1.13 the eigenvalues can be ordered in the Schur decomposition so that

$$D = \text{diag}(\lambda_1, \dots, \lambda_n), \quad \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n.$$

We now show how to obtain the following block diagonal form:

Theorem 10.1.17 (Block Diagonal Decomposition).

Assume that $A \in \mathbf{C}^{n \times n}$ have 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

$$(UZ)^{-1} AUZ = Z^{-1} TZ = D, \quad D = \text{diag}(\lambda_1 I + N_1, \dots, \lambda_k I + N_k),$$

where D is block diagonal and N_i , $i = 1 : k$ are strictly upper triangular matrices. In particular, if the matrix A has n distinct eigenvalues the matrix D diagonal.

Proof. Consider first the 2×2 case and perform the similarity transformation

$$M^{-1} TM = \begin{pmatrix} 1 & -m \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 & t \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda_1 & m(\lambda_1 - \lambda_2) + t \\ 0 & \lambda_2 \end{pmatrix}.$$

where $\lambda_1 \neq \lambda_2$ and M is an upper triangular elementary elimination matrix; see Section 7.2.3. By taking $m = t/(\lambda_2 - \lambda_1)$, we can annihilate the off-diagonal element in T .

In the general case let t_{ij} be an element in T outside the block diagonal. Let M_{ij} be a matrix which differs from the unit matrix only in the (i, j) th element, which is equal to m_{ij} . Then as above we can choose m_{ij} so that the element (i, j) is annihilated by the similarity transformation $M_{ij}^{-1} TM_{ij}$. Since T is upper triangular this transformation will not affect any already annihilated off-diagonal elements in T with indices (i', j') if $j' - i' < j - i$. Hence, we can annihilate all elements t_{ij} outside the block diagonal in this way, 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 is left to the reader. \square

In the general case when A has multiple eigenvalues we need to identify clusters of “equal” eigenvalues in order to compute the block diagonal form. Numerically this can be a very difficult computational task. Even more difficult is to attempt to determine the Jordan block structure of the diagonal blocks containing multiple eigenvalues. Indeed, these difficulties serve to illustrate the limited usefulness of the Jordan normal form in numerical computations.

10.1.5 Sylvester’s Equation and Spectral Decomposition

A useful similarity transformation for a in block upper triangular matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \in \mathbf{C}^{n \times n}$$

is given in the following lemma.

Lemma 10.1.18.

Consider the similarity transformation

$$\begin{pmatrix} I_k & -X \\ 0 & I_{n-k} \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} I_k & X \\ 0 & I_{n-k} \end{pmatrix} = \begin{pmatrix} A_{11} & Y \\ 0 & A_{22} \end{pmatrix} \quad (10.1.30)$$

where $A_{11} \in \mathbf{C}^{k \times k}$ and A_{22} are square matrices. Then

$$Y = A_{12} - X A_{22} + A_{11} X$$

and the result is a **block diagonal** matrix if and only if X can be chosen to satisfy the equation

$$A_{11} X - X A_{22} = -A_{12}. \quad (10.1.31)$$

Proof. The lemma follows by performing the multiplications in (10.1.30). \square .

Equation, (10.1.31), which has the general form

$$AX - XB = C, \quad (10.1.32)$$

is called a **Sylvester equation**. It is a linear equation in the elements of X . Using properties of Kronecker products (see Section 7.6.5) it can be verified that Sylvester’s equation (10.1.31) is equivalent to the linear system

$$(I_n \otimes A - B^T \otimes I_m) \text{vec } X = \text{vec } C. \quad (10.1.33)$$

Here $\text{vec } X$ is the vector obtained by stacking the columns of X on top of each other.

Conditions for existence and uniqueness of solutions to Sylvester’s equation are given in the following theorem.

Theorem 10.1.19.

Sylvester's matrix equation (10.1.31) 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. From Theorem 10.1.13 follows the existence of the Schur decompositions

$$U_1^H A U_1 = S, \quad U_2^H B U_2 = T, \quad (10.1.34)$$

where S and T are upper triangular and U_1 and U_2 are unitary matrices. Using (10.1.34) equation (10.1.31) can be reduced to

$$S Y - Y T = F, \quad Y = U_1^H X U_2, \quad F = U_1^H C U_2.$$

The k th column of this equation is

$$S y_k - Y t_k = f_k, \quad k = 1 : n. \quad (10.1.35)$$

For $k = 1$ this equation gives $(S - t_{11}I)y_1 = f_1$. Here t_{11} is an eigenvalue of T and hence is *not* an eigenvalue of S . Thus, the upper triangular matrix $S - t_{11}I$ is nonsingular and y_1 is obtained by backsubstitution. Now, suppose that we have found y_1, \dots, y_{k-1} . From the k th column of the system we have

$$(S - t_{kk}I)y_k = f_k + \sum_{i=1}^k t_{ik}y_i.$$

Here the right hand side is known and, by the argument above, the triangular matrix $S - t_{kk}I$ nonsingular. Hence, it can be solved for y_k . The proof now follows by induction. \square

The proof is constructive and shows that the solution of the Sylvester equation can be computed by first finding the Schur decompositions of A and B and then solving m triangular equations. This is the Bartels–Stewart algorithm [37]. A similar but more efficient algorithm, where A is only reduced to Hessenberg form, has been given by Golub, Nash, and Van Loan [269].

Setting Here $B = -A^H$ in (10.1.31) we obtain the important special case

$$AX + XA^H = C, \quad (10.1.36)$$

which is the **Lyapunov equation**. By Theorem 10.1.19 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 [303].

Several generalizations of Sylvester's equation have applications in systems and control theory. An example is the algebraic **Riccati equation**⁴⁴

$$AX - XB - XDX = C, \quad X \in \mathbf{R}^{n \times m}. \quad (10.1.37)$$

⁴⁴Jacopo Francesco Riccati (1676–1754) Italian mathematician. His work in 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 equation and its variations have been a central object of study in control system and many algorithms have been developed. The solution X can be used for block triangularization using the similarity transformation

$$\begin{pmatrix} I & -X \\ 0 & I \end{pmatrix} \begin{pmatrix} A & -C \\ D & B \end{pmatrix} \begin{pmatrix} I & X \\ 0 & I \end{pmatrix} = \begin{pmatrix} A - XD & 0 \\ D & B + DX \end{pmatrix}.$$

By Lemma 10.2.19 a consequence of this is that the columns of $\begin{pmatrix} X \\ I \end{pmatrix}$ form an invariant subspace of the block matrix. An important special version of (10.1.37) is when $m = n$, $B = A^H$ and C and D are Hermitian. This is commonly called the continuous-time algebraic Riccati equation and plays a fundamental role in systems theory.

There are several possible algorithms for solving the Riccati equation. If X_0 is a given initial approximation one can try the simple iteration

$$AX_{k+1} - X_{k+1}B = C + X_kDX_k, \quad k = 0, 1, 2, \dots$$

Each iteration step requires the solution of a Sylvester equation. If A and B are reduced to upper triangular/Hessenberg form this can be solved cheaply. Convergence is at best linear. A more rapidly convergent iteration is obtained by using Newton's method; see Problem 10.1.16.

10.1.6 Nonnegative Matrices

Non-negative matrices play an important role in the theory of matrices. They are important in the study of convergence of iterative methods and arise in many applications such as queuing theory, stochastic processes, and input-output analysis.

A matrix $A \in \mathbf{R}^{m \times n}$ is called **nonnegative** if its entries a_{ij} are nonnegative for all i and j . It is called **positive** if $a_{ij} > 0$ for each i and j . If A and B are nonnegative, then so is their sum $A + B$ and product AB . Hence, nonnegative matrices form a convex set. Further, if A is nonnegative, then so is A^k for all $k \geq 0$.

A partial order can be defined on the set of matrices.

Definition 10.1.20.

Let A and B be two $m \times n$ matrices. Then if

$$a_{ij} \leq b_{ij}, \quad i = 1 : m, \quad j = 1 : n,$$

we write $A \leq B$. The binary relation " \leq " then defines a partial ordering of the matrices in $\mathbf{R}^{m \times n}$. This ordering is transitive since if $A \leq B$ and $B \leq C$ then $A \leq C$.

Note that two arbitrary matrices cannot be compared by this relation. Some useful properties of nonnegative matrices are the following:

Lemma 10.1.21. Let A, B , and C be nonnegative $n \times n$ matrices with $A \leq B$. Then

$$AC \leq BC \quad CA \leq CB, \quad A^k \leq B^k, \quad \forall k \geq 0.$$

Definition 10.1.22.

A nonsingular matrix $A \in \mathbf{R}^{n \times n}$ is said to be an *M-matrix* if it has the following properties:

1. $a_{ii} > 0$ for $i = 1 : n$.
2. $a_{ij} \leq 0$ for $i \neq j$, $i, j = 1 : n$.
3. $A^{-1} \geq 0$.

Lemma 10.1.23. Let $A \in \mathbf{R}^{n \times n}$ be a square nonnegative matrix and let $s = Ae$, $e = (1 \ 1 \ \cdots \ 1)^T$ be the vector of row sums of A . Then

$$\min_i s_i \leq \rho(A) \leq \max_i s_i = \|A\|_1. \quad (10.1.38)$$

The class of nonnegative square matrices have remarkable spectral properties. These were discovered in (1907) by Perron⁴⁵ for positive matrices and amplified and generalized for nonnegative irreducible (see Def. 7.1.3) matrices by Frobenius (1912).

Theorem 10.1.24 (Perron–Frobenius Theorem).

Let $A \in \mathbf{R}^{n \times n}$ be a nonnegative irreducible matrix, Then

- (i) A has a real positive simple eigenvalue equal to $\rho(A)$;
- (ii) To $\rho(A)$ corresponds an eigenvector $x > 0$.
- (iii) $\rho(A)$ increases when any entry of A increases.
- (iv) The eigenvalues of modulus $\rho(A)$ are all simple. If there are m eigenvalues of modulus ρ , they must be of the form

$$\lambda_k = \rho e^{2k\pi i/m}, \quad k = 0 : m - 1.$$

If $m = 1$, the the matrix is called **primitive**.

- (v) If $m > 1$, then there exists a permutation matrix P such that

$$PAP^T = \begin{pmatrix} 0 & A_{12} & 0 & \cdots & 0 \\ 0 & 0 & A_{23} & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \cdots & A_{m-1,m} \\ A_{m1} & 0 & 0 & \cdots & 0 \end{pmatrix},$$

⁴⁵Oskar Perron (1880–1975) German mathematician held positions at Heidelberg and Munich. His work covered a vide range of topics. He also wrote important textbooks on continued fractions, algebra, and non-Euclidean geometry.

where the zero blocks down the diagonal are square. Such a matrix is called **cyclic** of index $m > 1$.

Since the proof of the Perron–Frobenius theorem is too long to be given here we refer to, e.g., Varga [594, Sec. 2.1–2.2] or Berman and Plemmons [50, pp. 27,32]. The Perron–Frobenius theorem is an important tool for analyzing the Markov chains and the asymptotic convergence of stationary iterative methods; see Section 11.1.3.

Review Questions

- 1.1** How are the eigenvalues and eigenvectors of A affected by a similarity transformation?
 - 1.2** What is meant by a (right) invariant subspace of A ? Describe how a basis for an invariant subspace can be used to construct a similarity transformation of A to block triangular form. How does such a transformation simplify the computation of the eigenvalues of A ?
 - 1.3** What is meant by the algebraic multiplicity and the geometric multiplicity of an eigenvalue of A ? When is a matrix said to be defective?
 - 1.4** What is the Schur decomposition of a matrix $A \in \mathbf{C}^{n \times n}$?
 - (b) If A is real how can the Schur decomposition be modified so that a real decomposition is obtained?
 - 1.5** Show that if A and B are normal and $AB = BA$, then AB is normal.
 - 1.6** How can the class of matrices which are diagonalizable by unitary transformations be characterized?
 - 1.7** What is meant by a defective eigenvalue? Give a simple example of a matrix with a defective eigenvalue.
 - 1.8** Give an example of a matrix, for which the minimal polynomial has a lower degree than the characteristic polynomial. Is the characteristic polynomial always divisible by the minimal polynomial?
 - 1.9** Prove the Cayley–Hamilton theorem for a diagonalizable matrix. Then generalize to an arbitrary matrix, either as in the text or by using Bellman’s approximation theorem, (Theorem 10.1.19).
 - 1.10** Give in terms of the spectra of A and B a necessary and sufficient condition for the solution of the matrix equation $AX - XB = C$. Describe one method to solve such a matrix equation.
-

Problems

- 1.1** A matrix $A \in \mathbf{R}^{n \times n}$ is called nilpotent if $A^k = 0$ for some $k > 0$. Show that a nilpotent matrix can only have 0 as an eigenvalue.

- 1.2** Show that if λ is an eigenvalue of a unitary matrix U , then $|\lambda| = 1$.
- 1.3** (a) Let $A = xy^T$, where x and y are vectors in \mathbf{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 $u \in \mathbf{R}^n$, $\|u\|_2 = 1$?
- 1.4** What are the eigenvalues of a Givens' rotation

$$R(\theta) = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}?$$

When are the eigenvalues real?

- 1.6** Show that if A is normal, Hermitian, or unitary, so is A^p for any integer p . (If $p < 0$ the A is assumed to be nonsingular.) If p is odd and A is skew-Hermitian, then A^p is skew Hermitian.
- 1.7** Let $A \in \mathbf{C}^{n \times n}$ be an Hermitian matrix, λ an eigenvalue of A , and z the corresponding eigenvector. Let $A = S + iK$, $z = x + iy$, where S, K, x, y are real. Show that λ is a double eigenvalue of the real symmetric matrix

$$\begin{pmatrix} S & -K \\ K & S \end{pmatrix} \in \mathbf{R}^{2n \times 2n},$$

and determine two corresponding eigenvectors.

- 1.8** Show that the matrix

$$K_n = \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

has the characteristic polynomial

$$p(\lambda) = (-1)^n(\lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n).$$

K_n is called the **companion matrix** of $p(\lambda)$. Determine the eigenvectors of K_n corresponding to an eigenvalue λ , and show that there is only one eigenvector even when λ is a multiple eigenvalue.

Remark: The term companion matrix is sometimes used for slightly different matrices, where the coefficients of the polynomial appear, e.g., in the last row or in the last column.

- 1.9** Draw the graphs $G(A)$, $G(B)$ and $G(C)$, where

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Show that A and C are irreducible but B is reducible.

1.10 Prove the Cayley–Hamilton theorem for a diagonalizable matrix. Then generalize to an arbitrary matrix, either as in the text or by using Bellman’s approximation theorem, (Theorem 10.1.19).

1.11 Find a similarity transformation $X^{-1}AX$ 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?

1.12 Show that the Sylvester equation (10.1.31) can be written as the linear system

$$(I_m \otimes A - B^T \otimes I_n) \text{vec}(X) = \text{vec}(C), \quad (10.1.39)$$

where \otimes denotes the Kronecker product and $\text{vec}(X)$ is the column vector obtained by stacking the column of X on top of each other.

1.13 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.

1.14 Show that the eigenvalues λ_i of a matrix A satisfy the inequalities

$$\sigma_{\min}(A) \leq \min_i |\lambda_i| \leq \max_i |\lambda_i| \sigma_{\max}(A).$$

Hint: Use the fact that the singular values of A and its Schur decomposition $Q^T AQ = \text{diag}(\lambda_i) + N$ are the same.

1.15 (Walter Gander) The eigenvalues of a matrix $A \in \mathbf{R}^n$ are the solution to the characteristic equation $f(\lambda) = \det(\lambda I - A) = 0$. For a given value λ the function $f(\lambda)$ can be stably evaluated by computing the LU factorization with partial pivoting $P(\lambda I - A) = LU$, where L is unit lower triangular, and using the formula

$$\det(\lambda I - A) = u_{11}u_{22} \cdots u_{nn}, \quad (10.1.40)$$

This suggests that the eigenvalues of A can be computed using some method for solving a nonlinear equation. Each function evaluation requires $O(n^3)$ operations. Since the expression (10.1.40) can easily overflow or underflow it is preferable to use the equation

$$g(\lambda) = \log \det(\lambda I - A) = \sum_{i=1}^n \log u_{ii} = 0.$$

Implement this idea and test it by generating a random test matrix with eigenvalues $1, 2, \dots, n$ using the MATLAB statements

```
x = [1:n]; Q = rand(n);
A = Q*diag(x)*inv(Q);
```

Try $n = 10$ and $n = 20$.

1.16 Verify that Sylvester's equation (10.1.31) 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 (10.1.8) 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.

1.17 Let X_k be an approximate solution to the Riccati equation

$$C_0 + C_1 X + X C_2 + X C_2 X = 0.$$

Suppose we have a good approximation X_0 to the solution X . Derive a Newton method for computing X , which in each step requires the solution of a Sylvester equation is solved.

Hint: Show that

$$F(X_k + \Delta X_k) = (C_1 + X_k C_3) \Delta X_k + \Delta X_k (C_2 + C_3 X_k) + O(\Delta X_k^2).$$

10.2 Perturbation Theory and Eigenvalue Bounds

10.2.1 Geršgorin's Theorem

Methods for computing eigenvalues and eigenvectors are subject to roundoff errors. In order to estimate the error in the computed result we need to study the effects of a perturbation of the matrix A on the eigenvalues and eigenvectors of A . We start with a classical result.

The simple and powerful **Geršgorin circle theorem**⁴⁶ can be used both to locate eigenvalues of a complex matrix and to derive perturbation results.

Theorem 10.2.1 (Geršgorin's Circle Theorem).

All the eigenvalues of the complex matrix $A \in \mathbf{C}^{n \times n}$ lie in the union of the Geršgorin disks

$$\mathcal{D}_i = \{z \mid |z - a_{ii}| \leq r_i\}, \quad r_i = \sum_{j=1, j \neq i}^n |a_{ij}|, \quad i = 1 : n. \quad (10.2.1)$$

in the complex plane

Proof. Let λ be an eigenvalue of A and $x = (x_1 \dots x_n)^T$ a corresponding eigenvector. Then $Ax = \lambda x$, or equivalently

$$(\lambda - a_{ii})x_i = \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j, \quad i = 1 : n.$$

⁴⁶Semyon Aranovich Geršgorin (1901–1933), Russian mathematician, who worked at the Leningrad Mechanical Engineering Institute. He published his circle theorem 1931 in [245].

Choose an index i such that $|x_i| = \|x\|_\infty$. Then, taking absolute values and

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \frac{|x_j|}{|x_i|} \leq r_i. \quad (10.2.2)$$

This shows that to each eigenvalue λ there is a disk with midpoint a_{ii} and radius r_i which contains this eigenvalue. Therefore all eigenvalues lie in the union of the disks. \square

If the matrix A is strictly diagonally dominant then $|a_{ii}| > r_i$, $i = 1 : n$ and the Geršgorin disks do not contain the origin. It follows from the theorem that A is nonsingular.

Geršgorin's theorem is very useful for getting estimates of the location of eigenvalues. For nearly diagonal matrices it can give accurate estimates for the eigenvalues. Since A and A^T have the same eigenvalues we can, in the non-Hermitian case, obtain more information about the location of the eigenvalues simply by applying the theorem also to A^T .

From (10.2.2) it follows that if the i th component of the eigenvector is maximal, then λ lies in the disk \mathcal{D}_i . Otherwise the Geršgorin theorem does not say in which of the disks the eigenvalues lie. Sometimes it is possible to decide this as the following theorem shows.

Theorem 10.2.2.

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

$$A(t) = tA + (1-t)D_A, \quad D_A = \text{diag}(a_{ii}).$$

The coefficients in the characteristic polynomial are continuous functions of t , and hence the eigenvalues $\lambda(t)$ of $A(t)$ are also continuous functions of t . Since $A(0) = D_A$ and $A(1) = A$ we have $\lambda_i(0) = a_{ii}$ and $\lambda_i(1) = \lambda_i$. 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 10.2.1.

The matrix

$$A = \begin{pmatrix} 2 & -0.1 & 0.05 \\ 0.1 & 1 & -0.2 \\ 0.05 & -0.1 & 1 \end{pmatrix},$$

with eigenvalues $\lambda_1 = 0.8634$, $\lambda_2 = 1.1438$, $\lambda_3 = 1.9928$, has the Geršgorin disks

$$\mathcal{D}_1 = \{z \mid |z - 2| \leq 0.15\}; \quad \mathcal{D}_2 = \{z \mid |z - 1| \leq 0.3\}; \quad \mathcal{D}_3 = \{z \mid |z - 1| \leq 0.15\}.$$

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 $\mathcal{D}_2 \cup \mathcal{D}_3 = \mathcal{D}_2$.

A useful sharpening of Geršgorin's theorem is obtained by using the fact that the eigenvalues are invariant under a diagonal similarity transformations

$$\hat{A} = DAD^{-1}, \quad D = \text{diag}(d_1, \dots, d_n).$$

where $d_i > 0$, $i = 1 : n$. Applying the Theorem 10.2.1 to \hat{A} it follows that all the eigenvalues of A lie in the union of the disks

$$|z - a_{ii}| \leq \frac{1}{d_i} \sum_{j=1, j \neq i}^n d_j |a_{ij}|, \quad i = 1 : n. \quad (10.2.3)$$

The scaling D can be chosen to minimize the radius of one particular disk; see Problem 10.2.3.

There is another useful sharpening of Geršgorin's Theorem in case the matrix A is irreducible, cf. Def. 7.1.3.

Theorem 10.2.3.

If the matrix A is irreducible then each eigenvalue λ lies in the interior of the union of the Geršgorin disks, unless it lies on the boundary of all Geršgorin disks.

Proof. If λ lies on the boundary of the union of the Geršgorin disks, then we have

$$|\lambda - a_{ii}| \geq r_i, \quad \forall i. \quad (10.2.4)$$

Let x be a corresponding eigenvector and assume that $|x_{i_1}| = \|x\|_\infty$. Then from the proof of Theorem 10.2.1 and (10.2.4) it follows that $|\lambda - a_{i_1 i_1}| = r_{i_1}$. But (10.2.2) 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 for any $j \neq i$ there is a path $i = i_1, i_2, \dots, i_p = j$. It follows that λ must lie on the boundary of all Geršgorin disks. \square

Example 10.2.2. Consider the real, symmetric matrix

$$A = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbf{R}^{n \times n}.$$

Its Geršgorin disks are

$$|z - 2| \leq 2, \quad i = 2 : n - 1, \quad |z - 2| \leq 1, \quad i = 1, n,$$

and it follows that all eigenvalues of A satisfy $\lambda \geq 0$. Since zero is on the boundary of the union of these disks, but *not* on the boundary of all disks, zero cannot be an eigenvalue of A . Hence, all eigenvalues are *strictly* positive and A is positive definite.

10.2.2 General Perturbation Theorems

In this section we consider the sensitivity of eigenvalue and eigenvectors to perturbations.

Theorem 10.2.4 (*Bauer–Fike’s Theorem*).

Let the matrix $A \in \mathbf{C}^{n \times n}$ be diagonalizable,

$$Y^H AX = D = \text{diag}(\lambda_1, \dots, \lambda_n), \quad Y^H X = I,$$

and let μ be an eigenvalue to the perturbed matrix $A + E$. Then for Hölder norm $\|\cdot\|_p$ it holds that

$$\min_{1 \leq i \leq n} |\mu - \lambda_i| \leq \kappa_p(X) \|E\|_p. \quad (10.2.5)$$

where $\kappa_p(X) = \|X^{-1}\|_p \|X\|_p$ is the condition number of the eigenvector matrix.

Proof. We can assume that μ is not an eigenvalue of A , since otherwise (10.2.5) holds trivially. Since μ is an eigenvalue of $A + E$ the matrix $A + E - \mu I$ is singular and so is also

$$Y^H (A + E - \mu I) X = (D - \mu I) + Y^H E X.$$

Then there is a vector $z \neq 0$ such that

$$(D - \mu I)z = -Y^H E X z.$$

Solving for z and taking norms we obtain

$$\|z\|_p \leq \kappa_p(X) \|(D - \mu I)^{-1}\|_p \|E\|_p \|z\|_p.$$

The theorem follows by dividing by $\|z\|_p$ and using the fact that for any Hölder norm $\|(D - \mu I)^{-1}\|_p = 1 / \min_{1 \leq i \leq n} |\lambda_i - \mu|$. \square

The Bauer–Fike theorem shows that $\kappa_p(X)$ is an upper bound for the condition number of the eigenvalues of any *diagonalizable matrix* A . If A is normal we know from the Schur decomposition that we can take X to be a unitary matrix. Then $\kappa_2(X) = 1$, which shows the important result that *the eigenvalues of a normal matrix are perfectly conditioned, also if they have multiplicity greater than one*. On the other hand, if the matrix A which is close to a defective matrix then some eigenvalues are very ill-conditioned, see Example 10.1.2, and the following example.

Example 10.2.3.

Consider the matrix $A = \begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix}$, $0 < \epsilon$ with eigenvector matrix

$$X = \begin{pmatrix} 1 & 1 \\ \sqrt{\epsilon} & -\sqrt{\epsilon} \end{pmatrix}, \quad X^{-1} = \frac{0.5}{\sqrt{\epsilon}} \begin{pmatrix} \sqrt{\epsilon} & 1 \\ \sqrt{\epsilon} & -1 \end{pmatrix}.$$

If $\epsilon \ll 1$ then

$$\kappa_\infty(X) = \|X^{-1}\|_\infty \|X\|_\infty = \frac{1}{\sqrt{\epsilon}} + 1 \gg 1.$$

Note that in the limit when $\epsilon \rightarrow 0$ the matrix A is not diagonalizable.

In general a matrix may have a mixture of well-conditioned and ill-conditioned eigenvalues. Therefore, it is useful to be able to estimate the perturbation of individual eigenvalues of a matrix A . To do this we now derive first order estimates for the perturbation of simple eigenvalues and corresponding eigenvectors.

Theorem 10.2.5.

Let the matrix A have the eigendecomposition $Y^H A X = D$, where $D = \text{diag}(\lambda_1, \dots, \lambda_n)$. Let λ_i be a simple eigenvalue of A and denote by x_i and y_i the corresponding right and left eigenvectors

$$Ax_i = \lambda_i x_i, \quad y_i^H A = \lambda_i y_i^H.$$

Then for sufficiently small ϵ the matrix $A + \epsilon E$, $\epsilon \geq 0$, has a simple eigenvalue $\lambda_i(\epsilon)$ such that

$$\lambda_i(\epsilon) = \lambda_i + \epsilon \frac{y_i^H E x_i}{y_i^H x_i} + O(\epsilon^2). \quad (10.2.6)$$

If the eigenvectors are normalized to have unit length $\|x_i\|_2 = \|y_i\|_2 = 1$, the $|y_i^H E x_i| \leq \|E\|_2$ and an upper bound for the perturbation is given by

$$|\lambda_i(\epsilon) - \lambda_i| \leq \epsilon \frac{\|E\|_2}{|y_i^H x_i|} + O(\epsilon^2). \quad (10.2.7)$$

Proof. Since λ_i is a simple eigenvalue there is a $\delta > 0$ such that the disk $\mathcal{D} = \{\mu \mid |\mu - \lambda_i| < \delta\}$ does not contain any eigenvalues of A other than λ_i . Then using Theorem 10.2.2 it follows that for sufficiently small values of ϵ the matrix $A + \epsilon E$ has a simple eigenvalue $\lambda_i(\epsilon)$ in \mathcal{D} . If we denote a corresponding eigenvector $x_i(\epsilon)$ then

$$(A + \epsilon E)x_i(\epsilon) = \lambda_i(\epsilon)x_i(\epsilon).$$

Using results from function theory, it can be shown that $\lambda_i(\epsilon)$ and $x_i(\epsilon)$ are analytic functions of ϵ for $\epsilon < \epsilon_0$ for some $\epsilon_0 > 0$. Differentiating with respect to ϵ and putting $\epsilon = 0$ we get

$$(A - \lambda_i I)x'_i(0) + Ex_i = \lambda'_i(0)x_i. \quad (10.2.8)$$

Since $y_i^H(A - \lambda_i I) = 0$ we can eliminate $x'_i(0)$ by multiplying this equation with y_i^H and solve for $\lambda'_i(0) = y_i^H E x_i / y_i^H x_i$, which proves (10.2.6). \square

The theorem shows that

$$s_i = \frac{|y_i^H x_i|}{\|y_i\|_2 \|x_i\|_2} = \cos \theta(x_i, y_i) \quad (10.2.9)$$

is the *reciprocal condition number of a simple eigenvalue* λ_i . Note that $\theta(x_i, y_i)$ is the acute angle between the left and right eigenvector corresponding to λ_i . If A is a normal matrix, we get $s_i = 1$.

Remark 10.2.1. The quantity s_i in (10.2.9) was introduced by Wilkinson [611, 68–69] as a measure of the sensitivity of an eigenvalue.

The above theorem shows that for perturbations in A of order ϵ , a simple eigenvalue λ of A will be perturbed by an amount approximately equal to $\epsilon/s(\lambda)$. If λ is a defective eigenvalue, then there is no similar result. *Indeed, if the largest Jordan block corresponding to λ is of order k , then perturbations to λ of order $\epsilon^{1/k}$ can be expected.* Note that for a Jordan box we have $x = e_1$ and $y = e_m$ and so $s(\lambda) = 0$ in (10.2.7).

Example 10.2.4.

Consider the perturbed diagonal matrix

$$A + \epsilon E = \begin{pmatrix} 1 & \epsilon & 2\epsilon \\ \epsilon & 2 & \epsilon \\ \epsilon & 2\epsilon & 2 \end{pmatrix}.$$

Here A is diagonal with left and right eigenvector equal to $x_i = y_i = e_i$. Thus, $y_i^H E x_i = e_{ii} = 0$ and the first order term in the perturbation of the simple eigenvalues are zero. For $\epsilon = 10^{-3}$ the eigenvalues of $A + E$ are

$$0.999997, \quad 1.998586, \quad 2.001417.$$

Hence, the perturbation in the simple eigenvalue λ_1 is of order 10^{-6} . Note that the Bauer–Fike theorem would predict perturbations of order 10^{-3} for all three eigenvalues.

We now consider the perturbation of an eigenvector x_i corresponding to a simple eigenvalue λ_i . Assume that the matrix A is diagonalizable and that x_1, \dots, x_n are linearly independent eigenvectors. Then we can write

$$x_i(\epsilon) = x_i + \epsilon x'_i(0) + O(\epsilon^2), \quad x'_i(0) = \sum_{k \neq i} c_{ki} x_k,$$

where we have normalized $x_i(\epsilon)$ to have unit component along x_i . Substituting the expansion of $x'_i(0)$ into (10.2.8) we get

$$\sum_{k \neq i} c_{ki} (\lambda_k - \lambda_i) x_k + E x_i = \lambda'_i(0) x_i.$$

Multiplying by y_j^H and using $y_j^H x_k = 0$, $k \neq j$, we obtain

$$c_{ji} = \frac{y_j^H E x_i}{(\lambda_j - \lambda_i) y_j^H x_j}, \quad j \neq i. \quad (10.2.10)$$

Normalizing so that $\|x_i\|_2 = \|y_i\|_2 = 1$, $i = 1 : n$, we obtain

$$|x_i(\epsilon) - x_i| \leq \|E\|_2 \sum_{j \neq i} \frac{1}{|\lambda_j - \lambda_i| |y_j^H x_j|}, \quad (10.2.11)$$

Hence, the sensitivity of the eigenvectors depend on the separation

$$\delta_j = \min_{i \neq j} |\lambda_i - \lambda_j|$$

between λ_j and the rest of the spectrum of A . If several eigenvectors corresponds to a multiple eigenvalue these are not uniquely determined, which is consistent with this result. However, even when the individual eigenvectors are sensitive to perturbations it may be that an invariant subspace containing these eigenvectors is well determined.

Let (μ, x) , $\|x\|_2 = 1$, be a given approximate eigenpair of a matrix A . The corresponding residual vector is

$$r = Ax - \mu x. \quad (10.2.12)$$

If $r = 0$, then (μ, x) is an exact eigenpair of A . By continuity we can expect that the size of the residual norm $\|r\|$ can be used as a measure of the accuracy of x and μ as an eigenpair of A . Indeed, the following simple backward error bound is easy to prove.

Theorem 10.2.6.

Let (μ, x) , $\|x\|_2 = 1$, be a given approximate eigenpair of a matrix $A \in \mathbf{C}^{n \times n}$, and let $r = Ax - \mu x$ be the corresponding residual vector. Then (μ, x) is an exact eigenpair of the matrix $\tilde{A} = A + E$, where

$$E = -rx^H, \quad \|E\|_2 = \|r\|_2. \quad (10.2.13)$$

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

$$|\lambda - \mu| \leq \kappa_p(X) \|r\|_2. \quad (10.2.14)$$

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 10.2.4) proves the second result. \square

We conclude that (μ, x) , $\|x\|_2 = 1$, is a numerically acceptable eigenpair of the matrix A if $\|Ax - \mu x\|_2$ is of the order of the round-off unit.

Often we are only 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 10.2.6. The optimal choice in this sense is given by the **Rayleigh quotient**.

Theorem 10.2.7.

Let x be a given nonzero vector and consider the residual $r(\mu) = Ax - \mu x$. Then $\|r(\mu)\|_2$ is minimized for $\mu = \rho_A(x)$, where

$$\rho_A(x) = \frac{x^H A x}{x^H x}. \quad (10.2.15)$$

is the Rayleigh quotient of x .

Proof. To minimize $|Ax - \mu x|_2$ is a linear least squares problem for the unknown scalar μ . The normal equations are $(x^H x)\mu = x^H A x$, which gives (10.2.15). \square

The Rayleigh quotient is a homogeneous function of x ,

$$\rho(\alpha x) = \rho(x)$$

for all scalar $\alpha \neq 0$. For a non-Hermitian matrix we can consider the more general Rayleigh quotient

$$\rho_A(x, y) = \frac{y^H A x}{y^H x}, \quad y^H x \neq 0. \quad (10.2.16)$$

When either x is right eigenvector or y a left eigenvector $\rho_A(x, y)$ equals the corresponding eigenvalue λ . By continuity, when x or y is close to an eigenvector then ρ is an approximation to λ .

10.2.3 Hermitian Matrices

We have seen that Hermitian, and real symmetric matrices have real and perfectly conditioned eigenvalues. For this class of matrices it is possible to get more informative perturbation bounds, than those derived earlier. In this section we give several classical theorems, which are all related to each other.

We assume in the following that the eigenvalues of A have been ordered 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 [205].

Theorem 10.2.8 (Fischer's Theorem). *Let the Hermitian matrix A have eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ ordered so that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. Then*

$$\lambda_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x \in \mathcal{S} \\ x^H x=1}} x^H A x \quad (10.2.17)$$

and

$$\lambda_i = \min_{\dim(\mathcal{S})=n-i+1} \max_{\substack{x \in \mathcal{S} \\ x^H x=1}} x^H A x. \quad (10.2.18)$$

where \mathcal{S} denotes a subspace of \mathbf{C}^n .

Proof. See Stewart and Sun [552, Sec. 4.2]. \square

The formulas (10.2.17) and (10.2.18) are called the max-min and the min-max characterization, respectively. In particular, the extreme eigenvalues λ_1 and λ_n are characterized by

$$\lambda_1 = \max_{\substack{x \in \mathbb{C}^n \\ x \neq 0}} \rho(x), \quad \lambda_n = \min_{\substack{x \in \mathbb{C}^n \\ x \neq 0}} \rho(x). \quad (10.2.19)$$

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 10.2.9.

Let $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_n$, $\beta_1 \geq \beta_2 \geq \dots \geq \beta_n$, and $\gamma_1 \geq \gamma_2 \geq \dots \geq \gamma_n$ be the eigenvalues of the Hermitian matrices A , B , and $C = A + B$. Then

$$\alpha_i + \beta_1 \geq \gamma_i \geq \alpha_i + \beta_n, \quad i = 1 : n. \quad (10.2.20)$$

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 (10.2.18). The first equality 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 which lies between the smallest and greatest eigenvalues of B . If the matrix $\text{rank}(B) < n$, the result can be sharpened, see Parlett [478, Sec. 10-3]. An important case is when $B = \pm z z^T$ is a rank one matrix. Then B has only one nonzero eigenvalue equal to $\rho = \pm \|z\|_2^2$. In this case the perturbed eigenvalues will satisfy the relations

$$\lambda'_i - \lambda_i = m_i \rho, \quad 0 \leq m_i, \quad \sum m_i = 1. \quad (10.2.21)$$

Hence, all eigenvalues are shifted by an amount which lies between zero and ρ .

An important application is to get bounds for the eigenvalues λ'_i of $A + E$, where A and E are Hermitian matrices. Usually the eigenvalues of E are not known, but we can get bounds using

$$\max\{|\lambda_1(E)|, |\lambda_n(E)|\} = \rho(E) = \|E\|_2.$$

This agrees with the previous result that the eigenvalues of a Hermitian matrix are perfectly conditioned. Note that this result also holds for *large perturbations*. This result will be much used in the following and we state it together with a slightly sharper result in the following theorem.

Theorem 10.2.10.

Let A and $A + E$ be symmetric matrices with eigenvalues λ_i and λ'_i , $i = 1 : n$. Then it holds that

$$|\lambda_i - \lambda'_i| \leq \|E\|_2, \quad (10.2.22)$$

and

$$\sqrt{\sum_{i=1}^n |\lambda_i - \lambda'_i|^2} \leq \|E\|_F. \quad (10.2.23)$$

Proof. The first result follows directly from (10.2.21). The second result is the **Wielandt–Hoffman theorem**. The proof is not simple and we refer to [336] or [611, Sec. 2.48]. \square

The following theorem, due to Cauchy 1829, relates the eigenvalues of a principal submatrix to the eigenvalues of the original matrix. I

Theorem 10.2.11 (Cauchy's Interlacing Theorem).

Let B be a principal submatrix of order $n - 1$ of a Hermitian matrix $A \in \mathbf{C}^{n \times n}$. Then, the eigenvalues of B , $\mu_1 \geq \mu_2 \geq \dots \geq \mu_{n-1}$ interlace the eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ of A , that is

$$\lambda_i \geq \mu_i \geq \lambda_{i+1}, \quad i = 1 : n - 1. \quad (10.2.24)$$

Proof. The theorem follows from Fischer's theorem. Without loss of generality, we assume that B is the leading principal submatrix of A ,

$$A = \begin{pmatrix} B & a^H \\ a & \alpha \end{pmatrix}.$$

Consider the subspace of vectors $\mathcal{S}' = \{x \in \mathbf{C}^n, x \perp e_n\}$. Then with $x \in \mathcal{S}'$ we have $x^H A x = (x')^H B x'$, where $x' = ((x')^H, 0)$. Using the minimax characterization (10.2.17) of the eigenvalue λ_i it follows that

$$\lambda_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H A x}{x^H x} \geq \max_{\substack{\dim(\mathcal{S})=i \\ \mathcal{S} \perp e_n} \atop \mathcal{S} \neq 0} \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H A x}{x^H x} = \mu_i.$$

The proof of the second inequality $\mu_i \geq \lambda_{i+1}$ is obtained by a similar argument applied to $-A$. \square

Since any principal submatrix of a Hermitian matrix also is Hermitian, this theorem can be used recursively. Hence, if B is a principal submatrix of A of order $n - k$, then it holds that

$$\lambda_i \geq \mu_i \geq \lambda_{i+k}, \quad i = 1 : n - k. \quad (10.2.25)$$

The interlacing property holds more generally. Let $U = (U_1 \ U_2)$ be a unitary matrix, where $U_1 \in \mathbf{C}^{n \times (n-k)}$. Then $B = U_1^H A U_1$ is called a section of A , and the eigenvalues μ_i of B satisfy (10.2.25).

It is sometimes desirable to determine the eigenvalues of a diagonal matrix modified by a symmetric matrix of rank one. For an application, see Section 10.6.1.

Theorem 10.2.12.

Let $D = \text{diag}(d_i)$ and $z = (z_1, \dots, z_n)^T$. If $\lambda \neq d_i$, $i = 1 : n$, then the eigenvalues of $D + \mu zz^T$ are the roots of the **secular equation**

$$\psi(\lambda) = 1 + \mu \sum_{i=1}^n \frac{z_i^2}{d_i - \lambda} = 0. \quad (10.2.26)$$

The eigenvalues λ_i interlace the elements d_i so that if $\mu \geq 0$ then

$$d_1 \leq \lambda_1 \leq d_2 \leq \lambda_2 \leq \dots \leq d_n \leq \lambda_n \leq d_n + \mu \|z\|_2^2. \quad (10.2.27)$$

The eigenvector x_i corresponding to λ_i satisfies $(D - \lambda_i)x_i + \mu z(z^T x_i) = 0$ and hence if $\lambda_i \neq d_j$, $j = 1 : n$,

$$x_i = \frac{(D - \lambda_i)^{-1}z}{\|(D - \lambda_i)^{-1}z\|_2},$$

is a unit eigenvector.

Proof. By the assumption the matrix $(D - \lambda I)$ is nonsingular, and hence

$$\det(D + \mu zz^T - \lambda I) = \det((D - \lambda I)) \det(I + (D - \lambda I)^{-1}\mu zz^T)).$$

Since $\det(D - \lambda I) \neq 0$ we conclude, using the identity $\det(I + xy^T) = 1 + y^T x$, that the eigenvalues satisfy

$$\det(I + (D - \lambda I)^{-1}\mu zz^T)) = 0,$$

which gives (10.2.26). The interlacing property (10.2.27) follows from Fischer's Theorem 10.2.9. \square

We now consider residual error bounds for Hermitian matrices. For this case the backward perturbation result of Theorem 10.2.6 is not satisfactory. We now show that the perturbation E can be chosen to be Hermitian. This allows us to use the powerful results shown previously.

Theorem 10.2.13.

Let A be a Hermitian matrix, x a given unit vector, and $\mu = x^H A x$ the Rayleigh quotient. Then (μ, x) is an exact eigenpair of the Hermitian matrix $\tilde{A} = A + E$, where

$$E = -(rx^H + xr^H), \quad \|E\|_2 = \|r\|_2. \quad (10.2.28)$$

Proof. Note that the choice of μ makes r orthogonal to x . It follows that $Ex = -r$. Further, $\|E\|_2^2$ 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 $\|r\|_2^2$. The other eigenvalues are zero and hence $\|E\|_2 = \|r\|_2$. \square

This result shows that each interval

$$[\mu_i - \|r_i\|_2, \mu_i + \|r_i\|_2], \quad i = 1 : n,$$

contains an eigenvalue of A . When several such intervals overlap, it is possible that two or more different μ_i are approximations to a single eigenvalue.

For a Hermitian matrix A , the Rayleigh quotient $\rho_A(x)$ may be a far more accurate approximate eigenvalue than x is an approximate eigenvector. The gradient of $\frac{1}{2}\rho_A(x)$ equals

$$\frac{1}{2}\nabla\rho_A(x) = \frac{Ax}{x^H x} - \frac{x^H Ax}{(x^H x)^2}x = \frac{1}{x^H x}(Ax - \rho x).$$

Hence the Rayleigh quotient $\rho_A(x)$ 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 , the Rayleigh quotient approximates the corresponding eigenvalue to precision ϵ^2 .

Theorem 10.2.14.

Let the Hermitian matrix A have eigenvalues $\lambda_1, \dots, \lambda_n$ and orthonormal eigenvectors x_1, \dots, x_n . If the vector $x = \sum_{i=1}^n \xi_i x_i$, satisfies

$$\|x - \xi_1 x_1\|_2 \leq \epsilon \|x\|_2. \quad (10.2.29)$$

then

$$|\rho_A(x) - \lambda_1| \leq 2\epsilon^2 \|A\|_2. \quad (10.2.30)$$

Proof. Writing $Ax = \sum_{i=1}^n \xi_i \lambda_i x_i$, the Rayleigh quotient becomes

$$\rho_A(x) = \sum_{i=1}^n |\xi_i|^2 \lambda_i / \sum_{i=1}^n |\xi_i|^2 = \lambda_1 + \sum_{i=2}^n |\xi_i|^2 (\lambda_i - \lambda_1) / \sum_{i=1}^n |\xi_i|^2.$$

Using (10.2.29) we get $|\rho_A(x) - \lambda_1| \leq \max_i |\lambda_i - \lambda_1| \epsilon^2$. Since the matrix A is Hermitian, we have $|\lambda_i| \leq \sigma_1(A) = \|A\|_2$, $i = 1 : n$, and the theorem follows. \square

Sharper error bounds can be obtained if $\sigma = \rho_A(v)$ is well separated from all eigenvalues except λ . We first show a lemma.

Lemma 10.2.15.

Let \tilde{x} be an approximate eigenvector of unit norm of a Hermitian matrix A and $\tilde{\rho} = \tilde{x}^H A \tilde{x}$ the corresponding Rayleigh quotient. If the interval $[\alpha, \beta]$ contains $\tilde{\rho}$ but no eigenvalue of A , then

$$(\beta - \tilde{\rho})(\tilde{\rho} - \alpha) \leq \|r\|_2, \quad r = A\tilde{x} - \tilde{\rho}\tilde{x}. \quad (10.2.31)$$

Proof. We can write

$$(A - \alpha I)\tilde{x} = r + (\tilde{\rho} - \alpha)\tilde{x}, \quad (A - \beta I)\tilde{x} = r + (\tilde{\rho} - \beta)\tilde{x}.$$

where $r = (A\tilde{x} - \tilde{\rho}x)$ is the residual vector. Using the observation that r is orthogonal to \tilde{x} , we have

$$\tilde{x}^H(A - \alpha I)^H(A - \beta I)\tilde{x}\|r\|^2 + (\tilde{\rho} - \alpha)(\tilde{\rho} - \beta).$$

Expanding \tilde{x} in the orthogonal eigenvectors of A , $\tilde{x} = \sum_{i=1}^n \xi_i u_i$, the left hand side can be written

$$\tilde{x}^H(A - \alpha I)(A - \beta I)\tilde{x} = \sum_{i=1}^n \xi_i^2(\lambda_i - \alpha)(\lambda_i - \beta).$$

From the assumption on the interval $[a, b]$ it follows that each term in the above sum is positive. Therefore $\|r\|^2 + (\tilde{\rho} - \alpha)(\tilde{\rho} - \beta) \geq 0$, which is the desired result. \square

This lemma can be used to prove 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 10.2.16.

Let A be a Hermitian matrix with eigenvalues λ_i , $i = 1 : n$. Let \tilde{x} be an approximate eigenvector of unit norm and $\tilde{\rho} = \tilde{x}^H A \tilde{x}$ its Rayleigh quotient. Assume that λ is the eigenvalue closest to $\tilde{\rho}$ and δ the distance from $\tilde{\rho}$ to the rest of the spectrum

$$\text{gap}(\tilde{\rho}) = \min_i \{|\lambda_i - \tilde{\rho}| \mid \lambda_i \neq \lambda\}. \quad (10.2.32)$$

Then

$$|\tilde{\rho} - \lambda| \leq \|r\|_2^2 / \text{gap}(\tilde{\rho}). \quad (10.2.33)$$

Further, if x is an eigenvector of A associated with λ we have

$$\sin \theta(\tilde{x}, x) \leq \|r\|_2 / \text{gap}(\tilde{\rho}). \quad (10.2.34)$$

Proof. Note that by the assumption one of the two intervals $[\tilde{\rho} - \delta, \tilde{\rho}]$, $[\tilde{\rho}, \tilde{\rho} + \delta]$ contains no eigenvalue of A and use Lemma 10.2.15. For the proof of the result for the eigenvector, see Saad [516, Thm. 3.9]. \square

Example 10.2.5.

With $x = (1, 0)^T$ and $A = \begin{pmatrix} 1 & \epsilon \\ \epsilon & 0 \end{pmatrix}$, we get $\tilde{\rho} = 1$, and

$$Ax - x\tilde{\rho} = \begin{pmatrix} 0 \\ \epsilon \end{pmatrix}.$$

From Theorem 10.2.16 we get $|\lambda - 1| \leq \epsilon$, whereas Theorem 10.2.16 gives the improved bound $|\lambda - 1| \leq \epsilon^2/(1 - \epsilon^2)$.

Often $\text{gap}(\tilde{\rho})$ is not known and the bounds in Theorem 10.2.16 are only theoretical. However, by the method of spectrum slicing (see Section 10.6.2) an interval around a given value ρ can be determined which contains no eigenvalues of A .

The residual bounds for the Hermitian eigenvalue problem can be applied also to the singular vectors and singular values of a matrix A . It is no restriction to assume that $A \in \mathbf{C}^{n \times n}$ is square, since we can adjoin zero rows or columns to A . If $A = U\Sigma V^H$ then using Theorem 10.5.2, we have

$$Cx = \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \begin{pmatrix} u \\ \pm v \end{pmatrix} = \pm\sigma \begin{pmatrix} u \\ \pm v \end{pmatrix}. \quad (10.2.35)$$

This relates the singular vectors u and v and singular value σ to the eigenvectors and eigenvalues of the real symmetric matrix C . If u, v are unit vectors, then the Rayleigh quotient for the singular value problem of A is

$$\rho_A(u, v) = \frac{1}{\sqrt{2}}(u^H, \pm v^H) \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} u \\ \pm v \end{pmatrix} = u^H A v \geq 0, \quad (10.2.36)$$

where the sign of v is chosen to give a nonnegative value of $\rho_A(u, v)$. From Theorem 10.2.13 we obtain the following residual error bound.

Theorem 10.2.17. *For any scalar α and unit vectors u, v there is a singular value σ of A such that*

$$|\sigma - \alpha| \leq \frac{1}{\sqrt{2}} \left\| \begin{pmatrix} Av - u\alpha \\ A^H u - v\alpha \end{pmatrix} \right\|_2. \quad (10.2.37)$$

For fixed u, v this error bound is minimized by taking α equal to the Rayleigh quotient $\rho_A(u, v) = u^H A v$. defined by (10.2.36).

From Theorem 10.2.14 it follows that if u and v approximate a pair of left and right singular vectors with an error that is $O(\epsilon)$ then $u^H A v$ approximates the corresponding singular value with an error that is $O(\epsilon^2)$. The following improved error bound is an application of Theorem 10.2.16.

Theorem 10.2.18.

Let A have singular values σ_i , $i = 1 : n$. Let u and v be unit vectors, $\rho = u^H A v$ the corresponding Rayleigh quotient, and

$$\delta = \frac{1}{\sqrt{2}} \left\| \begin{pmatrix} Av - u\rho \\ A^H u - v\rho \end{pmatrix} \right\|_2$$

the residual norm. If σ_s is the closest singular value to ρ and $Au_s = \sigma_s v_s$, then

$$|\sigma_s - \rho| \leq \delta^2 / \text{gap}(\rho), \quad (10.2.38)$$

$$\max\{\sin \theta(u_s, u), \sin \theta(v_s, v)\} \leq \delta / \text{gap}(\rho). \quad (10.2.39)$$

where $\text{gap}(\rho) = \min_{i \neq s} |\sigma_i - \rho|$.

10.2.4 Invariant Subspaces and Newton-Based Methods

We now generalize some results shown in previous sections to invariant subspaces of a matrix A . In the proof of the Schur decomposition it was shown that when an

eigenvector was known the dimension of the eigenproblem can be reduced by one by a technique called deflation. Similarly, if an invariant subspace of dimension p is known, the dimension can be reduced by p .

Lemma 10.2.19.

Let $A, X \in \mathbf{C}^{n \times n}$, where $X = (X_1 \ X_2)$ is unitary. If the columns of $X_1 \in \mathbf{C}^{n \times p}$ span a right invariant subspace of dimension $p < n$, then

$$\begin{pmatrix} X_1^H \\ X_2^H \end{pmatrix} A (X_1 \ X_2) = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}. \quad (10.2.40)$$

where $A_{11} \in \mathbf{C}^{p \times p}$. The remaining eigenvalues of A are equal to those of A_{22} . Further, $X_2^H A = A_{22} X_2^H$, i.e., X_2 spans a left invariant subspace of A .

Proof. If the columns of X_1 span a right invariant subspace then $AX_1 = X_1 A_{11}$, for some matrix $A_{11} \in \mathbf{C}^{p \times p}$. Then we have

$$X^H AX = X^J(AX_1, AX_2) = X^H(X_1 A_{11}, AX_2) = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

which proves the theorem. \square

The residual bounds given in Theorem 10.2.6 can also be generalized to invariant subspaces. Suppose $X_1 \in \mathbf{C}^{n \times p}$ is an invariant subspace of the matrix $A \in \mathbf{C}^{n \times n}$. Then, by Theorem 10.2.19 there is a unique matrix $B = A_{11} = X_1^H AX_1$ such that

$$R = AX_1 - X_1 B = 0.$$

The eigenvalues of B are a subset of the eigenvalues of A . Suppose now that X is only an approximation to an invariant subspace of A . The following theorem gives backward error bounds in terms of the norm of the residual

$$R = AX_1 - X_1 B.$$

Theorem 10.2.20.

Let $X_1 \in \mathbf{C}^{n \times p}$ have orthonormal columns and let $R = AX_1 - X_1 B$. Then it holds that

$$(A + E)X_1 = X_1 B, \quad E = -RX_1^H.$$

and $\|E\|_p = \|R\|_p$, $p = 2, F$. Hence, X_1 is an exact invariant subspace of $A?E_1$. If A is Hermitian and

$$B = X_1^H AX_1, \quad E = -(RX_1^H + X^H R),$$

where E is Hermitian, $\|E\|_2 = \|R\|_2$, and $\|E\|_F = \sqrt{2}\|R\|_F$.

For a matrix $X_1 \in \mathbf{C}^{n \times p}$ of full column rank we define the **matrix Rayleigh quotient** by

$$B = (X_1^H X_1)^{-1} X_1^H A X_1. \quad (10.2.41)$$

If $X_1^H X_1 = I_p$ this simplifies to $R = X_1^H A X_1$. For $p = 1$ we know that the residual norm $\|R\|_2$ is minimized when B equals the scalar Rayleigh quotient. The following theorem, which generalizes Theorem 10.2.6, shows that the matrix Rayleigh quotient has similar optimal properties for $p > 1$.

Theorem 10.2.21.

Let $A \in \mathbf{C}^{n \times n}$ and $X_1 \in \mathbf{C}^{n \times p}$ with orthonormal columns be given. Let $X = (X_1 \ X_2)$ be unitary. Then any unitarily invariant norm of the residuals matrices

$$R = AX_1 - X_1 B, \quad S = X_1^H A - BX_1^H$$

is minimized when B equals the matrix Rayleigh quotient $B = X_1^H A X_1$. The minima equals

$$\|R\| = \|X_2^H A X_1\|, \quad \|S\| = \|X_1^H A X_2\|.$$

Proof. Since X is unitary, we have

$$\|R\| = \|(X_1 \ X_2)^H R\| = \left\| \begin{pmatrix} X_1^H A X_1 - B \\ X_2^H A X_1 \end{pmatrix} \right\|.$$

Using 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 nonsymmetric 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

$$B = (Y_1^H X_1)^{-1} Y_1^H A X_1. \quad (10.2.42)$$

is a Rayleigh quotient of A . If $Y_1^H X_1 = I$, this simplifies to $B = Y_1^H A X_1$.

It is important to realize that the residual norm and matrix Rayleigh quotient depend only on the subspace spanned by X_1 and not on the particular orthogonal basis X_1 chosen. Let $\tilde{X}_1 = X_1 P$, where $P \in \mathbf{C}^{p \times p}$ is unitary. Then

$$\tilde{B} = P^H (X_1^H A X_1) P, \quad \tilde{R} = A X_1 P - X_1 (P P^H) B P = R P.$$

It follows that $\|\tilde{R}\|_2 = \|R\|_2$ and the residual matrix \tilde{B} is similar to B .

Methods based on Newton's method with quadratic or even cubic rate of convergence can be used to improve approximations to invariant subspaces. Such methods are closely related to perturbation theory for invariant subspaces and we start with a study of this topic.

To any subset of eigenvalues of A there is an invariant subspace of A . However, this subspace may not be unique unless it contains all eigenvectors corresponding to multiple eigenvalues in the subset. In the following we will assume that this is the case. Then the Sylvester equation $A_{11}Q - QA_{22} = -A_{12}$ has a solution for which (see Lemma 10.1.18)

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} I_k & Q \\ 0 & I_{n-k} \end{pmatrix} = \begin{pmatrix} I_k & Q \\ 0 & I_{n-k} \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix}.$$

Equating the first block columns shows that $X_1 = \begin{pmatrix} I_k \\ 0 \end{pmatrix}$ span a right invariant subspace of the block A_{11} . Equating the last block row shows that the rows of $Y_2^H = (0 \quad I_{n-k})$ span a left invariant subspace of A_{22} .

Similarly, equating the first block row and last block column in

$$\begin{pmatrix} I_k & -Q \\ 0 & I_{n-k} \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} I_k & -Q \\ 0 & I_{n-k} \end{pmatrix}.$$

we find that $Y_1^H = (I_k \quad -Q)$ and $X_2 = \begin{pmatrix} Q \\ I_{n-k} \end{pmatrix}$ give bases for the right invariant subspaces of A_{11} and left invariant subspace of A_{22} , respectively. It can be verified that $Y_1^H X_1 = I_k$ and $Y_2^H X_2 = I_{n-k}$. The results are summarized in the following theorem.

Theorem 10.2.22.

Assume that the diagonal blocks A_{11} and A_{22} of the block triangular matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \quad (10.2.43)$$

have no eigenvalues in common. Let Q satisfy the Sylvester equation $A_{11}Q - QA_{22} = -A_{12}$. Then

$$P_1 = X_1 Y_1^H = \begin{pmatrix} I_k & -Q \\ 0 & 0 \end{pmatrix}, \quad P_2 = X_2 Y_2^H = \begin{pmatrix} 0 & Q \\ 0 & I_{n-k} \end{pmatrix}. \quad (10.2.44)$$

are the **spectral projectors** corresponding to the blocks A_{11} and A_{22} . Further

$$A = X_1 A_{11} Y_1^H + X_2 A_{22} Y_2^H, \quad (10.2.45)$$

which is the **spectral decomposition** of A .

We assume in the following that the matrix $A \in \mathbf{C}^{n \times n}$ has the block triangular form (10.2.43), where $A_{11} \in \mathbf{C}^{k \times k}$. This form can always be achieved by a unitary similarity transformation (the Schur decomposition), and will not change the sensitivity of the eigenvalues and eigenvectors.

We recall (Theorem 10.2.22) that the spectral projector belonging to A_{11} is

$$P = \begin{pmatrix} I_k & -Q \\ 0 & 0 \end{pmatrix}$$

where Q satisfies the Sylvester equation

$$A_{11}Q - QA_{22} = -A_{12}.$$

It follows that $\|P\|_2 = (1 + \|Q\|_2^2)^{1/2}$, which will play an important role in several bounds. One can attempt to derive perturbation bounds in terms of the minimum distance between $\lambda(A_{11})$ and $\lambda(A_{22})$, i.e.,

$$\min |\lambda(A_{11}) - \lambda(A_{22})|$$

However, since the eigenvalues of A_{11} and A_{22} may be sensitive to small perturbations this number can vary greatly. Instead we will use a more robust measure of the separation between $\lambda(A_{11})$ and $\lambda(A_{22})$ introduced by Stewart in 1973.

Definition 10.2.23.

The separation $\text{sep}(A_{11}, A_{22})$ of the matrices A_{11} and A_{22} is the smallest singular value of the linear map which takes X to $A_{11}X - XA_{22}$, i.e.,

$$\begin{aligned}\text{sep}(A_{11}, A_{22}) &= \min_{X \neq 0} \frac{\|A_{11}X - XA_{22}\|_F}{\|X\|_F}. \\ &= \sigma_{\min}(I_n \otimes A_{11} - A_{22}^T \otimes I_m).\end{aligned}\quad (10.2.46)$$

An important property of $\text{sep}(A_{11}, A_{22})$ is

$$\begin{aligned}\text{sep}(A, B) - \|E\|_2 - \|F\|_2 &\leq \text{sep}(A + E, B + F) \\ &\leq \text{sep}(A, B) + \|E\|_2 + \|F\|_2;\end{aligned}\quad (10.2.47)$$

see Stewart [542]. Further, $\text{sep}(A_{11}, A_{22})$ is zero if and only if A_{11} and A_{22} have a common eigenvalue. It is small when there is a small perturbation of A_{11} and A_{22} , which make them have a common eigenvalue. If A_{11} and A_{22} are both normal the separation is just the minimum distance between an eigenvalues of A_{11} and A_{22} .

The quantity $\text{sep}(A_{11}, A_{22})$ can be estimated using Hager's 1-norm estimator; see Algorithm 7.5.3. This estimates the norm $\|M\|_1$ of a linear operator M provided that given vectors x and y , we are able to compute Mx and M^Ty cheaply. In this case computing Mx is equivalent to solve the Sylvester equation with an arbitrary right hand side. Computing M^Ty 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 done in $O(n^3)$ operations.

Example 10.2.6 (Varah [593]).

We consider two different measures of sensitivity for the simple eigenvalue λ of the matrix

$$A = \begin{pmatrix} \lambda & c^H \\ 0 & B \end{pmatrix}$$

To apply Wilkinson's measure, we note that the left and right eigenvectors corresponding to λ are $y^H = (1 \ x^H)$ and $x = (1 \ 0)^T$, where x is the solution to $(B^T - \lambda I)x = c$. Hence, the spectral projector is $P_\lambda = \begin{pmatrix} 1 & x \\ 0 & 0 \end{pmatrix}$ and its norm $s_\lambda = (1 + \|x\|_2^2)^{1/2}$. On the other hand, we have

$$\text{sep}(\lambda, B) = \min_{\|x\|_2=1} \|B - \lambda I\|_2 = \sigma_{\min}(B - \lambda I).$$

Notice that s_λ depends on c but $\text{sep}(\lambda, B)$ does not. The two quantities are related by

$$s_\lambda^2 - 1 = \|x\|_2^2 \leq \|(B - \lambda I)^{-1}\|_2^2 \|c\|_2 = \frac{\|c\|_2}{(\text{sep}(\lambda, B))^2}.$$

Hence $\text{sep}(\lambda, B)$ gives the smallest possible s_λ over all vectors c of norm one.

Theorem 10.2.24 (Stewart [543]).

Let $A \in \mathbf{C}^{n \times n}$ have the block triangular form

$$A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

and consider the eigenvalues of the perturbed matrix $A + E$. Let $P = X_1 Y_1^H$ be the spectral projector of A_{11} (see (10.2.44)). Then as long as

$$\|E\|_F < \frac{\text{sep}(A_{11}, A_{22})}{4\|P\|_2},$$

to A_{11} , the perturbed eigenvalues belonging to A_{11} will remain disjoint from those of A_{22} .

Although this bound can greatly underestimate the smallest perturbation needed to make eigenvalues coalesce it is exact for some matrices.

To study the sensitivity of an invariant subspace under perturbations we need to measure the distance between subspaces. In Definition 8.1.23 the distance between two subspaces \mathcal{X} and \mathcal{Y} of the same dimension k was defined as $\sin \theta_k$, where θ_k is the largest principal angle between the subspaces. If X and Y are orthonormal matrices such that $\mathcal{X} = \mathcal{R}(X)$ and $\mathcal{Y} = \mathcal{R}(Y)$, then

$$\theta(\mathcal{X}, \mathcal{Y}) = \arccos \sigma_{\min}(X^H Y). \quad (10.2.48)$$

When simultaneous iteration is used to compute an invariant subspace, it will display only linear convergence. Methods based on Newton's method with quadratic or cubic rate of convergence can be developed. Many of the proposed methods lead to a Riccati equation of the form $AR - RB = C + RDR$, where R represents the error in the initial estimate. We want to solve the equation

$$AX - XB = 0 \quad (10.2.49)$$

where $X \in \mathbf{R}^{n \times p}$ is a matrix whose column span the desired invariant subspace. This is np equations for $np + p^2$ unknowns in X and B . We reduce the number of unknowns by holding p rows of X fixed. If the initial X has orthonormal columns, by an orthogonal transformation we may assume that in the new basis X has the form

$$X = \begin{pmatrix} I \\ R \end{pmatrix}.$$

Partitioning the matrix A conformally, we can rewrite (10.2.49) as

$$\begin{pmatrix} A_{11} + A_{12}R \\ A_{21} + A_{22}R \end{pmatrix} = \begin{pmatrix} B \\ RB \end{pmatrix}.$$

Substituting the first equation for B into the second now yields the Riccati equation

$$A_{22}R - RA_{11} = -A_{21} + RA_{12}R. \quad (10.2.50)$$

A generalization of RQI to the computation of an invariant subspace goes as follows:

Algorithm 10.1. *Subspace Rayleigh Quotient Iteration.*

Let $X_0 \in \mathbf{R}^{n \times p}$ be an orthonormal matrix and for $k = 0, 1, 2, \dots$, repeat

1. Compute the Rayleigh quotient $R_k = X_k^T A X_k$.
2. Compute the solution Z to the Sylvester equation

$$AZ - ZR_k = X_k \quad (10.2.51)$$

3. Take X_{k+1} to be the Q factor in the QR factorization of Z .

Optimization problems to minimize a function $F(Y)$, where $Y \in \mathbf{R}^{n \times p}$ satisfies the constraint $Y^T Y = I$, occurs quite often in linear algebra. This is a problem defined on the set of $n \times p$ orthonormal matrices. This constraint surface is called the **Stiefel manifold**, after Eduard Stiefel, who considered its topology in his thesis from 1932.

A related optimization problem is where Y satisfies the further homogeneity condition

$$F(Y) = F(YQ) \quad \forall \quad Q \in \mathbf{R}^{p \times p}, \quad Q^T Q = I.$$

The set of p -dimensional subspaces in \mathbf{R}^n is called the **Grassmann manifold**.⁴⁷ Several recent iteration methods for computing invariant subspaces rely on the quotient geometry of Grassmann manifolds.

Review Questions

- 2.1** State Geršgorin's Theorem, and discuss how it can be sharpened.
- 2.2** What can you say about the sensitivity to perturbations of eigenvalues and eigenvectors of a Hermitian matrix?
- 2.3** Suppose that $(\bar{\lambda}, \bar{x})$ is an approximate eigenpair of A . How can you compute an upper bound of $\|E\|_2$, where E is such that $(\bar{\lambda}, \bar{x})$ is an exact eigenpair of $(A + E)$?

⁴⁷Hermann Günther Grassmann (1809–1877) German mathematician, born in Stettin, studied theology, languages, and philosophy at University of Berlin. Later he undertook mathematical research on his own and took up teaching at the Gymnasium in Stettin. In 1844 he published a textbook in which the symbols representing geometric entities such as points, lines, and planes were manipulated using certain rules. This was highly original at the time and 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.

2.4 (a) Tell the minimax and maximin properties of the eigenvalues (of what kind of matrices?), and the related properties of the singular values (of what kind of matrices?).

(b) Show how the theorems in (a) can be used for deriving an interlacing property for the eigenvalues of a matrix in $\mathbf{R}^{n \times n}$ (of what kind?) and the eigenvalues of its principal submatrix in $\mathbf{R}^{(n-1) \times (n-1)}$.

Problems

2.1 An important problem is to decide if all the eigenvalues of a matrix A have negative real part. Such a matrix is called **stable**. Show that if

$$\operatorname{Re}(a_{ii}) + r_i \leq 0, \quad \forall i,$$

and $\operatorname{Re}(a_{ii}) + r_i < 0$ for at least one i , then the matrix A is stable if A is irreducible.

2.2 Suppose that the matrix A is real, and all Geršgorin disks of A are distinct. Show that from Theorem 10.2.2 it follows that all eigenvalues of A are real.

2.3 Let $A \in \mathbf{C}^{n \times n}$, and assume that $\epsilon = \max_{i \neq j} |a_{ij}|$ is small. Choose the diagonal matrix $D = \operatorname{diag}(\mu, 1, \dots, 1)$ so that the first Geršgorin disk of DAD^{-1} is as small as possible, without overlapping the other disks. Show that if the diagonal elements of A are distinct then

$$\mu = \frac{\epsilon}{\delta} + O(\epsilon^2), \quad \delta = \min_{i \neq 1} |a_{ii} - a_{11}|,$$

and hence the first Geršgorin disk is given by

$$|\lambda - a_{11}| \leq r_1, \quad r_1 \leq (n-1)\epsilon^2/\delta + O(\epsilon^3).$$

2.4 Compute the eigenvalues of B and A , where

$$B = \begin{pmatrix} 0 & \epsilon \\ \epsilon & 0 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Show that they interlace.

2.5 Use a suitable diagonal similarity and Geršgorin's theorem to show that the eigenvalues of the tridiagonal matrix

$$T = \begin{pmatrix} a & b_2 & & & \\ c_2 & a & b_3 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & a & b_n \\ & & c_n & a & a \end{pmatrix}.$$

satisfy the inequality

$$|\lambda - a| < 2\sqrt{\max_i |b_i| \max_i |c_i|}.$$

2.6 Let A and B be square Hermitian matrices and

$$H = \begin{pmatrix} A & C \\ C^H & B \end{pmatrix}.$$

Show that for every eigenvalue $\lambda(B)$ of B there is an eigenvalue $\lambda(H)$ of H such that

$$|\lambda(H) - \lambda(B)| \leq (\|C^H C\|_2)^{1/2}.$$

Hint: Use the estimate in Theorem 10.2.13.

10.3 The Power Method and Its Generalizations

10.3.1 The Simple Power Method

One of the oldest methods for computing eigenvalues and eigenvectors of a matrix is the **power method**. Until the 1950s this method was combined with deflation was the method of choice for finding a few of the eigenvalues of a nonsymmetric matrix. Although the power method in its basic form is no longer competitive for most problems, it is closely related to many other currently used algorithms. With a small modification it is also used in Google's Pagerank algorithm to compute an eigenvector of a matrix of order 2.7 billion; see [440] and [81].

Given a matrix $A \in \mathbf{C}^{n \times n}$ and a starting vector $q_0 \neq 0$ the power method forms the sequence of vectors q, Aq, A^2q, A^3q, \dots , using the recursion

$$q_k = Aq_{k-1}, \quad k = 1, 2, 3, \dots$$

In practice, the vectors q_k have to be normalized in order to avoid overflow or underflow. Hence, assume that $\|q_0\|_2 = 1$, and modify the initial recursion as follows:

$$\hat{q}_k = Aq_{k-1}, \quad \mu_k = \|\hat{q}_k\|_2, \quad q_k = \hat{q}_k / \mu_k, \quad k = 1, 2, \dots \quad (10.3.1)$$

Assume that the eigenvalues are ordered so that

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|,$$

i.e., λ_1 is a unique eigenvalue of maximum magnitude, An approximation to λ_1 is obtained by from the Rayleigh quotients

$$\rho(q_k) = q_k^H A q_k = q_k^H \hat{q}_{k+1}. \quad (10.3.2)$$

An attractive feature of the power method is that it suffices to be able to form the matrix times vector product Aq for a given vector q . The matrix powers A^k are never computed and the matrix A is not modified. A more systematic treatment of methods for large-scale eigenvalue problems will be given in Section 11.6. Since the power method also directly or indirectly serves as the basis of many algorithms for dense eigenvalue problems it is treated here.

To simplify the analysis, we assume that the matrix A has a set of linearly independent eigenvectors x_1, x_2, \dots, x_n associated with the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$. Then the initial vector q_0 can be expanded along the eigenvectors

$$q_0 = \sum_{j=1}^n \alpha_j x_j, \quad \alpha_1 \neq 0.$$

It follows that

$$\begin{aligned} A^k q_0 &= \sum_{j=1}^n \lambda_j^k \alpha_j x_j = \lambda_1^k \left(\alpha_1 x_1 + \sum_{j=2}^n \left(\frac{\lambda_j}{\lambda_1} \right)^k \alpha_j x_j \right) \\ &= \lambda_1^k \alpha_1 x_1 + O \left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right). \quad k = 1, 2, \dots \end{aligned} \quad (10.3.3)$$

It follows that q_k converges to a normalized eigenvector x_1 . The *rate of convergence is linear and equals $|\lambda_2|/|\lambda_1|$* . One can show that this result holds also when A is not diagonalizable by writing q_0 as a linear combination of the vectors associated with the Schur decomposition of A , see Theorem 10.1.13.

For a Hermitian matrix A the eigenvalues are real and the eigenvectors can be chosen so that $x_i^H x_j = 0$, $i \neq j$. Using (10.3.3) it follows that the Rayleigh quotients converge twice as fast,

$$\lambda_1 = \rho(q_k) + O \left(\left| \frac{\lambda_2}{\lambda_1} \right|^{2k} \right). \quad (10.3.4)$$

Example 10.3.1. The eigenvalues of the symmetric matrix

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 4 \end{pmatrix}$$

are $(4.732051, 3, 1.267949)$, correct to 6 decimals. If we take $q_0 = (1, 1, 1)^T$ then we obtain the Rayleigh quotients ρ_k and errors $e_k = \lambda_1 - \rho_k$ are given in the table below. The ratios of successive errors converge to $(\lambda_2/\lambda_1)^2 = 0.4019$.

k	ρ_k	e_k	e_k/e_{k-1}
1	4.333333	0.398718	
2	4.627119	0.104932	0.263
3	4.694118	0.037933	0.361
4	4.717023	0.015027	0.396
5	4.729620	0.006041	0.402

Convergence of the power method can be shown only for *almost all starting vectors* since 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 Aq_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 $\lambda_1 = \lambda_2 = \dots = \lambda_r$, 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.

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 the two complex conjugate eigenvectors will tend to belong to the subspace spanned by two successive iterates q_k and q_{k+1} . No complex arithmetic is necessary and q_0 can be chosen as a real vector; see Fröberg [221, p. 194]. A modification for the case when there are two dominant eigenvalues of opposite sign, $\lambda_1 = -\lambda_2$ is given in Problem 10.3.2.

A simple modification of the power method is to apply the power method to the matrix $(A - \mu I)$, 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. For convergence to x_1 the shift $\mu = \frac{1}{2}(\lambda_2 + \lambda_n)$ is optimal. The rate of convergence is then governed by

$$\frac{\lambda_2 - \mu}{\lambda_1 - \mu} = \frac{\lambda_2 - \lambda_n}{2\lambda_1 - (\lambda_2 + \lambda_n)}.$$

Similarly, for convergence to x_n the shift $\mu = \frac{1}{2}(\lambda_1 + \lambda_{n-1})$ is optimal. Unfortunately the improvement in convergence using this device is often quite small.

So far we have neglected the effect of rounding errors in the power method. These errors will limit the accuracy which can be attained. If we include rounding errors we will in (10.3.1) compute

$$\mu_k q_k = Aq_{k-1} + f_k,$$

where f_k is a small error vector. If Aq_{k-1} is computed using floating point arithmetic, then by (7.1.83) we have

$$fl(Aq_{k-1}) = (A + F_k)q_{k-1}, \quad |F_k| < \gamma_n |A|.$$

We cannot guarantee any progress after having reached vector q_k , which is an exact eigenvector of some matrix $(A + G)$.

In the power method we compute the sequence of vectors u, Au, A^2u, \dots one by one. Each vectors is overwritten the previous one. This saves storage but is wasteful in other respects. For example, even for a 2 by 2 matrix the power method will in general needs an infinite number of iterations. However, unless $Au = \lambda u$, the subspace $\text{span}[u, Au]$ spans \mathbf{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 $\text{span}[u, Au]$. This insight leads to the Lanczos and Arnoldi's methods, which will be described in Chapter 10.

Many methods for solving the eigenvalue problem developed by Krylov and others in the 1930's and 40's aimed at bringing the characteristic equation into polynomial form. This is generally a bad idea, since the roots of a polynomial can be extremely sensitive to small perturbations in the coefficients.

10.3.2 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 $A \in \mathbf{C}^{n \times n}$. How can we proceed if we want to compute further eigenvalues and their eigenvectors? An old technique for achieving this is to use what is known as deflation. By this we mean a method that forms a modified matrix A_1 such that the eigenvalue λ_1 is eliminated by shifting it to zero, but does not change the other eigenvalues. Such a matrix A could be constructed in a stable way by using a similarity transformation as indicated in Section 10.1; see (10.2.40). However, this has the drawback that when A is sparse the matrix A_1 will in general be dense.

Suppose an eigenpair (λ_1, x_1) , with $x_1^H x_1 = 1$, of a Hermitian matrix A is known. In a simple deflation method, due to Hotelling, the matrix A_1 is taken to be

$$A_1 = A - \lambda_1 x_1 x_1^H, \quad (10.3.5)$$

which is a rank one modification of A . Note that

$$A_1 = A - \lambda_1 x_1 x_1^H = (I - x_1 x_1^T)A = P_1 A,$$

where P_1 is an orthogonal projection. From the orthogonality of the eigenvectors x_i it follows that

$$A_1 x_i = Ax_i - \lambda_1 x_1 (x_1^T x_i) = \begin{cases} 0, & \text{if } i = 1; \\ \lambda_i x_i, & \text{if } i \neq 1. \end{cases}$$

This shows that the eigenvalues of A_1 are $0, \lambda_2, \dots, \lambda_n$ and the eigenvectors are unchanged. The power method can now be applied to A_1 to determine the dominating eigenvalue λ_2 of A_1 . An important practical point is that the matrix A_1 does not have to be formed explicitly. In the power method only products $y = A_1 x = (A - \lambda_1 x_1 x_1^H)x$ need to be computed, which can be performed as follows

$$A_1 x = Ax - \lambda_1 x_1 (x_1^T y).$$

This only requires that the matrix A and the vector x_1 are available.

Hotelling's deflation is a special case of a more general deflation scheme due to Wielandt.

Theorem 10.3.1 (Wielandt [608]).

Let the eigenvalues and right eigenvectors of $A \in \mathbf{C}^{n \times n}$ be λ_i, x_i , $i = 1 : n$, and set

$$A_1 = A - \sigma x_1 z^H, \quad (10.3.6)$$

where z is an arbitrary vector such that $z^H x_1 = 1$. Then the eigenvalues of A_1 are $\lambda_1 - \sigma, \lambda_2, \dots, \lambda_n$.

Proof. For $i \neq 1$ the left eigenvectors y_i satisfy

$$A_1^H y_i = (A^H - \bar{\sigma} z x_1^H) y_i = \begin{cases} (\lambda_1 - \sigma) y_i, & \text{if } i = 1; \\ \lambda_i y_i, & \text{if } i \neq 1. \end{cases}$$

because $x_1^H y_i = 0$, $i \neq 1$. \square

From the proof it is clear that the left eigenvectors are preserved in A_1 . To see what the right eigenvectors are, we seek a right eigenvector of the form $\hat{x}_i = x_i - \gamma_i x_1$. Then, since $z^H x_1 = 1$, we have

$$A_1 \hat{x}_i = (A - \sigma x_1 z^H)(x_i - \gamma_i x_1) = \lambda_i x_i - (\gamma_i \lambda_1 + \sigma(z^H x_i) + \sigma \gamma_i) x_1$$

For $i = 1$, taking $\gamma_1 = 0$, we see that the eigenvector x_1 is preserved. For $i \neq 1$, the vector \hat{x}_i is an eigenvector of A_1 associated with the eigenvalue λ_i if

$$\gamma_i = \frac{\sigma(z^H x_i)}{\sigma - (\lambda_1 - \lambda_i)}. \quad (10.3.7)$$

As in the symmetric case it is not necessary to form the matrix A_1 explicitly. The power method, as well as many other methods, only requires operations of the form $y = A_1 x$, which can be performed as

$$A_1 x = (A - \sigma x_1 z^H)x - Ax - \tau x_1, \quad \tau = \sigma(z^H x).$$

Therefore, it suffices to have the vectors x_1, 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 such repeated deflation has to be used with caution. Errors will accumulate which can be disastrous if the currently computed eigenvalue is badly conditioned; see Wilkinson [611, pp. 584–601].

There are many ways to choose the vector z . One of the most common is to take $z = y_1$, the left eigenvector and set $\sigma = \lambda_1$,

$$A_1 = A - \lambda_1 x_1 y_1^T,$$

which is Hotelling's deflation for non-Hermitian matrices. From the biorthogonality of the left and right eigenvectors we have $y_1^H x_i = 0$, $i \neq 1$. Therefore, by (10.3.7) $\gamma_i = 0$, $i = 1 : n$, and

$$A_1 x_i = Ax_i - \lambda_1 x_1 (y_1^H x_i) = \begin{cases} 0, & \text{if } i = 1; \\ \lambda_i x_i, & \text{if } i \neq 1. \end{cases}$$

This deflation has the advantage that both left and right eigenvectors are preserved. Another interesting choice is to take $z = x_1$ in (10.3.6) so that the modification is Hermitian. This choice has the useful property that it preserves the Schur vectors of A .

Theorem 10.3.2.

Let x_1 , $\|x_1\|_2 = 1$, be a right eigenvectors of $A \in \mathbf{C}^{n \times n}$ with the associated eigenvalue and λ_1 , and set

$$A_1 = A - \sigma x_1 x_1^H. \quad (10.3.8)$$

Then the eigenvalues of A_1 are $\lambda_1 - \sigma, \lambda_2, \dots, \lambda_n$ 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

$$A_1 Q = (A - \sigma x_1 x_1^H) Q = QU - \sigma x_1 e_1^H = Q(U - \sigma e_1 e_1^H)$$

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. The next Schur vector is then obtained by orthonormalizing \hat{x}_{k+1} against the 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 .10.1.16. Then in a step where a pair of complex conjugate eigenvalues are determined, two new columns will be added to Q_k .

10.3.3 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 eigenvalues of the matrix A satisfy $|\lambda_1| \geq \dots \geq |\lambda_{n-1}| > |\lambda_n|$. The eigenvalues of the inverse matrix A^{-1} are $1/\lambda_j$. If the power method is applied to A^{-1} ,

$$q_{k+1} = A^{-1} q_k, \quad k = 1, 2, 3, \dots,$$

q_k will converge to the eigenvector x_n corresponding to the eigenvalue of smallest magnitude λ_n . In practice, the iteration is accomplished by computing the LU factorization $A = LU$, and then solving

$$LU q_{k+1} = q_k, \quad k = 1, 2, 3, \dots,$$

by forward and backsubstitution. This is called **inverse iteration** and was introduced in 1944 by Wielandt [608]. By performing inverse iteration on a shifted matrix $(A - \mu I)$ with a suitably chosen shift, we can get convergence to any eigenvalue of the matrix.

Lemma 10.3.3. *Spectral Transformation*

Let μ be a chosen shift of the spectrum of a matrix A . Then the eigenvalues λ_i , $i = 1 : n$, of A are related to the eigenvalues θ_i , $i = 1 : n$ of the matrix $B = (A - \mu I)^{-1}$ by

$$\theta_i = \frac{1}{\lambda_i - \mu}, \quad \lambda_i = \mu + \frac{1}{\theta_i} \quad (10.3.9)$$

By the spectral transformation eigenvalues close to the shift μ are transformed into large and well separated eigenvalues of B . The power method can then be used to find the eigenvalues close to μ by using the power method to B . This iteration takes the form

$$(A - \mu I)\hat{q}_k = q_{k-1}, \quad q_k = \hat{q}_k / \|\hat{q}_k\|_2, \quad k = 1, 2, \dots \quad (10.3.10)$$

An approximation $\bar{\lambda}_i$ to the eigenvalue λ_i of A is obtained from the Rayleigh quotient

$$\frac{1}{\lambda_i - \mu} \approx q_{k-1}^T (A - \mu I)^{-1} q_{k-1} = q_{k-1}^T \hat{q}_k,$$

where \hat{q}_k satisfies $(A - \mu I)\hat{q}_k = q_{k-1}$. Thus,

$$\bar{\lambda}_i = \mu + \frac{1}{q_{k-1}^T \hat{q}_k}. \quad (10.3.11)$$

An a posteriori bound for the error in the approximate eigenvalue $\bar{\lambda}_i$ of A can be obtained from the residual corresponding to $(\bar{\lambda}_i, \hat{q}_k)$, which equals

$$r_k = A\hat{q}_k - \left(\mu + 1/(q_{k-1}^T \hat{q}_k) \right) \hat{q}_k = q_{k-1} - \hat{q}_k / (q_{k-1}^T \hat{q}_k).$$

Then, by Theorem 10.2.6, $(\bar{\lambda}_i, \hat{q}_k)$ is an exact eigenpair of a matrix $A + E$, where $\|E\|_2 \leq \|r_k\|_2 / \|\hat{q}_k\|_2$. If A is real symmetric then the error in the approximative eigenvalue $\hat{\lambda}_i$ of A is bounded by $\|r_k\|_2 / \|\hat{q}_k\|_2$.

There is no need to explicitly invert $A - \mu I$. In the symmetric case we compute a factorization

$$A - \mu I = LDL^T,$$

where L is (block) lower triangular and D is (block) diagonal (see Section 7.3.4). Note that even if A is positive definite this property is in general not shared by the shifted matrix $(A - \mu I)$. In the unsymmetric case we compute using partial pivoting the factorization

$$P(A - \mu I) = LU.$$

Each step of (10.3.10) then only requires the solution of two triangular systems, which for a dense matrix A is no more costly than one step of the simple power method. However, if the matrix is sparse, the total number of nonzero elements in L and U may be much larger than in A .

Inverse iteration has been developed into a very effective method for computing the associated eigenvector to an eigenvalue λ_i , for which an accurate approximation

already is known. If the shift μ is chosen sufficiently close to an eigenvalue λ_i , 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 ,

$$|\lambda_i - \mu|^{-1} \gg |\lambda_j - \mu|^{-1}, \quad \lambda_i \neq \lambda_j. \quad (10.3.12)$$

This ensures that q_k will converge fast to the eigenvector x_i . If μ equals λ_i up to machine precision then $A - \mu I$ in (10.3.10) is numerically singular. Therefore, it may seem that inverse iteration was doomed to failure if μ was chosen too close to an eigenvalue. Fortunately a careful analysis shows that this is not the case!

If Gaussian elimination with partial pivoting is used the computed factorization of $(A - \mu I)$ will satisfy

$$P(A + E - \mu I) = \bar{L}\bar{U},$$

where $\|E\|_2/\|A\|_2 = f(n)O(u)$, and u is the unit roundoff and $f(n)$ a modest function of n . Since the rounding errors in the solution of the triangular systems usually are negligible the computed q_k will nearly satisfy

$$(A + E - \mu I)\hat{q}_k = q_{k-1}.$$

This shows that 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 we can apply the simple a posteriori error bound in Theorem 10.2.6 to inverse iteration. By (10.3.10) q_{k-1} is the residual vector corresponding to the approximate eigenpair (μ, \hat{q}_k) . Hence, where u is the unit roundoff, \hat{q}_k is a numerically acceptable eigenvector if

$$\|q_{k-1}\|_2/\|\hat{q}_k\|_2 \leq u\|A\|_2. \quad (10.3.13)$$

Example 10.3.2.

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 $q_0 = (1, 0)^T$ gives

$$A - \mu I = LU = \begin{pmatrix} 1 & 0 \\ 0.37009623 & 1 \end{pmatrix} \begin{pmatrix} 0.2702 & 1 \\ 0 & 0.0001038 \end{pmatrix}$$

and $\hat{q}_1 = 10^4(1.3202568, -0.3566334)^T$, $q_1 = (0.9653989, -0.2607777)^T$, which agrees with the correct eigenvector to more than four decimals. From the backward error bound it follows that 0.7298 and q_1 is an exact eigenpair to a matrix $A + E$, where $\|E\|_2 \leq 1/\|\hat{q}_1\|_2 = 0.73122 \cdot 10^{-4}$.

Inverse iteration is a useful algorithm for calculation of specified eigenvectors corresponding to well separated eigenvalues for dense matrices. Often a random

initial vector with elements from a uniform distribution in $[-1, 1]$ is used. In order to save work in the triangular factorizations the matrix is usually first reduced to Hessenberg or real tridiagonal form, by the methods that will be described in Section 10.4.3 and Section 10.5.1.

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 the nonsymmetric case the situation can be worse, in particular 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.

Example 10.3.3.

The matrix

$$A = \begin{pmatrix} 1 + \epsilon & 1 \\ \epsilon & 1 + \epsilon \end{pmatrix}$$

has eigenvalues $\lambda = (1 + \epsilon) \pm \sqrt{\epsilon}$. Assume that $|\epsilon| \approx u$, where u is the machine precision. Then the eigenpair $\lambda = 1$, $x = (1, 0)^T$ is a numerically acceptable eigenpair of A , since it is exact for the matrix $A + E$, where

$$E = -\begin{pmatrix} \epsilon & 0 \\ \epsilon & \epsilon \end{pmatrix}, \quad \|E\|_2 < \sqrt{3}u.$$

If we perform one step of inverse iteration starting from the acceptable eigenvector $q_0 = (1, 0)^T$ then we get

$$\hat{q}_1 = \frac{1}{1 - \epsilon} \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

No growth occurred and it can be shown that $(1, q_1)$ is not an acceptable eigenpair of A . If we carry out one more step of inverse iteration we again get an acceptable eigenvector!

Equation (10.2.13) 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 Section 10.2. 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 Section 10.2.2. 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 $s(\lambda)$.

10.3.4 Rayleigh Quotient Iteration

So far we have used a fixed shift in the inverse power method. This is very efficient provided an accurate approximation to the eigenvalue is known. Otherwise it is possible to use a variable shift in each iteration. This is considerably more costly since a new factorization of the shifted matrix then has to be computed at

each iteration step. The gain is that instead of linear convergence we can achieve quadratic or even cubic convergence.

The previous analysis suggests choosing a shift equal to the Rayleigh quotient of the current eigenvector approximation. This leads to the Rayleigh quotient iteration:⁴⁸

Algorithm 10.2. *Rayleigh Quotient Iteration (RQI).*

Let q_0 , $\|q_0\|_2 = 1$, be a starting vector. For $k = 0, 1, 2, \dots$, do

1. Compute the Rayleigh quotient $\rho_k = q_k^T A q_k$.
 2. If $A - \rho_k I$ is singular, then solve $(A - \rho_k I)q_{k+1} = 0$ for a unit vector q_{k+1} and stop. Otherwise, solve
- $$(A - \rho_k I)z_{k+1} = q_k, \quad (10.3.14)$$
3. Normalize $q_{k+1} = z_{k+1}/\|z_{k+1}\|_2$. If $\|z_{k+1}\|$ is sufficiently large, then stop

In RQI a new triangular factorization must be computed of the matrix $A - \rho_k I$ for each iteration step. This makes RQI much more expensive than ordinary inverse iteration. However, if the matrix A is of Hessenberg (or tridiagonal) form the extra cost is small.

The convergence properties of Algorithm 10.3.4 have been extensively studied both for the symmetric and unsymmetric case. A key fact in the analysis is that *the norm of the residuals norms are monotonic*.

Theorem 10.3.4.

In the Rayleigh quotient iteration the residual norms are monotonic $\|r_{k+1}\|_2 \leq \|r_k\|_2$, and equality holds only if $\rho_{k+1} = \rho_k$ and x_k is an eigenvector of $(A - \rho_k I)^2$.

Proof. Using the minimum property of the Rayleigh quotient and that x_k is a multiple of $(A - \rho_k I)x_{k+1}$ we have

$$\begin{aligned} \|r_{k+1}\|_2 &\equiv \|(A - \rho_{k+1} I)x_{k+1}\|_2 \leq \|(A - \rho_k I)x_{k+1}\|_2 \\ &= \|x_k^H (A - \rho_k I)x_{k+1}\|_2. \end{aligned}$$

Here equality holds only if $\rho_{k+1} = \rho_k$. By Cauchy–Schwarz inequality it follows that

$$\|r_{k+1}\|_2 \leq \|(A - \rho_k I)x_k\|_2 \|x_{k+1}\|_2 = \|r_k\|_2.$$

Here equality holds in the first inequality only if r_k is a multiple of x_{k+1} , i.e. only if for some μ_k

$$(A - \rho_k I)x_k = (A - \rho_k I)^{-1}x_k \mu_k.$$

⁴⁸John William Strutt (Lord Rayleigh) (1842–1919) English mathematician. In his major text *The Theory of Sound* he studied the mechanics of a vibrating string and explained wave propagation. After years of doing research at his estate he took up a position experimental physics at Cambridge 1879–1884. In 1879 he wrote a paper on traveling waves, now known as solitons. He held many official positions including President of the London Mathematical Society. In 1904 he earned the Nobel prize for the discovery of the inert gas argon.

The last equality follows since $A - \rho_k I$ is Hermitian and $\|x_j\|_2 = 1$ for all j . \square

If RQI converges towards an eigenvector corresponding to a *simple* eigenvalue then it can be shown that convergence is at least quadratic. More precisely, it holds that

$$\eta_k \leq c_k \eta_{k-1}^2, \quad \eta_k = \|Aq_k - \rho_k q_k\|_2,$$

where c_k changes only slowly; see Stewart [543, Sec. 7.2].

If the matrix A is real and symmetric (or Hermitian), then the situation is even more satisfactory because of the result in Theorem 10.2.14. This theorem says that if an eigenvector is known to precision ϵ , then the Rayleigh quotient approximates the corresponding eigenvalue to precision ϵ^2 . This leads to local *cubic convergence* for the RQI for real symmetric (or Hermitian) matrices.

Example 10.3.4.

Applying RQI to the symmetric matrix

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 4 \end{pmatrix}.$$

with starting vector $x_0 = (1 \ 1 \ 1)^T / \sqrt{3}$, yields the sequence of approximations

$$\begin{aligned} \rho_0 &= 4.33333333333334, & \rho_1 &= 4.707417179712694, & \rho_2 &= 4.732046272438819, \\ \rho_3 &= 4.732050807568877. \end{aligned}$$

The corresponding normalized eigenvector is

$$x = (0.211324865405187 \ 0.577350269189626 \ 0.788675134594813)^T.$$

The eigenvalue corresponding to the eigenvector closest to x_0 produced by the MATLAB function `eig` is 4.732050807568877. RQI gives full IEEE double precision accuracy in only three iterations.

In the Hermitian case it is no longer necessary to assume that the iteration converges to an eigenvector corresponding to a simple eigenvalue. It can be shown that the iterates x_k will either converge to an eigenvector of A or converge to the bisectors of a pair of eigenvectors of A ; see Parlett [476, Sec. 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* x_0 .

10.3.5 Subspace Iteration

In some cases, for example, in for multiple and clustered eigenvalues it is advisable to compute an invariant subspace rather than a single eigenpair. A natural generalization of the power method, which may improve the convergence in

such situations, is to *simultaneously* iterate with several independent vectors. Let $Z_0 = S = (s_1, \dots, s_p) \in \mathbf{R}^{n \times p}$, be an initial matrix of rank $p > 1$. If we compute a sequence of matrices $\{Z_k\}$, from

$$Z_k = AZ_{k-1}, \quad k = 1, 2, \dots, \quad (10.3.15)$$

then it holds

$$Z_k = A^k S = (A^k s_1, \dots, A^k s_p). \quad (10.3.16)$$

In applications A is often a very large sparse matrix and $p \ll n$.

At first it is not clear that much will be gained by iterating with several vectors. If A has a dominant eigenvalue λ_1 all the columns of Z_k will converge to a scalar multiple of the dominant eigenvector x_1 . Hence, Z_k will be close to a matrix of numerical rank one.

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 this subspace becomes more and more ill-conditioned. To force the vectors to stay independent Bauer [38] suggested the following procedure, which he called **Treppeniteration**. At the k th step the current basis is represented by the unit lower triangular matrix L_k . One then forms AL_k , which is factored into the product $L_{k+1}R_{k+1}$.

Since orthogonal reduction techniques have superior stability it is natural to consider instead to maintain orthogonality between the basis columns. Starting with a matrix Q_0 with orthogonal columns, compute

$$Z_k = AQ_{k-1} = Q_k R_k, \quad k = 1, 2, \dots, \quad (10.3.17)$$

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 (10.3.17) is also called **orthogonal iteration**. Note that R_k plays the role of a normalizing matrix. We have $Q_1 = Z_1 R_1^{-1} = AQ_0 R_1^{-1}$. Similarly, it can be shown by induction that

$$Q_k = A^k Q_0 (R_k \cdots R_1)^{-1}. \quad (10.3.18)$$

It is important to note that if $Z_0 = Q_0$, then both iterations (10.3.15) and (10.3.17) will generate the same sequence of subspaces. $\mathcal{R}(A^k Q_0) = \mathcal{R}(Q_k)$. However, in orthogonal iteration an orthogonal bases for the subspace is calculated at each iteration. (Since the iteration (10.3.15) is less costly it is sometimes preferable to perform the orthogonalization in (10.3.17) 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

$$|\lambda_1| \geq \dots \geq |\lambda_p| > |\lambda_{p+1}| \geq \dots \geq |\lambda_n| \quad (10.3.19)$$

and let

$$\begin{pmatrix} U_1^H \\ U_2^H \end{pmatrix} A(U_1 \ U_2) = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}, \quad (10.3.20)$$

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 almost always the subspaces $\mathcal{R}(Q_k)$ in orthogonal iteration (10.3.17) converge to \mathcal{U}_1 when $k \rightarrow \infty$.

Theorem 10.3.5.

Let $\mathcal{U}_1 = \mathcal{R}(U_1)$ be a dominant invariant subspace of A defined in (10.3.20). Let \mathcal{S} be a p -dimensional subspace of \mathbf{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 (see Definition 8.1.23).

Proof. See Golub and Van Loan [277, pp. 333]. \square

If we perform subspace iteration on p vectors, we are simultaneously performing subspace iteration on a nested sequence of subspaces

$$\text{span}(s_1), \quad \text{span}(s_1, s_2), \dots, \quad \text{span}(s_1, s_2, \dots, s_p).$$

This is also true for orthogonal iteration since this property is not changed by the orthogonalization procedure. Hence, Theorem 10.3.5 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.

We now show that there is a duality between direct and inverse subspace iteration.

Lemma 10.3.6. (Watkins [1982])

Let \mathcal{S} and \mathcal{S}^\perp be orthogonal complementary subspaces of \mathbf{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 $x \perp y \in \mathbf{C}^n$. 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

$$S, AS, A^2 S, \dots, \quad S^\perp, (A^H)^{-1} S^\perp, (A^H)^{-2} S^\perp, \dots$$

are equivalent in that they yield orthogonal complements! This result will be important in Section 10.4.1 for the understanding of the QR algorithm.

Approximations to eigenvalues of A can be obtained from eigenvalues of the sequence of matrices

$$B_k = Q_k^T A Q_k = Q_k^T Z_{k+1} \in \mathbf{R}^{p \times p}. \quad (10.3.21)$$

Note that B_k is a generalized Rayleigh quotient; see Section 10.7.1–10.7.2. Finally, both direct and inverse orthogonal iteration can be performed using a sequence of shifted matrices $A - \mu_k I$, $k = 0, 1, 2, \dots$. $(A - \mu I)^{-1}$

Review Questions

- 3.1** Describe the power method and its convergence properties. Why is it necessary in practice to normalize the sequence of vectors?
- 3.2** Suppose that a good approximation to one of the eigenvalues of A is known. How can inverse iteration be used to compute the corresponding eigenvector?
- 3.3** If the Rayleigh Quotient Iteration converges to a simple eigenvalue of a general matrix A , what is the asymptotic rate of convergence? If A is Hermitian, what can you say then?
- 3.4** Describe how the power method can be generalized to simultaneously iterating with several starting vector.

Problems

- 3.1** Let $A \in \mathbf{R}^{n \times n}$ 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?
- 3.2** The matrix A has one real eigenvalue $\lambda = \lambda_1$ and another $\lambda = -\lambda_1$. All remaining eigenvalues satisfy $|\lambda| < |\lambda_1|$. Generalize the simple power method so that it can be used for this case.
- 3.3** (a) Compute the residual vector corresponding to the last eigenpair obtained in Example 10.3.1, and give the corresponding backward error estimate.
(b) Perform Aitken extrapolation on the Rayleigh quotient approximations in Example 10.3.1 to compute an improved estimate of λ_1 .
- 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 $A - 4I = LDL^T$.

- 3.5** For a symmetric matrix $A \in \mathbf{R}^{n \times n}$ it holds that $\sigma_i = |\lambda_i|$, $i = 1 : n$. Compute with inverse iteration using the starting vector $x = (1, -2, 1)^T$ the smallest

singular value of the matrix

$$A = \begin{pmatrix} 1/5 & 1/6 & 1/7 \\ 1/6 & 1/7 & 1/8 \\ 1/7 & 1/8 & 1/9 \end{pmatrix}$$

with at least two significant digits.

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, and then perform inverse iteration to determine the corresponding eigenvectors. Try as starting vectors both $x = (1, 0)^T$ and $x = (0, 1)^T$.

10.4 The QR Algorithm

10.4.1 The Basic QR Algorithm

Suppose that the matrix $A \in \mathbf{R}^{n \times n}$ has the LU factorization $A = LU$. Then $U = L^{-1}A$, and if the factors are multiplied in reverse order this performs the similarity transformation

$$L^{-1}AL = L^{-1}(LU)L = UL,$$

cf. Lemma 10.1.6. In the LR algorithm⁴⁹ this process is iterated. Setting $A_1 = A$, and compute

$$A_k = L_k U_k, \quad A_{k+1} = U_k L_k, \quad k = 1, 2, \dots \quad (10.4.1)$$

The **LR algorithm** is due to Rutishauser [511, 1958]. It is related to a more general algorithm, the qr algorithm, which can be used to find poles of rational functions or zeros of polynomials; see Volume I, Section 3.5.5.

Repeated application of (10.4.1) gives

$$A_k = L_{k-1}^{-1} \cdots L_2^{-1} L_1^{-1} A_1 L_1 L_2 \cdots L_{k-1}. \quad (10.4.2)$$

or

$$L_1 L_2 \cdots L_{k-1} A_k = A_1 L_1 L_2 \cdots L_{k-1}. \quad (10.4.3)$$

The two matrices defined by

$$T_k = L_1 \cdots L_{k-1} L_k, \quad U_k = U_k U_{k-1} \cdots U_1, \quad (10.4.4)$$

are lower and upper triangular respectively. Forming the product $T_k U_k$ and using (10.4.3) we have

$$\begin{aligned} T_k U_k &= L_1 \cdots L_{k-1} (L_k U_k) U_{k-1} \cdots U_1 \\ &= L_1 \cdots L_{k-1} A_k U_{k-1} \cdots U_1 \\ &= A_1 L_1 \cdots L_{k-1} U_{k-1} \cdots U_1. \end{aligned}$$

⁴⁹In German the LU factorization is called the LR factorization (L and R stands for “links” and “rech”).

Repeating this we obtain the basic relation

$$T_k U_k = A^k. \quad (10.4.5)$$

which shows the close relationship between the LR algorithm and the power method.

Under certain restrictions it can be shown that the matrix A_k converges to an upper triangular matrix U_∞ . The eigenvalues of A then lie on the diagonal of U_∞ . To establish this result several assumptions need to be made. It has to be assumed that the LU factorization exists at every stage. This is not the case for the simple matrix

$$A = \begin{pmatrix} 0 & 1 \\ -3 & 4 \end{pmatrix},$$

with eigenvalues 1 and 3. We could equally well work with the shifted matrix $A + I$, for which the LR algorithm converges. However, there are other problems with the LR algorithm, which make a robust implementation difficult.

To avoid the problems with the LR algorithm it seems natural to devise a similar algorithm using *orthogonal* similarity transformations. This leads to the QR algorithm, published independently by Francis [217, 1961] and Kublanovskaya [387, 1961]. However, Francis paper also contained algorithmic developments needed for the practical implementation. The QR algorithm is still the most important algorithm for computing eigenvalues and eigenvectors of matrices.⁵⁰ For symmetric (Hermitian) matrices alternative algorithms have been developed that can compete with the QR algorithm in terms of speed and accuracy; see Section 10.6.

In the QR algorithm a sequence of matrices $A_{k+1} = Q_k^T A_k Q_k$ similar to $A_1 = A$ are computed by

$$A_k = Q_k R_k, \quad A_{k+1} = R_k Q_k, \quad k = 1, 2, \dots, \quad (10.4.6)$$

where Q_k is orthogonal and R_k is upper triangular. That is, in the k th step the QR decomposition of A_k is computed and then the factors are multiplied in reverse order giving A_{k+1} .

The successive iterates of the QR algorithm satisfy relations similar to those derived for the LR algorithm. If we define

$$P_k = Q_1 Q_2 \cdots Q_k, \quad U_k = R_k \cdots R_2 R_1,$$

where P_k is orthogonal and U_k is upper triangular, then by repeated applications of (10.4.6) it follows that

$$A_{k+1} = P_k^T A P_k. \quad (10.4.7)$$

Further, we have

$$P_k U_k = Q_1 \cdots Q_{k-1} (Q_k R_k) R_{k-1} \cdots R_1 \quad (10.4.8)$$

$$= Q_1 \cdots Q_{k-1} A_k R_{k-1} \cdots R_1 \quad (10.4.9)$$

$$= A_1 Q_1 \cdots Q_{k-1} R_{k-1} \cdots R_1. \quad (10.4.10)$$

⁵⁰The QR algorithm was chosen as one of the 10 algorithms with most influence on scientific computing in the 20th century by the editors of the journal Computing in Science and Engineering.

Repeating this gives

$$P_k U_k = A^k. \quad (10.4.11)$$

We now show that in general the QR iteration is related to orthogonal iteration. Given an orthogonal matrix $\tilde{Q}_0 \in \mathbf{R}^{n \times n}$, orthogonal iteration computes a sequence $\tilde{Q}_1, \tilde{Q}_2, \dots$, where

$$Z_k = A\tilde{Q}_k, \quad Z_k = \tilde{Q}_{k+1}R_k. \quad k = 0, 1, \dots \quad (10.4.12)$$

The matrices $B_k = \tilde{Q}_k^T A \tilde{Q}_k = \tilde{Q}_k^T Z_k$ are similar to A can be computed directly. Using (10.4.12) we have $B_k = (\tilde{Q}_k^T \tilde{Q}_{k+1}) R_k$, which is the QR decomposition of B_k , and

$$B_{k+1} = (\tilde{Q}_{k+1}^T A) \tilde{Q}_{k+1} = (\tilde{Q}_{k+1}^T A \tilde{Q}_k) \tilde{Q}_k^T \tilde{Q}_{k+1} = R_k (\tilde{Q}_k^T \tilde{Q}_{k+1}).$$

Hence, B_{k+1} is obtained by multiplying the QR factors of B_k in reverse order, which is just one step of QR iteration! If, in particular, we take $\tilde{Q}_0 = I$ then $B_0 = A_0$, and it follows that $B_k = A_k$, $k = 0, 1, 2, \dots$, where A_k is generated by the QR iteration (10.4.6). From the definition of B_k and (10.4.6) we have $\tilde{Q}_k = P_{k-1}$, and (compare (10.3.4))

$$A^k = \tilde{Q}_k \tilde{R}_k, \quad \tilde{R}_k = R_k \cdots R_2 R_1. \quad (10.4.13)$$

From this we can conclude that the first p columns of \tilde{Q}_k form an orthogonal basis for the space spanned by the first p columns of A^k , i.e., $A^k(e_1, \dots, e_p)$.

In the QR algorithm subspace iteration takes place on the subspaces spanned by the unit vectors (e_1, \dots, e_p) , $p = 1 : n$. It is important for the understanding of the QR algorithm to recall that therefore, according to Theorem 10.3.5, also inverse iteration by $(A^H)^{-1}$ takes place on the orthogonal complements, i.e., the subspaces spanned by (e_{p+1}, \dots, e_n) , $p = 0 : n - 1$. Note that this means that in the QR algorithm direct iteration is taking place in the top left corner of A , and inverse iteration in the lower right corner. (For the QL algorithm this is reversed, see below.)

Assume that the eigenvalues of A satisfy $|\lambda_p| > |\lambda_{p+1}|$, and let (10.3.20) be a corresponding Schur decomposition. Let $P_k = (P_{k1}, P_{k2})$, $P_{k1} \in \mathbf{R}^{n \times p}$, be defined by (10.4.6). Then by Theorem 10.3.5 with linear rate of convergence equal to $|\lambda_{p+1}/\lambda_p|$

$$\mathcal{R}(P_{k1}) \rightarrow \mathcal{R}(U_1).$$

where U_1 spans the dominant invariant subspace of dimension p of A . It follows that A_k will tend to reducible form

$$A_k = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} + O\left((|\lambda_{p+1}/\lambda_p|)^k\right).$$

This result can be used to show that under rather general conditions A_k will tend to an upper triangular matrix R whose diagonal elements then are the eigenvalues of A .

Theorem 10.4.1.

If the eigenvalues of A satisfy $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$, then the matrices A_k generated by the QR algorithm will tend to upper triangular form. The lower triangular elements $a_{ij}^{(k)}$, $i > j$, converge to zero with linear rate equal to $|\lambda_i/\lambda_j|$.

Proof. A proof can be based on the convergence properties of orthogonal iteration; see Watkins [601]. \square

If the product P_k , $k = 1, 2, \dots$, of the transformations are accumulated the eigenvectors may then be found by calculating the eigenvectors of the final triangular matrix and then transforming them back.

In practice, to speed up convergence, a shifted version of the QR algorithm is used, where

$$A_k - \tau_k I = Q_k R_k, \quad R_k Q_k + \tau_k I = A_{k+1}, \quad k = 0, 1, 2, \dots \quad (10.4.14)$$

and τ_k is a **shift**. It is easily verified that since the shift is restored at the end of the step it holds that $A_{k+1} = Q_k^T A_k Q_k$,

If τ_k approximates a simple eigenvalue λ_j of A , then in general $|\lambda_i - \tau_k| \gg |\lambda_j - \tau_k|$ for $i \neq j$. By the result above the off-diagonal elements in the *last* row of \tilde{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 10.4.2.

Let Q_j and R_j , $j = 0 : k$, be computed by the QR algorithm (10.4.14). Then it holds that

$$(A - \tau_k I) \cdots (A - \tau_1 I)(A - \tau_0 I) = P_k U_k, \quad (10.4.15)$$

where

$$P_k = Q_0 Q_1 \cdots Q_k, \quad U_k = R_k R_{k-1} \cdots R_0. \quad (10.4.16)$$

Proof. For $k = 0$ the relation (10.4.15) is just the definition of Q_0 and R_0 . Assume now that the relation is true for $k - 1$. From $A_{k+1} = Q_k^T A_k Q_k$ and using the orthogonality of P_k

$$A_{k+1} - \tau_k I = P_k^T (A - \tau_k I) P_k. \quad (10.4.17)$$

Hence, $R_k = (A_{k+1} - \tau_k I) Q_k^T = P_k^T (A - \tau_k I) P_k Q_k^T = P_k^T (A - \tau_k I) P_{k-1}$. Post-Multiplying this equation by U_{k-1} we get

$$R_k U_{k-1} = U_k = P_k^T (A - \tau_k I) P_{k-1} U_{k-1},$$

and thus $P_k U_k = (A - \tau_k I) P_{k-1} U_{k-1}$. Using the inductive hypothesis the theorem follows. \square

A variant called the QL algorithm is based on the iteration

$$A_k = Q_k L_k, \quad L_k Q_k = A_{k+1}, \quad k = 0, 1, 2, \dots, \quad (10.4.18)$$

where L_k is *lower* triangular, and is merely a reorganization of the QR algorithm. Let J be a permutation matrix such that JA reverses the rows of A . Then AJ reverses the columns of A and hence JAJ reverses both rows and columns. If R is upper triangular then JRJ is lower triangular. It follows that if $A = QR$ is the QR decomposition then $JAJ = (JQJ)(JRJ)$ is the QL decomposition of JAJ . It follows that the QR algorithm applied to A is the same as the QL algorithm applied to JAJ . Therefore, the convergence theory is the same for both algorithms. However, in the QL algorithm inverse iteration is taking place in the top left corner of A , and direct iteration in the lower right corner.

When combined with a preliminary reduction to Hessenberg or symmetric tridiagonal form the QR algorithm yields a very efficient method for finding all eigenvalues and eigenvectors of small to medium size matrices. Then the necessary modifications to make it into a practical method are described. The general nonsymmetric case is treated in Section 10.4.3 and the real symmetric case in Section 10.5.2.

10.4.2 Reduction to Hessenberg Form

For a real matrix $A \in \mathbf{R}^{n \times n}$ the cost for one QR iteration is $8n^3/3$ flops, which is too much to make it a practical algorithm. An important observation is that if A has the property that $a_{i,j} = 0$, if $i > j + p$, *this form is preserved by the QR algorithm*. The QR factorization of such a matrix is reduced from $O(n^3)$ to $O(pn^2)$. For $p = 1$, this means that the matrix A is an upper Hessenberg matrix.

Let $H \in \mathbf{C}^{n \times n}$ be an upper Hessenberg matrix with positive subdiagonal elements. One step of the QR algorithm, with shift τ , applied to H may be described by the equations

$$H - \tau I = QR, \quad RQ + \tau I = \bar{H}, \quad k = 0, 1, 2, \dots \quad (10.4.19)$$

Here Q is unitary and R right triangular with real positive diagonal elements. To show that the Hessenberg form is preserved we first note that addition or subtraction of τI does not affect the Hessenberg form. Further, if R is nonsingular then $Q = (H - \tau I)R^{-1}$ is a product of an upper Hessenberg matrix and an upper triangular matrix, and therefore a Hessenberg matrix (cf. Problem 7.4.1). By the same arguments RQ and \bar{H} are upper Hessenberg.

Thus, if A is first reduced to Hessenberg form one step of the QR algorithm requires only $O(n^2)$ flops. Since only minor modifications are needed in the complex case we assume that $A \in \mathbf{C}^{n \times n}$. The reduction is achieved by a unitary similarity transformation.

$$Q^H A Q = H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,n-1} & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2,n-1} & h_{2n} \\ & h_{32} & \ddots & \vdots & \vdots \\ & \ddots & \ddots & \ddots & \vdots \\ & & & h_{n,n-1} & h_{nn} \end{pmatrix}. \quad (10.4.20)$$

With $A = A^{(1)}$ the reduction is performed in $(n - 2)$ steps $A^{(2)}, \dots, A^{(n-1)} = H$. We take

$$A^{(k+1)} = (P_k A^{(k)}) P_k, \quad k = 1 : n - 2,$$

where P_k is a complex Householder reflection

$$P_k = I - \frac{2}{\gamma_k} u_k u_k^H, \quad \gamma_k = u_k^H u_k. \quad (10.4.21)$$

The first k elements in the vector in u_k are zero and the remaining component are chosen so that the elements in column k of $P_k A^{(k)}$ below the first subdiagonal are annihilated. The similarity transformation is completed by postmultiplying by P_k . Note that the postmultiplication will not affect the elements in columns $1 : k$.

After the first step the transformed matrix has the form

$$A^{(2)} = P_1 A P_1 = \left(\begin{array}{cc|cccc} h_{11} & h_{12} & \tilde{a}_{13} & \dots & \tilde{a}_{1n} \\ h_{21} & h_{22} & \tilde{a}_{23} & \dots & \tilde{a}_{2n} \\ \hline 0 & \tilde{a}_{32} & \tilde{a}_{33} & \dots & \tilde{a}_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{n2} & \tilde{a}_{n3} & \dots & \tilde{a}_{nn} \end{array} \right),$$

It is possible to make the subdiagonal element real positive by a simple scaling. We assume that $h_{21} = e^{\theta_1} |h_{21}|$, multiply the second row by $e^{-\theta_1}$, and the second column by e^{θ_1} . This is a unitary similarity transformation 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.

An observation, that will turn out to be important, is that each of the Householder matrices P_j , $j = 1 : n$, satisfy $P_j e_1 = e_1$. Therefore, we have

$$Qe_1 = P_1 P_2 \cdots P_{n-2} e_1 = e_1.$$

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, taking $Q = P_0 P_1 \cdots P_{n-2}$ we have $Qe_1 = P_0 e_1 \beta z$.

Note that P_k is completely specified by u_k and γ_k , and that the required products of the form $P_k A$ and AP_k , can be computed by rank one update

$$P_k A = A - u_k (A^H u_k)^H / \gamma_k, \quad AP_k = A - (Au_k) u_k^H / \gamma_k.$$

A simple operation count shows that in the real case these updates require $4(n - k)^2$ flops and hence the reduction

$$4 \sum_{k=1}^n (k^2 + nk) = 10n^3/3 \text{ flops.}$$

Assembling the matrix $Q = Q_0 Q_1 \cdots Q_{n-2}$ adds another $4n^3/3$ flops. As described the reduction to Hessenberg form involves level 2 operations. Dongarra, Hammarling and Sorensen [169] have shown how to speed up the reduction by introduce level 3 operations.

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 transformation of a matrix $A + E$, where

$$\|E\|_F \leq cn^2 u \|A\|_F, \quad (10.4.22)$$

and c is a constant of order unity. Moreover if we use the information stored to generate the product $U = P_1 P_2 \cdots P_{n-2}$ then the computed result is close to the matrix U that reduces $A + E$. This will guarantee that the eigenvalues and transformed eigenvectors of \bar{H} are accurate approximations to those of a matrix close to A .

Remark 10.4.1. It should be noted that *backward stability does not imply that the computed \bar{H} will be close to the matrix H corresponding to the exact reduction of A . Even 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 may be called **irrelevant instability**. It still continues to cause much unnecessary concern. The backward stability of the reduction ensures that each matrix will be similar to A to working precision and will yield approximate eigenvalues to as much absolute accuracy as is warranted.

Definition 10.4.3.

An upper Hessenberg matrix is called **unreduced** if all its subdiagonal elements are nonzero.

If $H \in \mathbf{R}^{n \times n}$ is an unreduced Hessenberg matrix, then $\text{rank}(H) \geq n - 1$, and that therefore if H has a multiple eigenvalue it must be defective. In the following we assume that H is unreduced. This is no restriction because if H has a zero subdiagonal entry, then it can be partitioned into the block-diagonal form

$$H = \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix}.$$

Then the eigenvalue problem for H **decouples** into two eigenproblems for the Hessenberg matrices H_{11} and H_{22} . If these are not unreduced, then the eigenvalue problem can be split into even smaller pieces.

The following important theorem states that for an unreduced Hessenberg matrix the decomposition (10.4.20) is uniquely determined by the first column in Q , i.e., $q_1 = Qe_1$,

Theorem 10.4.4 (Implicit Q Theorem).

Given $A, H, Q \in \mathbf{C}^{n \times n}$, where $Q = (q_1, \dots, q_n)$ is unitary and $H = Q^H A Q$ is upper Hessenberg with positive subdiagonal elements. Then H and Q are uniquely determined by the first column q_1 in Q .

Proof. Assume we have already computed q_1, \dots, q_k and the first $k - 1$ columns in H . (Since q_1 is known this assumption is valid for $k = 1$.) Equating the k th

columns in

$$QH = (q_1, q_2, \dots, q_n)H = A(q_1, q_2, \dots, q_n) = AQ$$

we obtain the equation

$$h_{1,k}q_1 + \cdots + h_{k,k}q_k + h_{k+1,k}q_{k+1} = Aq_k, \quad k = 1 : n-1.$$

Multiplying this by q_i^H and using the orthogonality of Q , we obtain

$$h_{ik} = q_i^H A q_k, \quad i = 1 : k.$$

Since H is unreduced $h_{k+1,k} \neq 0$, and therefore q_{k+1} and $h_{k+1,k}$ are determined by

$$q_{k+1} = h_{k+1,k}^{-1} \left(Aq_k - \sum_{i=1}^k h_{ik}q_i \right),$$

and the conditions that $\|q_{k+1}\|_2 = 1$ and $h_{k+1,k}$ real positive. \square

The proof of the above theorem is constructive and gives an alternative algorithm for generating Q and H known as Arnoldi's process; see Section 11.4.1. This algorithm has the property that only matrix–vector products Aq_k are required, which makes the process attractive when A is large and sparse. The drawback is that roundoff errors will cause a loss of orthogonality in the generated vectors q_1, q_2, q_3, \dots , which has to be taken into account.

10.4.3 The Hessenberg QR Algorithm

We first describe the **explicit-shift** QR algorithm. In this the matrix $H - \tau I$ is first formed and its QR factorization computed. A sequence of (unitary) Givens rotations (see Section 8.3.1) are applied, so that

$$Q^H(H - \tau I) = R, \quad Q^H = G_{n-1,n} \cdots G_{23}G_{12}.$$

Here R is upper triangular with positive diagonal elements. At a typical step ($n = 6$, $j = 3$) the partially reduced matrix has the form

$$\begin{pmatrix} \rho_{11} & \times & \times & \times & \times & \times \\ & \rho_{22} & \times & \times & \times & \times \\ & & \nu_{33} & \times & \times & \times \\ & & & h_{43} & \times & \times \\ & & & & \times & \times \\ & & & & & \times & \times \end{pmatrix}.$$

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 \bar{H} we must now compute

$$RQ + \tau I = RG_{12}^H G_{23}^H \cdots G_{n-1,n}^H + \tau I.$$

The product RG_{12}^H will affect only the first two columns of R , which are replaced by linear combinations of the two columns. This will add a nonzero element in the

(2, 1) position. The rotation G_{23}^T will similarly affect the second and third columns in RG_{12}^T , and adds a nonzero element in the (3, 2) position. The final result is a Hessenberg matrix.

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 the *last* diagonal element r_{nn} 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 $h'_{nn} = \tau$,

The above result shows that if the shift is equal to an eigenvalue τ then the QR algorithm converges in one step to this eigenvalue. This indicates that τ should be chosen as an approximation to an eigenvalue λ . Then $h_{n,n-1}$ will converge to zero at least with linear rate equal to $|\lambda - \tau| / \min_{\lambda' \neq \lambda} |\lambda' - \tau|$. The choice

$$\tau = h_{nn} = e_n^T H e_n$$

is called the **Rayleigh quotient shift**, since it can be shown to produce the same sequence of shifts as the RQI starting with the vector $q_0 = e_n$. With this shift convergence is therefore *asymptotically quadratic*.

If H is real, the QR algorithm can only converge to a complex eigenvalue if complex shifts are used. We could shift by the eigenvalue of

$$C = \begin{pmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{n,n} \end{pmatrix}, \quad (10.4.23)$$

closest to $h_{n,n}$. This has the disadvantage of introducing complex arithmetic even when A is real. A way to avoid complex shifts for real matrices will be described later.

A important question is when to stop the iterations and accept an eigenvalue approximation. If

$$|h_{n,n-1}| \leq \epsilon(|h_{n-1,n-1}| + |h_{n,n}|),$$

where ϵ is a small constant times the unit roundoff, we set $h_{n,n-1} = 0$ and accept h_{nn} as an eigenvalue. This criterion can be justified since it corresponds to a small backward error. In practice, the size of *all* subdiagonal elements should be monitored. Whenever

$$|h_{i,i-1}| \leq \epsilon(|h_{i-1,i-1}| + |h_{i,i}|),$$

for some $i < n$, we set $|h_{i,i-1}|$ and continue to work on smaller subproblems. This is important for the efficiency of the algorithm, since the work is proportional to the square of the dimension of the Hessenberg matrix. An empirical observation is that on the average less than two QR iterations per eigenvalue are required.

When the shift is explicitly subtracted from the diagonal elements this may introduce large relative errors in any eigenvalue of much smaller magnitude than the shift. The **implicit-shift QR-algorithm** avoids this type of error. It uses the fact (see Theorem 10.4.4) that, provided the matrix H_{k+1} in the QR algorithm is unreduced, it is *uniquely defined by the first column in Q_k* . To find the first column $q_1 = Qe_1$ we note that from $H - \tau I = QR$, it follows that

$$h_1 = (H - \tau I)e_1 = Q(Re_1) = r_{11}Qe_1 = r_{11}q_1,$$

i.e., q_1 is proportional to $h_1 = (h_{11} - \tau, h_{21}, 0, \dots, 0)^T$. If the Givens rotation G_{12} is chosen so that

$$G_{12}^T h_1 = \pm \|h_1\|_2 e_1,$$

then $G_{12}e_1$ is proportional to q_1 . In the similarity transformation ($n = 6$)

$$G_{12}^T H = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{pmatrix} \quad G_{12}^T H G_{12} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{pmatrix},$$

the multiplication from the right with 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

$$G_{23}^T G_{12}^T H G_{12} G_{23} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{pmatrix}.$$

The result of this transformation is to push the element outside the Hessenberg form 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. Then we have obtained a Hessenberg matrix $Q^T H Q$, where the first column in Q equals

$$Qe_1 = G_{12}G_{23} \cdots G_{n-1,n}e_1 = G_{12}e_1.$$

By uniqueness it follows that this Hessenberg matrix is the result of a QR step with shift τ . Note that the information about the shift is contained in G_{12} , but it is not explicitly subtracted from the diagonal elements. In the real case the cost of one implicit QR iteration (10.4.19) equals $6n^2$ flops.

The implicit QR algorithm can be generalized to perform several shifts τ_1, \dots, τ_p in one step. These shifts can be chosen to approximate certain eigenvalues of A . The matrix Q is fully determined by its first column, which is proportional to the vector

$$QR e_1 = (H - \tau_p I) \cdots (H - \tau_1 I) e_1 = z_p.$$

The vector $z_1 = (H - \tau_1 I)e_1$ will have all of its entries except the first two equal to zero. The next $z_2 = (H - \tau_2 I)z_1$ is a linear combination of the first two columns of a Hessenberg matrix and therefore has all except its first three elements equal to zero. By induction it follows that z_p will have all but its first $p+1$ elements equal to zero. The QR algorithm then starts with a Householder reflection P_0 such that $P_0 z_p = \beta e_1$. When this is applied from the right $P_0 A P_0$ it will create a “bulge” of $p(p+1)/2p$ elements outside the Hessenberg form. The QR step is completed by chasing this bulge down the diagonal until it disappears.

To avoid complex arithmetic when H is real one can *adopt the implicit-shift QR algorithm to compute the real Schur form* in Theorem 10.1.16, where R is quasi-triangular with 1×1 and 2×2 diagonal blocks. For real matrices this will save a factor of 2–4 over using complex arithmetic. Let τ_1 and τ_2 be the eigenvalues of the matrix C in (10.4.23), and consider a double implicit QR iterations using these shifts. Proceeding as above we compute

$$QR e_1 = (H - \tau_2 I)(H - \tau_1 I)e_1 = (H^2 - (\tau_1 + \tau_2)H + \tau_1 \tau_2 I)e_1.$$

where $(\tau_1 + \tau_2)$ and $\tau_1 \tau_2$ are real. Taking out a factor $h_{21} \neq 0$ this can be written $h_{21}(p, q, r, 0, \dots, 0)^T$, where

$$\begin{aligned} p &= (h_{11}^2 - (\tau_1 + \tau_2)h_{11} + \tau_1 \tau_2)/h_{21} + h_{12}, \\ q &= h_{11} + h_{22} - (\tau_1 + \tau_2), \quad r = h_{32}. \end{aligned} \quad (10.4.24)$$

Note that we do not even have to compute τ_1 and τ_2 , since we have $\tau_1 + \tau_2 = h_{n-1,n-1} + h_{n,n}$, and $\tau_1 \tau_2 = \det(C)$. Substituting this into (10.4.24), and grouping terms to reduce roundoff errors, we get

$$\begin{aligned} p &= [(h_{nn} - h_{11})(h_{n-1,n-1} - h_{11}) - h_{n,n-1}h_{n-1,n}]/h_{21} + h_{12} \\ q &= (h_{22} - h_{11}) - (h_{nn} - h_{11}) - (h_{n-1,n-1} - h_{11}), \quad r = h_{32}. \end{aligned}$$

The double QR step iteration can now be implemented by a chasing algorithm. We first choose rotations G_{23} and G_{12} so that $G_1^T g_1 = G_{12}^T G_{23}^T g_1 = \pm \|g_1\|_2 e_1$, and carry out a similarity transformation

$$G_1^T H = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix}, \quad G_1^T H G_1 = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ + & + & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix}.$$

To preserve the Hessenberg form we then choose the transformation $G_2 = G_{34}G_{23}$ to zero out the two elements $+$ in the first column. Then

$$G_2^T G_1^T H G_1 G_2 = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ + & + & \times & \times & \times & \times \\ & & \times & \times & \times & \times \end{pmatrix}.$$

Note that this step is similar to the first step. The “bulge” of $+$ elements has now shifted one step down along the diagonal, and we continue to chase these elements until they disappear below the last row. We have then completed one double step of the implicit QR algorithm.

If we only want the eigenvalues, then it is not necessary to save the sequence of orthogonal transformations in the QR algorithm. Storing the rotations can be

avoided by alternating pre-multiplications and post-multiplications. For example, once we have formed $G_{23}G_{12}H_k$ the first two columns do not enter in the remaining steps and we can perform the post-multiplication with G_{12}^T . In the next step we compute $(G_{34}((G_{23}G_{12}H_k)G_{12}^T))G_{23}^T$, and so on.

Suppose the QR algorithm has converged to the final upper triangular matrix T . Then we have

$$P^T HP = T,$$

where P is the product of all Givens rotations used in the QR algorithm. The eigenvectors z_i , $i = 1 : n$ of T satisfy $Tz_i = \lambda_i z_i$, $z_1 = e_1$, and z_i is a linear combination of e_1, \dots, e_i . The nonzero components of z_i can then be computed by back-substitution

$$z_{ii} = 1, \quad z_{ji} = -\left(\sum_{k=j+1}^i t_{jk} z_{ki}\right)/(\lambda_j - \lambda_i), \quad j = i-1 : (-1) : 1. \quad (10.4.25)$$

The eigenvectors of H are then given by Pz_i , $i = 1 : n$. Finally $H = Q^T AQ$ has been obtained by reducing a matrix A to Hessenberg form as described in Section 10.6.1, then the eigenvectors of A can be computed from

$$x_i = QPz_i, \quad i = 1 : n. \quad (10.4.26)$$

When only a few selected eigenvectors are wanted, then a more efficient way is to compute these by using inverse iteration. However, if more than a quarter of the eigenvectors are required, it is better to use the procedure outlined above.

It must be remembered that the matrix 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, since with any procedure that involves rounding errors one cannot demonstrate that a matrix is defective. Usually, one therefore should attempt to find a complete set of eigenvectors. If the matrix is nearly defective this will often be evident, in that corresponding computed eigenvectors will be almost parallel.

From the real Schur form $Q^T AQ = T$ computed by the QR algorithm, we get information about some of the invariant subspaces of A . If

$$T = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}, \quad Q = (Q_1 \quad Q_2),$$

and $\lambda(T_{11}) \cap \lambda(T_{22}) = \emptyset$, then Q_1 is an orthogonal basis for the unique invariant subspace associated with $\lambda(T_{11})$. However, this observation is useful only if we want the invariant subspace corresponding to a set of eigenvalues appearing at the top of the diagonal in T . Fortunately, it is easy to modify the real Schur decomposition so that an arbitrary set of eigenvalues are permuted to the top position. Clearly we can achieve this by performing a sequence of transformations, where in each step we interchange two nearby eigenvalues in the Schur form. Thus, we only need to consider the 2×2 case,

$$Q^T AQ = T = \begin{pmatrix} \lambda_1 & h_{12} \\ 0 & \lambda_2 \end{pmatrix}, \quad \lambda_1 \neq \lambda_2.$$

To reverse the order of the eigenvalues we note that $Tx = \lambda_2 x$, where

$$x = \begin{pmatrix} h_{12} \\ \lambda_2 - \lambda_1 \end{pmatrix}.$$

Let G^T be a Givens rotation such that $G^T x = \gamma e_1$. Then $G^T T G(G^T x) = \lambda_2 G^T x$, i.e. $G^T x$ is an eigenvector of $\hat{T} = GTG^T$. It follows that $\hat{T}e_1 = \lambda_2 e_1$ and \hat{T} must have the form

$$\hat{Q}^T A \hat{Q} = \hat{T} = \begin{pmatrix} \lambda_2 & \pm h_{12} \\ 0 & \lambda_1 \end{pmatrix},$$

where $\hat{Q} = QG$.

Example 10.4.1.

A classical example of a Hessenberg matrix whose eigenvalues are ill-conditioned is the **Frank matrix** F_n ([218]), exemplified for $n = 6$ by

$$F_6 = \begin{pmatrix} 6 & 5 & 4 & 3 & 2 & 1 \\ 5 & 5 & 4 & 3 & 2 & 1 \\ & 4 & 4 & 3 & 2 & 1 \\ & & 3 & 3 & 2 & 1 \\ & & & 2 & 2 & 1 \\ & & & & 1 & 1 \end{pmatrix}. \quad (10.4.27)$$

It can be shown that all eigenvalues of F_6 are real, but the smaller ones have huge condition numbers and cannot be computed accurately except using high precision.

An important case where the choice of either the OR or QL algorithm should be preferred is when the matrix A is **graded**; see Section 10.6.2. A matrix is graded if there is a gradual decrease or increase 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. If the large elements instead occur in the lower right corner then the QL algorithm is more stable. (Note that the reduction to Hessenberg form should then be done from bottom up; see also the remark in Section 10.6.2.) Of course, the same effect can be achieved by explicitly reversing the ordering of the rows and columns.

10.4.4 Balancing an Unsymmetric Matrix

By (10.4.22) 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 transformation $\tilde{A} = D^{-1}AD$ which reduces the Frobenius norm. (Note that only the off-diagonal elements are effected by such a transformation.) This can be achieved by **balancing** the matrix A .

Definition 10.4.5.

A matrix $A \in \mathbf{R}^{n \times n}$ is said to be balanced in the norm $\|\cdot\|_p$ 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 .

There are classes of matrices which do not need balancing; for example normal matrices are already balanced for $p = 2$. An iterative algorithm for balancing a matrix has been given by Osborne [457], 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, see Contribution II/11 in [614] and Parlett and Reinsch [481]. More recent work on balancing a matrix has been done by Knight and Ruiz [377].

Example 10.4.2.

As an example consider the matrix

$$A = \begin{pmatrix} 1 & 0 & 10^{-4} \\ 1 & 1 & 10^4 \\ 10^4 & 10^2 & 1 \end{pmatrix}.$$

With $D = \text{diag}(100, 1, 0.01)$ we get

$$B = DAD^{-1} = \begin{pmatrix} 1 & 0 & 1 \\ 10^{-2} & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

The Frobenius norm has been reduced from $\|A\|_F \approx 10^4$ to $\|B\|_F \approx 2.6$.

We describe a slightly simplified balancing algorithm. Let A_0 be the off-diagonal part of A . Note that a diagonal similarity leaves the main diagonal of A 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

$$\alpha_k = \|a_{:,i}\|_p, \quad \beta_k = \|a_{i,:}\|_p.$$

Usually the matrix is balanced in the 1-norm since this requires fewer multiplications than the 2-norm. Assume that $\alpha_k \beta_k \neq 0$ and set

$$\bar{D}_k = I + \gamma_k e_i e_i^T, \quad \gamma_k = \alpha_k / \beta_k,$$

and $D_k = \bar{D}_k D_{k-1}$. Then the matrix

$$A_k = \bar{D}_k A_{k-1} \bar{D}_k^{-1} = D_k A_0 D_k^{-1}$$

will be balanced in its i th row and columns.

The above iterative process will under some conditions converge to a balanced matrix. However, convergence is linear and can be slow.

Note that there are classes of matrices which do not need balancing. For example, normal matrices are already balanced in the 2-norm. Also, there is no need to balance the matrix if an eigenvalue algorithm is to be used which is invariant under scaling as, e.g., some vector iterations.

Review Questions

- 4.1** (a) If the nonzero structure of the matrix A is such that $a_{i,j} = 0$, if $i > j + p$, this form is preserved by the QR algorithm. How much is the cost of a step of the QR algorithm reduced if A has this structure? What is such a matrix called if $p = 1$?
(b) Describe how an arbitrary square matrix can be reduced to Hessenberg form by a sequence of orthogonal similarity transformations.
 - 4.2** What is meant by a graded matrix, and what precautions need to be taken when transforming such a matrix to condensed form?
 - 4.3** If one step of the QR algorithm is performed on A with a shift τ equal to an eigenvalue of A , what can you say about the result? Describe how the shift usually is chosen in the QR algorithm applied to a real symmetric tridiagonal matrix.
 - 4.4** What are the advantages of the implicit shift version of the QR algorithm for a real Hessenberg matrix H ?
 - 4.5** Suppose the eigenvalues to a Hessenberg matrix have been computed using the QR algorithm. How are the eigenvectors best computed (a) if all eigenvectors are needed; (b) if only a few eigenvectors are needed.
 - 4.6** What is meant by balancing a matrix $A \in \mathbf{R}^{n \times n}$? Why can it be advantageous to balance a matrix before computing its eigenvalues?
-

Problems

- 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

$$\hat{L}\hat{U} = UL,$$

where $\hat{L} = (\hat{l}_{ij})$ and $\hat{U} = (\hat{u}_{ij})$ are two new bidiagonal matrices of the same form. Show that both LU and $\hat{U}\hat{L}$ are tridiagonal matrices with all super-diagonal elements equal to one.

- (b) Show that, setting $e_1 = \hat{e}_5 = 0$, the remaining nonzero elements in \hat{L} and \hat{U} are determined by the relations

$$\hat{e}_m + \hat{q}_{m-1} = e_m + q_m, \quad \hat{e}_m \hat{q}_m = e_m q_{m+1}$$

which are the rhombus rules in Rutishauser's qr algorithm.

- 4.2** Let A be the matrix in Example 10.4.2. Apply the balancing procedure described in Section 10.4.4 to A . Use the 1-norm and terminate the iterations when the matrix is balanced to a tolerance equal to 0.01. How much is the Frobenius norm reduced?
- 4.3** The reduction to Hessenberg form can also be achieved by using elementary elimination matrices of the form

$$L_j = I + m_j e_j^T, \quad m_j = (0, \dots, 0, m_{j+1,j}, \dots, m_{n,j})^T.$$

Only the elements *below* the main diagonal in the j th column differ from the unit matrix. If a matrix A is pre-multiplied by L_j we get

$$L_j A = (I + m_j e_j^T) A = A + m_j (e_j^T A) = A + m_j a_j^T,$$

i.e., multiples of the row a_j^T are *added* to the last $n-j$ rows of A . The similarity transformation $L_j A L_j^{-1} = \tilde{A} L_j^{-1}$ is completed by post-multiplying

$$\tilde{A} L_j^{-1} = \tilde{A} (I - m_j e_j^T) = \tilde{A} - (\tilde{A} m_j) e_j^T.$$

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} .

10.5 The Hermitian QR Algorithm

10.5.1 Reduction to Symmetric Tridiagonal Form

Consider a Hermitian band matrix A for which $a_{i,j} = a_{j,i} = 0$, if $i > j+p$. It is easy to show that this band form is preserved by the QR algorithm. In the particular case that $p = 1$, the matrix $A \in \mathbf{R}^{n \times n}$ is Hermitian and tridiagonal. The QR algorithm can be applied very efficiently to such a matrix.

Any orthogonal similarity transformation preserves symmetry, since

$$(Q^T A Q)^T = Q^T A^T Q.$$

It follows that if the reduction to Hessenberg form described before is applied, the result is a *real symmetric tridiagonal matrix*,

$$Q^T A Q = T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}. \quad (10.5.1)$$

As in the unsymmetric case, if T is unreduced, that is, $\beta_k \neq 0$, $k = 2 : n$, then the decomposition is uniquely determined by the first column $q_1 = Qe_1$ of Q . If T has a zero subdiagonal element, then it decomposes into block diagonal form

$$T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix}$$

with symmetric tridiagonal blocks T_1 and T_2 and the eigenproblem splits into smaller pieces.

It is important to take advantage of symmetry to save storage and operations. In the k th step of the orthogonal reduction we compute $A^{(k+1)} = P_k A^{(k)} P_k$, where P_k is again 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 post-multiplication P_k . However, the intermediate matrix $P_k A^{(k)}$ is not symmetric. Therefore, we must compute $P_k A^{(k)} P_k$ directly. Dropping the subscripts k we can write

$$PAP = \left(I - \frac{1}{\gamma} uu^T \right) A \left(I - \frac{1}{\gamma} uu^T \right) \quad (10.5.2)$$

$$= A - up^T - pu^T + u^T p u u^T / \gamma \quad (10.5.3)$$

$$= A - uq^T - qu^T,$$

where

$$p = Au/\gamma, \quad q = p - \beta u, \quad \beta = u^T p / (2\gamma). \quad (10.5.4)$$

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. Since all matrices appearing in the reduction are symmetric, only the lower halves of the matrices need to be stored.

The orthogonal reduction to tridiagonal form has the same stability property as the corresponding algorithm for the unsymmetric case, i.e., the computed tridiagonal matrix is the exact result for a matrix $A+E$, where E satisfies (10.4.22). Hence, the eigenvalues of T will differ from the eigenvalues of A by at most $cn^2 u \|A\|_F$. However, as shown by Wilkinson [611, Sec. 5.28], the computed tridiagonal matrix can differ significantly from the matrix corresponding to exact computation. This has many times alarmed an unaware user. But in spite of this the computed eigenvalues will be accurate. This is an example of a calculation where rounding errors cancel!

There is a class of symmetric matrices for which small eigenvalues are determined with a very small error compared to $\|A\|_F$. This is the class of **scaled diagonally dominant** matrices, see Barlow and Demmel [29, 1990]. A symmetric scaled diagonally dominant (s.d.d) matrix is a matrix of the form DAD , where A is symmetric and diagonally dominant in the usual sense, and D is an arbitrary diagonal matrix. An example of a s.d.d. matrix is the graded matrix

$$A_0 = \begin{pmatrix} 1 & 10^{-4} & \\ 10^{-4} & 10^{-4} & 10^{-8} \\ & 10^{-8} & 10^{-8} \end{pmatrix}$$

whose elements decrease progressively in size as one proceeds diagonally from top to bottom. However, the matrix

$$A_1 = \begin{pmatrix} 10^{-6} & 10^{-2} & \\ 10^{-2} & 1 & 10^{-2} \\ & 10^{-2} & 10^{-6} \end{pmatrix}.$$

is neither diagonally dominant or graded in the usual sense.

The matrix A_0 has an eigenvalue λ of magnitude 10^{-8} , which is quite insensitive to small *relative* perturbations in the elements of the matrix. If the Householder reduction is performed starting from the *top* row of A as described here it is important that the matrix is presented so that the larger elements of A occur in the top left-hand corner. Then the errors in the orthogonal reduction will correspond to small relative errors in the elements of A , and the small eigenvalues of A will not be destroyed.⁵¹

A similar algorithm can be used to transform a Hermitian matrix into a tridiagonal Hermitian matrix using the complex Householder transformation introduced in Section 8.3.1. With $U = P_1 P_2 \cdots P_{n-2}$ we obtain $T = U^H A U$, where T is Hermitian and therefore has positive real diagonal elements. By a diagonal similarity $D T D^{-1}$, $D = \text{diag}(e^{i\phi_1}, e^{i\phi_2}, \dots, e^{i\phi_n})$ it is possible to further transform T so that the off-diagonal elements are real and nonnegative.

If the orthogonal reduction to tridiagonal form is carried out for a symmetric banded matrix A , then the banded structure will be destroyed. By annihilating pairs of elements using Givens rotations in an ingenious order it is possible to perform the reduction *without* increasing the bandwidth. However, it will then take several rotation to eliminate a single element. This algorithm is described in Parlett [478, Sec. 10.5.1]; see also Contribution II/8 in Wilkinson and Reinsch [614]. An operation count shows that the standard reduction is slower if the bandwidth is less than $n/6$. Note that the reduction of storage is often equally important!

10.5.2 The Real Symmetric QR Algorithm

When A is real symmetric and positive definite we can modify the LR algorithm and use the Cholesky factorization $A = LL^T$ instead. The algorithm then takes the form

$$A_k = L_k L_k^T, \quad A_{k+1} = L_k^T L_k, \quad k = 1, 2, \dots \quad (10.5.5)$$

and we have

$$A_{k+1} = L_k^{-1} A_k L_k = L_k^T A_k L_k^{-T}. \quad (10.5.6)$$

Clearly all matrices A_k are symmetric and positive definite and the algorithm is well defined. Repeated application of (10.5.6) gives

$$A_k = T_{k-1}^{-1} A_1 T_{k-1} = T_{k-1}^T A_1 (T_{k-1}^{-1})^T, \quad (10.5.7)$$

⁵¹Note that in the Householder tridiagonalization described in [614], Contribution II/2 the reduction is performed instead from the bottom up.

where $T_k = L_1 L_2 \cdots L_k$. Further, we have

$$A_1^k = (L_1 L_2 \cdots L_k)(L_k^T \cdots L_2^T L_1^T) = T_k T_k^T. \quad (10.5.8)$$

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

$$A_k^T A_k = A_k^2 = R_k^T Q_k^T Q_k R_k = R_k^T R_k, \quad (10.5.9)$$

which shows that R_k^T is the lower triangular Cholesky factor of A_k^2 .

For the Cholesky LR algorithm we have from (10.4.4) and (10.4.5)

$$A_k^2 = L_k L_{k+1} (L_k L_{k+1})^T. \quad (10.5.10)$$

These two Cholesky factorizations (10.5.9) and (10.5.10) of the matrix 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 (10.5.7) 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 using the QR algorithm.

By the methods described in Section 10.5.1 any Hermitian (real symmetric) matrix can by a unitary (orthogonal) similarity transformation be reduced into real, symmetric tridiagonal form. We can also assume without restriction that T is unreduced, since otherwise it can be split up in smaller unreduced tridiagonal matrices.

If T is unreduced and λ an eigenvalue of T , then clearly $\text{rank}(T - \lambda I) = n - 1$ (the submatrix obtained by crossing out the first row and last column of $T - \lambda I$ has nonzero determinant, $\beta_2 \cdots \beta_n \neq 0$). Hence, there is only one eigenvector corresponding to λ and since T is diagonalizable λ must have multiplicity one. *It follows that all eigenvalues of an unreduced symmetric tridiagonal matrix are distinct.*

The QR algorithm also preserves symmetry. Hence, it follows that if T is symmetric tridiagonal, and

$$T - \tau I = QR, \quad T' = RQ + \tau I, \quad (10.5.11)$$

then also $T' = Q^T T Q$ is symmetric tridiagonal.

From Theorem 10.4.4) we have the following result, which can be used to develop an implicit QR algorithm.

Theorem 10.5.1.

Let A be real symmetric, $Q = (q_1, \dots, q_n)$ orthogonal, and $T = Q^T A Q$ an unreduced symmetric tridiagonal matrix. Then Q and T are essentially uniquely determined by the first column q_1 of Q .

Suppose we can find an orthogonal matrix Q with the same first column q_1 as in (10.5.11) such that $Q^T A Q$ is an unreduced tridiagonal matrix. Then, by Theorem 10.5.1, it must be the result of one step of the QR algorithm with shift τ . Equating the first columns in $T - \tau I = QR$ it follows 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

$$G_{12}^T t_1 = \pm \|t_1\|_2 e_1, \quad t_1 = (\alpha_1 - \tau, \beta_2, 0, \dots, 0)^T.$$

We now perform the similarity transformation $G_{12}^T T G_{12}$, which results in fill-in in positions (1,3) and (3,1), pictured below for $n = 5$:

$$G_{12}^T T = \begin{pmatrix} \times & \times & + & & \\ \times & \times & \times & & \end{pmatrix}, \quad G_{12}^T T G_{12} = \begin{pmatrix} \times & \times & + & & \\ \times & \times & \times & & \\ + & \times & \times & \times & \\ & \times & \times & \times & \\ & \times & \times & \times & \\ & \times & \times & \times & \end{pmatrix}.$$

To preserve the tridiagonal form a rotation G_{23} can be used to zero out the fill-in elements.

$$G_{23}^T G_{12}^T T G_{12} G_{23} = \begin{pmatrix} \times & \times & & & \\ \times & \times & \times & + & \\ \times & \times & \times & & \\ + & \times & \times & \times & \\ & \times & \times & \times & \\ & \times & \times & \times & \end{pmatrix}.$$

We continue to “chase the bulge” of $+$ elements down the diagonal, with transformations $G_{34}, \dots, G_{n-1,n}$ after which it disappears. We have then obtained a symmetric tridiagonal matrix $Q^T T Q$, where the first column in Q is

$$G_{12} G_{23} \cdots G_{n-1,n} e_1 = G_{12} e_1.$$

By Theorem 10.4.4 it follows that the result must be the matrix T' in (10.5.11).

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

$$\begin{pmatrix} \alpha_{r-1} & \beta_r \\ \beta_r & \alpha_r \end{pmatrix}, \quad (10.5.12)$$

The Rayleigh quotient shift $\tau = \alpha_r$, gives the same result as Rayleigh Quotient Iteration starting with e_r . This leads to generic cubic convergence, but not guaranteed. In practice, taking the shift to be the eigenvalue of the 2×2 trailing submatrix (10.5.12), closest to α_r , has proved to be more efficient. This is called the **Wilkinson shift**. In case of a tie ($\alpha_{r-1} = \alpha_r$) the smaller $\alpha_r - |\beta_r|$ is chosen. A suitable formula for computing this shift is

$$\tau = \alpha_r - \beta_r^2 / \left(|\delta| + \text{sign}(\delta) \sqrt{\delta^2 + \beta_r^2} \right), \quad \delta = (\alpha_{r-1} - \alpha_r)/2 \quad (10.5.13)$$

(cf. Algorithm (10.6.3)). A great advantage of the Wilkinson shift is that it gives guaranteed *global* convergence.⁵² It can also be shown to give almost always *local*

⁵²For a proof see Wilkinson [612] or Parlett [478, Chapter 8].

cubic convergence, although quadratic convergence might be possible.

Example 10.5.1. 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 $\hat{T} = RQ + \tau I$, has elements

$$\hat{t}_{32} = \hat{t}_{23} = -c_1\epsilon^3/(\epsilon^2 + a^2).$$

This shows that if $\epsilon \ll$ the QR method tends to converge cubically.

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} .

We will not give more details of the algorithm here. 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 the fastest way to get the eigenvalues alone, 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 Section 10.6.2. For a detailed discussion of variants of the symmetric tridiagonal QR algorithm, see Parlett [478].

If T has been obtained by reducing a Hermitian matrix to real symmetric tridiagonal form, $U^H AU = T$, then the eigenvectors are given by

$$x_i = U P e_i, \quad i = 1 : n, \quad (10.5.14)$$

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 orthogonal.

If eigenvectors are to be 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. We then perform the QR algorithm 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 drawback of this approach, however, is the difficulty of getting orthogonal eigenvectors to clustered eigenvalues.

For symmetric tridiagonal matrices one often uses the QL algorithm instead of the QR algorithm. We showed in Section 10.4.3 that the QL algorithm is just the QR algorithm on JAJ , where J is the permutation matrix that reverses the elements in a vector. If A is tridiagonal then JAJ is tridiagonal with the diagonal elements in reverse order.

In the implicit QL algorithm one chooses the shift from the top of A and chases the bulge from bottom to top. The reason for preferring the QL algorithm is simply that in practice it is 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 reversed before the QL algorithm is applied.

10.5.3 The QR–SVD Algorithm

As a preprocessing step for the QR–SVD algorithm the matrix $A \in \mathbf{R}^{m \times n}$ is first reduced to upper triangular form. This is achieved by computing the QR decomposition

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}. \quad (10.5.15)$$

In the following we assume that $m \geq n$. This is no restriction, since otherwise we can consider A^T . Then, if $R = U_R \Sigma V^T$ is the SVD of R , it follows that

$$A = U \Sigma V^T, \quad U = Q \begin{pmatrix} U_R \\ 0 \end{pmatrix}. \quad (10.5.16)$$

Hence, the singular values and the right singular vectors of A and R are the same and the first n left singular vectors of A are easily obtained from those of R .

In the unshifted QR algorithm we set $R_1 = R$ and compute a sequence of upper triangular matrices R_{k+1} , $k = 1, 2, 3, \dots$, as follows. In step k , R_{k+1} is

computed from the QR factorization of the *lower* triangular matrix

$$R_k^T = Q_{k+1} R_{k+1}, k = 1, 2, 3, \dots \quad (10.5.17)$$

Using (10.5.17) we observe that

$$R_k^T R_k = Q_{k+1} (R_{k+1} R_k)$$

is the QR factorization of $R_k^T R_k$. Forming the product in reverse order gives

$$\begin{aligned} (R_{k+1} R_k) Q_{k+1} &= R_{k+1} R_{k+1}^T Q_{k+1}^T Q_{k+1} = R_{k+1} R_{k+1}^T \\ &= R_{k+2}^T Q_{k+2}^T Q_{k+2} R_{k+2} = R_{k+2}^T R_{k+2}. \end{aligned}$$

Hence, two successive iterations of (10.5.17) are equivalent to one iteration of the basic QR algorithm for $R^T R$. Moreover this is achieved without forming $R^T R$, which is essential to avoid loss of accuracy.

Using the orthogonality of Q_{k+1} it follows from (10.5.17) that $R_{k+1} = Q_{k+1}^T R_k^T$, and hence

$$R_{k+1}^T R_{k+1} = R_k (Q_{k+1} Q_{k+1}^T) R_k^T = R_k R_k^T.$$

Further, we have

$$R_{k+2} R_{k+2}^T = R_{k+2} R_{k+1} Q_{k+2} = Q_{k+2}^T (R_k R_k^T) Q_{k+2}. \quad (10.5.18)$$

which shows that we are simultaneously performing an iteration on $R_k R_k^T$, again without explicitly forming this matrix.

One iteration of (10.5.17) is equivalent to one iteration of the Cholesky LR algorithm applied to $B_k = R_k R_k^T$. This follows since B_k has the Cholesky factorization $B_k = R_{k+1}^T R_{k+1}$ and multiplication of these factors in reverse order gives $B_{k+1} = R_{k+1} R_{k+1}^T$. (Recall that for a symmetric, positive definite matrix two steps of the LR algorithm is equivalent to one step of the QR algorithm.)

matrices.

The convergence of this algorithm is enhanced provided the QR factorization of A in the first step is performed using column pivoting. It has been shown that then already the diagonal elements of R_2 are often surprisingly good approximations to the singular values of A .

A further reduction to a more compact bidiagonal form can be achieved using the Golub–Kahan bidiagonalization algorithm presented in given in Section 8.4.5. This uses sequence of Householder transformations alternatingly applied from left and right. Performing this reduction on R we have $Q_B^T R P_B = B$, where

$$B = \begin{pmatrix} q_1 & r_2 & & & & \\ & q_2 & r_3 & & & \\ & & q_3 & \ddots & & \\ & & & \ddots & r_{n-1} & \\ & & & & q_{n-1} & r_n \\ & & & & & q_n \end{pmatrix} \in \mathbf{R}^{n \times n}, \quad (10.5.19)$$

where

$$Q_B = Q_1 \cdots Q_n \in \mathbf{R}^{n \times n}, \quad P_B = P_1 \cdots P_{n-2} \in \mathbf{R}^{n \times n}.$$

This reduction can be carried out in $\frac{4}{3}n^3$ flops. If Q_B and P_B are explicitly required they can be accumulated at a cost of $2(m^2n - mn^2 + \frac{1}{3}n^3)$ and $\frac{2}{3}n^3$ 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 using complex Householder transformations.

We now give some important relationships between the SVD and some symmetric eigenvalue problems. These are useful for proving theoretical results as well as for the development of stable algorithm for computing the SVD.

Theorem 10.5.2.

Let the SVD of $A \in \mathbf{C}^{m \times n}$ be $A = U\Sigma V^H$, where $U \in \mathbf{C}^{m \times m}$ and $V \in \mathbf{C}^{n \times n}$ are unitary. Let $r = \text{rank}(A) \leq \min(m, n)$ and $\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_r) > 0$ and set

$$U = (U_1 \ U_2), \quad U_1 \in \mathbf{C}^{m \times r}, \quad V = (V_1 \ V_2), \quad V_1 \in \mathbf{C}^{n \times r}.$$

Then it holds that

$$C = \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} = Q \begin{pmatrix} \Sigma_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\Sigma_1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} Q^H, \quad (10.5.20)$$

where

$$Q = \frac{1}{\sqrt{2}} \begin{pmatrix} U_1 & \sqrt{2}U_2 & U_1 & 0 \\ V_1 & 0 & -V_1 & \sqrt{2}V_2 \end{pmatrix}, \quad (10.5.21)$$

is a matrix of unitary eigenvectors of C . Hence, 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 (10.5.20) and note that $A = U_1\Sigma_1 V_1^H$ and $A^H = V_1\Sigma_1 U_1^H$. \square

Note that the square of the matrix C in (10.5.20) has block diagonal form

$$\begin{aligned} C^2 &= \begin{pmatrix} AA^H & 0 \\ 0 & A^H A \end{pmatrix} = \begin{pmatrix} U\Sigma\Sigma^H U^H & 0 \\ 0 & V\Sigma^H\Sigma V^H \end{pmatrix} \\ &= \begin{pmatrix} U_1\Sigma_1^2 U_1^H & 0 \\ 0 & V_1\Sigma_1^2 V_1^H \end{pmatrix}. \end{aligned} \quad (10.5.22)$$

Such a matrix C is called **two-cyclic**. This shows that the singular values of the matrices A equal the positive square root of the eigenvalues of the Hermitian matrices $A^H A$ and AA^H . However, these relations do not directly lead to a numerically stable way to compute the SVD since small singular of A will not be determined accurately. values.)

The lack of a stable numerical method for computing the SVD was for a long time an impediment for using the SVD as a computational tool. It seems that

using the relationship to the symmetric eigenvalue problem given above it should be possible to develop a QR-type algorithm for the SVD. However, the explicit computation of $A^T A$ or AA^T must be avoided, since it may lead to a severe loss of accuracy in the smaller singular values and corresponding singular vectors of A . Further, an application of the QR algorithm to the matrix C in (10.5.20) would require a special shift strategy and double the work.

By Theorem 10.5.2 it follows that the eigenvalues of the matrix

$$\begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}.$$

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 this matrix can be brought into the special tridiagonal form

$$T = PAP^T = \begin{pmatrix} 0 & q_1 & & & & \\ q_1 & 0 & r_2 & & & \\ & r_2 & 0 & q_2 & & \\ & & q_2 & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & r_n & 0 & q_n \\ & & & & q_n & 0 & \end{pmatrix}, \quad (10.5.23)$$

with zero diagonal elements. This shows the close connection to a special tridiagonal eigenvalue problem.

We notice that if in (10.5.19) $r_i = 0$, then the matrix 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, $G_{i,i+1}, G_{i,i+2}, \dots, G_{i,n}$ the i th row be zeroed out, and again the matrix breaks up into two parts. Hence, we may without loss of generality assume that none of the elements q_1, q_i, r_i , $i = 2 : n$ are zero. This assumption implies that the matrix $B^T B$ has offdiagonal 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 $\sigma_1 > \dots > \sigma_n > 0$.

To achieve rapid convergence when computing all singular values of a matrix it is essential to introduce shifts. We now describe an algorithm due to Golub and Reinsch [273],⁵³ which uses the bulge chasing implicit shift QR algorithm to BB^T .

⁵³Gene H. Golub (1932–2007) American mathematician and a pioneer in modern matrix computations. He studied at the University of Illinois, where he learnt how to program for the ILLIAC computer by 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 remained for the rest of his life.

Since forming BB^T could lead to a severe loss of accuracy in the smaller singular values and vectors the algorithm works directly with the matrix B . To guarantee global convergence the Wilkinson shift for BB^T is used. This shift is determined from the trailing 2×2 submatrix in BB^T , which is⁵⁴

$$B_2 B_2^T = \begin{pmatrix} q_{n-1}^2 + r_n^2 & q_n r_n \\ q_n r_n & q_n^2 \end{pmatrix}.$$

We note that the sum and product of the eigenvalues are

$$\lambda_1 + \lambda_2 = \text{trace}(B_2^T B_2) = q_{n-1}^2 + q_n^2 + r_n^2, \quad \lambda_1 \lambda_2 = \det(B_2^T B_2) = (q_{n-1} q_n)^2.$$

The eigenvalue closest to $q_n^2 + r_n^2$ is chosen as the shift. Using the formula (10.5.13) for computing the shift we obtain

$$\tau = q_n^2 - \text{sign}(\delta)(q_n r_n)^2 / (|\delta| + \sqrt{\delta^2 + (q_n r_n)^2}), \quad (10.5.24)$$

where

$$\delta = \frac{1}{2}((q_n + q_{n-1})(q_n - q_{n-1}) - r_n^2).$$

These expressions should not be used directly, since they suffer from possible overflow or underflow in the squared subexpressions. A method based on these expressions, which computes the singular values and vectors with high *relative accuracy* is given by Demmel and Kahan [150, 1990].

In the implicit shift QR algorithm for $B^T B$ we first determine a Givens rotation $T_1 = G_{12}$ so that $G_{12}^T t_1 = \pm \|t_1\|_2 e_1$, where

$$t_1 = (BB^T - \tau I)r_1 = \begin{pmatrix} q_1^2 + r_2^2 - \tau \\ q_2 r_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (10.5.25)$$

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

$$T_{n-1}^T \cdots T_2^T T_1^T BB^T T_1 T_2 \cdots T_{n-1}$$

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 $n = 5$),

$$BT_1 \xrightarrow{\quad \rightarrow \quad} \begin{pmatrix} \times & \times & & & \\ + & \times & \times & & \\ & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \end{pmatrix}.$$

⁵⁴Golub and Reinsch use the trailing 2×2 submatrix of $B^T B$, which leads to a slightly different shift.

If we now pre-multiply 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 +:

$$S_1^T BT_1 \rightarrow \begin{pmatrix} \times & \times & + \\ \oplus & \times & \times \\ & \times & \times \\ & & \times & \times \\ & & & \times \end{pmatrix}, \quad S_1^T BT_1 T_2 \rightarrow \begin{pmatrix} \downarrow & \downarrow \\ \times & \times & \oplus \\ \times & \times & \\ + & \times & \times \\ & \times & \times \end{pmatrix}.$$

We can now continue to chase the element + down, with transformations alternately from the right and left until we get a new bidiagonal matrix

$$\hat{B} = (S_{n-1}^T \cdots S_1^T) B (T_1 \cdots T_{n-1}) = U^T B P.$$

But then the matrix

$$\hat{T} = \hat{B}^T \hat{B} = P^T B^T U U^T B P = P^T T P$$

is tridiagonal, where the first column of P equals the first column of T_1 . Hence, if \hat{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 pre-multiplication by a sequence Givens transformations $R_{i,i+1}, \dots, R_{i,n}$, and the matrix then splits as above. In practice, two convergence criteria are used. After each QR step if

$$|r_{i+1}| \leq 0.5u(|q_i| + |q_{i+1}|),$$

where u is the unit roundoff, we set $r_{i+1} = 0$. We then find the smallest p and the largest q such that B splits into quadratic subblocks

$$\begin{pmatrix} B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \end{pmatrix},$$

of dimensions $p, n - p - q$ and, q where B_3 is diagonal and B_2 has a nonzero subdiagonal. Second, if diagonal elements in B_2 satisfy

$$|q_i| \leq 0.5u(|r_i| + |r_{i+1}|),$$

set $q_i = 0$, zero the superdiagonal element in the same row, and repartition 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 $Q_S^T B T_S = \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$. Hence

$$U^T A V = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}, \quad U = Q_B \text{diag}(Q_S, I_{m-n}), \quad V = T_B T_S \quad (10.5.26)$$

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 the left singular vectors U need not be saved or accumulated since they can be applied directly to the right hand side b . Asymptotic flop counts for some different variants are summarized in Table 10.5.3. As usual, only the highest order terms in m and n are shown.

Table 10.5.1. Approximate flop counts for the QR–SVD algorithm.

Option	Golub–Reinsch SVD
Σ, U_1, V	$14mn^2 + \frac{22}{3}n^3$
Σ, U_1	$14mn^2 - 2n^3$
Σ, V	$4n^2(m + 2n)$
Σ	$4n^2(m - n/3)$

The implicit QR–SVD algorithm can be shown to be backward stable. This essentially follows from the fact that we have only applied a sequence of orthogonal transformations to A . Hence, the computed singular values $\bar{\Sigma} = \text{diag}(\bar{\sigma}_k)$ are the exact singular values of a nearby matrix $A + E$, where $\|E\|_2 \leq c(m, n) \cdot u\sigma_1$. Here $c(m, n)$ is a constant depending on m and n and u the unit roundoff. From (8.1.53) it follows that

$$|\bar{\sigma}_k - \sigma_k| \leq c(m, n) \cdot u\sigma_1. \quad (10.5.27)$$

Thus, if A is nearly rank deficient, this will always be revealed by the computed singular values. Note, however, that the smaller singular values may not be computed with high relative accuracy.

The backward error bound (10.5.27) 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 the accuracy can be improved by first sorting the rows after decreasing norm and then performing a QR decomposition of PA using column pivoting.

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. If an initial QR decomposition of A with column pivoting has been done the bidiagonal matrix is usually graded from large at upper left to small at lower right as illustrated below

$$\begin{pmatrix} 1 & 10^{-1} & & \\ & 10^{-2} & 10^{-3} & \\ & & 10^{-4} & 10^{-5} \\ & & & 10^{-6} \end{pmatrix}. \quad (10.5.28)$$

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. However, 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 which are graded in different ways.

The following perturbation result due to Demmel and Kahan [150] shows the remarkable fact that all singular values of a bidiagonal matrix are determined to *full relative precision independent of their magnitudes*.

Theorem 10.5.3.

Let $B \in \mathbf{R}^{n \times n}$ be a bidiagonal matrix with singular values $\sigma_1 \geq \dots \geq \sigma_n$. Let $|\delta B| \leq \omega |B|$, and let $\bar{\sigma}_1 \geq \dots \geq \bar{\sigma}_n$ be the singular values of $\bar{B} = B + \delta B$. Then if $\eta = (2n - 1)\omega < 1$,

$$|\bar{\sigma}_i - \sigma_i| \leq \frac{\eta}{1 - \eta} |\sigma_i|, \quad (10.5.29)$$

$$\max\{\sin \theta(u_i, \tilde{u}_i), \sin \theta(v_i, \tilde{v}_i)\} \leq \frac{\sqrt{2}\eta(1 + \eta)}{\text{relgap}_i - \eta}, \quad (10.5.30)$$

$i = 1 : n$, where the **relative gap** between singular values is

$$\text{relgap}_i = \min_{j \neq i} \frac{|\sigma_i - \sigma_j|}{\sigma_i + \sigma_j}. \quad (10.5.31)$$

From Theorem 10.5.3 it follows that it should be possible to compute all singular values of a bidiagonal matrix to *full relative precision independent of their magnitudes*. For the small singular values this can be achieved by using the *unshifted QR–SVD* algorithm given by (10.5.17). This uses the iteration

$$B_k^T = Q_{k+1} B_{k+1}, \quad k = 0, 1, 2, \dots. \quad (10.5.32)$$

In each step the lower bidiagonal matrix B_k^T is transformed into an upper bidiagonal

matrix B_{k+1} .

$$Q_1^T B \xrightarrow{\quad\quad\quad} \begin{pmatrix} \times & + \\ \otimes & \times \\ \times & \times \\ \times & \times \\ \times & \times \end{pmatrix}, \quad Q_2 Q_1^T B \xrightarrow{\quad\quad\quad} \begin{pmatrix} \times & \times \\ \times & + \\ \otimes & \times \\ \times & \times \\ \times & \times \end{pmatrix},$$

etc. Each iteration in (10.5.32) can be performed with a sequence of $n - 1$ Givens rotations at a cost of only $2n$ multiplications and the generation of $n - 1$ Givens rotations. Two steps of the iteration is 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.)

If two successive steps of the unshifted QR–SVD algorithm are interleaved we get the **zero shift QR algorithm**. The zero shift algorithm is very simple and uses no subtractions, This means that each entry of the transformed matrix is computed to high *relative* accuracy.

Algorithm 10.3. THE ZERO SHIFT QR–SVD ALGORITHM.

The algorithm performs p steps of the zero shift QR algorithm on the bidiagonal matrix B in (10.5.19):

```

for  $k = 1 : 2p$ 
  for  $i = 1 : n - 1$ 
     $[c, s, r] = \text{givrot}(q_i, r_{i+1});$ 
     $q_i = r;$   $q_{i+1} = q_{i+1} * c;$ 
     $r_{i+1} = q_{i+1} * s;$ 
  end
end
```

The implementation of a QR–SVD algorithm that uses the zero shift option for small singular values has been studied in depth by Demmel and Kahan [150]. To give 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.

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 are 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 [587, Chap. 4].

10.5.4 Skew-Symmetric and Unitary Matrices

In a skew-Hermitian matrix $A^H = -A$ the diagonal elements must have zero real part. For example,

$$A = \begin{pmatrix} i & -1+1 \\ 1+i & 2i \end{pmatrix}.$$

is skew-Hermitian. Therefore, a real skew-symmetric matrix $A \in \mathbf{R}^{n \times n}$, $A^T = -A$, has zero diagonal elements. Computing the eigenvalues and eigenvectors of interest, e.g., in nuclear physics.

By an orthogonal similarity transformation A can be transformed to tridiagonal form with real subdiagonal elements. This transformation will preserve the skew-Hermitian (skew-symmetric) form. Hence, the reduced matrix will have the form

$$K = \begin{pmatrix} i\beta_1 & \alpha_1 & & & \\ -\alpha_1 & i\beta_2 & \alpha_2 & & \\ & -\alpha_2 & \ddots & \ddots & \\ & & \ddots & i\beta_{n-1} & \alpha_{n-1} \\ & & & -\alpha_{n-1} & i\beta_n \end{pmatrix} \quad (10.5.33)$$

with α_i and β_i real. It is easily verified that after a diagonal similarity transformation

$$DKD^{-1} = iT \quad D = \text{diag}(1, i, i^2, \dots, (-i)^{n-1}),$$

where the matrix T is a real symmetric tridiagonal matrix with subdiagonal elements equal to α_i , $i = 1 : n - 1$. The eigenvalue problem for T can then be solved by the standard symmetric QR algorithm.

If $A \in \mathbf{R}^{n \times n}$ is a real skew-symmetric matrix then the diagonal elements in T are zero. In this case T is defined by at most $(n - 1)$ nonzero elements and the eigenproblem simplifies. The eigenvalues of K must have zero real part and the nonzero eigenvalues must occur in complex conjugate pairs. It follows that if n is odd, then K has a zero eigenvalue.

When n is odd the zero eigenvalue can be deflated from the matrix using an orthogonal similarity transformation consisting of a product of $(n - 1)/2$ Givens rotations. The following Wilkinson diagram illustrates the case $n = 5$. First rows and columns 3 and 5 are rotated. This introduces two new nonzero elements, which are zeroed out by rotation rows and columns 1 and 5. Note that skew-symmetry is preserved.

$$\rightarrow \begin{pmatrix} 0 & \times & & & \\ \times & 0 & \times & + & \\ & \times & \times & & \\ & & \times & 0 & \oplus \\ & & + & \oplus & 0 \end{pmatrix}, \quad \rightarrow \begin{pmatrix} 0 & \times & & & \\ \times & 0 & \times & & \\ & \times & 0 & \times & \\ & & \times & 0 & \oplus \\ & & \oplus & \oplus & 0 \end{pmatrix}.$$

We now assume that the dimension n is even. Comparing with the tridiagonal matrix (10.5.23) 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

$$B = \begin{pmatrix} \alpha_1 & \alpha_2 & & \\ & \alpha_3 & \ddots & \\ & & \ddots & \alpha_{n-2} \\ & & & \alpha_{n-1} \end{pmatrix}.$$

The algorithm clearly also applies to symmetric tridiagonal matrices with a constant diagonal.

The problem of computing the eigenvalues of a unitary (or real orthogonal) matrix U arises in signal processing. Such a matrix is normal and therefore unitarily diagonalizable. Since $Q^{-1} = Q^H$ its eigenvalues satisfy $\bar{\lambda} = \lambda^{-1}$, or $|\lambda|^2 = 1$. It follows that the eigenvalues lie on the unit circle, and we write

$$\lambda_k = e^{i\theta_k} = \cos \theta_k + i \sin \theta_k, \quad k = 1 : n.$$

A straightforward way to proceed is to note that $U^H = Q\Lambda^H Q^H$ and thus

$$\frac{1}{2}(U + U^H) = \frac{1}{2}Q(\Lambda + \Lambda^H)Q^H.$$

Thus, the Hermitian matrix $\frac{1}{2}(U + U^T)$ has the same eigenvectors and its eigenvalues are $\cos \theta_i$, $i = 1 : n$. A drawback with this approach is that when $|\theta_i|$ is small $\sin \theta_i$ will not be accurately determined. This could be handled by also computing the pure imaginary eigenvalues $i \sin \theta_i$ of the skew-Hermitian matrix $\frac{1}{2}(U - U^T)$. For a real orthogonal matrix this involves solving one real 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 is then to reduce U to Hessenberg form as described in Section 10.4.3. Since this is an orthogonal similarity transformation 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. Therefore, without loss of generality we can assume 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**⁵⁵. Thus H can be written as a product of Givens reflections

$$G_{n-1} \cdots G_2 G_1 H = R = \begin{pmatrix} I_{n-1} & \\ & c_n \end{pmatrix}, \quad |c_n| = 1.$$

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

$$H = H(c_1, c_2, \dots, c_{n-1}) = G_1 G_2 \cdots G_{n-1}.$$

For stability reasons it is essential to retain also the quantities s_1, s_2, \dots, s_{n-1} .

A step in the *shifted* QR algorithm performs the transformation

$$H - \tau I = QR, \quad \bar{H} = RQ + \tau I = Q^T HQ,$$

where τ is usually complex shift. The matrix \bar{H} is unitary Hessenberg matrix that can be shown to have real subdiagonal elements. Hence it has a representation

$$\bar{H} = \bar{H}(\bar{c}_1, \bar{c}_2, \dots, \bar{c}_{n-1}).$$

By taking this structure into account it is possible to perform the QR step in $O(n^2)$ flops instead of $O(n^3)$ flops that is required in the general Hessenberg QR algorithm, see Gragg [280]. The choice of shifts and global convergence is treated in [182].

For a real orthogonal matrix $Q \in \mathbf{R}^{n \times n}$ a more efficient algorithm for computing eigenvalues and eigenvectors can be devised. has been developed by Ammar, Gragg, and Reichel [8]. This algorithm has significant advantages in speed and accuracy. In the first step it is to determine if Q has a real eigenvalue $\lambda = \pm 1$. This can be deflated and the remaining orthogonal matrix must have n even. Next two bidiagonal matrices of dimension roughly $n/2$ are derived, whose singular values are $\cos(\theta_k)$ and $\sin(\theta_k)$, where the eigenvalues of Q are $e^{i\theta_k}$. This algorithm is also described in Golub and Van Loan [277, Sec. 12.6.4].

Review Questions

- 5.1** (a) Show that the symmetry of a matrix is preserved by the QR algorithm. What about normality?
- 5.2** For a certain class of symmetric matrices small eigenvalues are determined with a very small error compared to $\|A\|_F$. Which class?

⁵⁵This name was chosen because of the connection with Schur's work on bounded analytic functions.

- 5.3** What condensed form is usually chosen for the singular value decomposition? What kind of transformations are used for bringing the matrix to condensed form?
 (b) Does the reduction in (a) apply to a complex matrix A ?
-

Problems

- 5.1** Perform a QR step without shift on the matrix

$$A = \begin{pmatrix} \cos \theta & \sin \theta \\ \sin \theta & 0 \end{pmatrix}$$

and show that the offdiagonal elements are reduced to $-\sin^3 \theta$.

- 5.2** Reduce to tridiagonal form, using an exact orthogonal similarity, the real symmetric matrix

$$A = \begin{pmatrix} 1 & \sqrt{2} & \sqrt{2} & \sqrt{2} \\ \sqrt{2} & -\sqrt{2} & -1 & \sqrt{2} \\ \sqrt{2} & -1 & \sqrt{2} & \sqrt{2} \\ 2 & \sqrt{2} & \sqrt{2} & -3 \end{pmatrix}$$

- 5.3** Perform this reduction of the skew-symmetric circulant matrix K with first row equal to

$$(0, 1, 1, 0, -1, -1).$$

- 5.4** (a) Let $Q \in \mathbf{R}^{3 \times 3}$ 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 $+1$, which corresponds to the screw axis of rotation. Show that the two other eigenvalues are of the form $\lambda = e^{\pm i\phi}$.

(b) Let

$$M = \frac{1}{2}(Q^T + Q), \quad K = \frac{1}{2}(Q^T - Q),$$

be the symmetric and skew-symmetric part of Q . Show that M and K have the same eigenvectors as Q . What are their eigenvalues of M and K ?

(c) Show that the eigenvector corresponding to the zero eigenvalue of

$$K = \begin{pmatrix} 0 & k_{12} & k_{13} \\ -k_{12} & 0 & k_{23} \\ -k_{13} & -k_{23} & 0 \end{pmatrix}$$

equals $u_1 = (k_{23}, -k_{13}, k_{12})^T$. Derive the characteristic equation $\det(K - \lambda I) = 0$ and conclude that the two remaining eigenvalues are $\pm i \sin \phi$, where

$$\sin^2 \phi = k_{12}^2 + k_{13}^2 + k_{23}^2.$$

- 5.5** To compute the eigenvalues of the following real symmetric pentadiagonal matrix

$$A = \begin{pmatrix} 4 & 2 & 1 & 0 & 0 & 0 \\ 2 & 4 & 2 & 1 & 0 & 0 \\ 1 & 2 & 4 & 2 & 1 & 0 \\ 0 & 1 & 2 & 4 & 2 & 1 \\ 0 & 0 & 1 & 2 & 4 & 2 \\ 0 & 0 & 0 & 1 & 2 & 4 \end{pmatrix},$$

the matrix is first reduced to A tridiagonal form.

- (a) Determine a Givens rotation G_{23} which zeros the element in position $(3, 1)$. Compute the transformed matrix $A^{(1)} = G_{23}AG_{23}^T$.
- (b) In the matrix $A^{(1)}$ a new nonzero element has been introduced. 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 $A \in \mathbf{R}^{n \times n}$ to symmetric tridiagonal form. How many rotations are needed? How many flops?
- 5.6** Let T be the tridiagonal matrix in (10.5.1), and suppose a QR step using the shift $\tau = \alpha_n$ is carried out,

$$T - \alpha_n I = QR, \quad \tilde{T} = RQ + \alpha_n I.$$

Generalize the result from Problem 10.5.2, and show that if $\gamma = \min_i |\lambda_i(T_{n-1}) - \alpha_n| > 0$, then $|\tilde{\beta}_n| \leq |\beta_n|^3/\gamma^2$.

- 5.7** Let B be the matrix in (10.5.19) 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 the matrix $P^T CP$ becomes a tridiagonal matrix T of the form in (10.5.23).
- 5.8** Modify Algorithm 9.7.1 for the zero shift QR–SVD algorithm so that the two loops are merged into one.
- 5.9** Let $A \in \mathbf{C}^{n \times n}$ be a complex skew-Hermitian matrix (a) Show that the diagonal of A is pure imaginary, but need not be null.
 (b) Show that $\Re(\det(A)) = 0$ if n is odd and $\Im(\det(A)) = 0$ if n is even.
Hint: Show that $\overline{\det(A)} = (-1)^n \det(A)$.
- 5.10** (a) Let σ_i be the singular values of the matrix

$$M = \begin{pmatrix} z_1 & & & \\ z_2 & d_2 & & \\ \vdots & & \ddots & \\ z_n & & & d_n \end{pmatrix} \in \mathbf{R}^{n \times n},$$

where the elements d_i are distinct. Show the interlacing property

$$0 < \sigma_1 < d_2 < \dots < d_n < \sigma_n < d_n + \|z\|_2.$$

(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 10.6.1.

10.6 Some Alternative Algorithms

10.6.1 A Divide and Conquer Algorithm

The QR algorithm is one of the most elegant and efficient algorithms in linear algebra. However, it is basically a sequential algorithm and does not lend itself well to parallelization. For the symmetric tridiagonal eigenproblem and the bidiagonal SVD there are several alternative algorithms, which are faster and in some situations also more accurate.

A divide and conquer algorithm for the symmetric tridiagonal case was first suggested by Cuppen [127] and later modified by Dongarra and Sorensen [170] and Gu and Eisenstat [291].

The basic idea in the divide and conquer algorithm for the symmetric tridiagonal eigenproblem is to divide the tridiagonal matrix $T \in \mathbf{R}^{n \times n}$ into two smaller symmetric tridiagonal matrices T_1 and T_2 as follows:

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \beta_3 & \\ & \ddots & \ddots & \\ & \ddots & \ddots & \\ & & \alpha_{n-1} & \beta_n \\ & & \beta_n & \alpha_n \end{pmatrix} = \begin{pmatrix} T_1 & & 0 \\ \beta_k e_{k-1}^T & \alpha_k & \beta_{k+1} e_1^T \\ 0 & \beta_{k+1} e_1 & T_2 \end{pmatrix} \quad (10.6.1)$$

Here e_j is the j th unit vector of appropriate dimension and T_1 and T_2 are $(k-1) \times (k-1)$ and $(n-k) \times (n-k)$ symmetric tridiagonal principle submatrices of T .

Suppose now that the eigendecompositions of $T_i = Q_i D_i Q_i^T$, $i = 1, 2$ are known. Permuting row and column k to the first position and substituting into (10.6.1) we get

$$T = \begin{pmatrix} \alpha_k & \beta_k e_{k-1}^T & \beta_{k+1} e_1^T \\ \beta_k e_{k-1} & Q_1 D_1 Q_1^T & 0 \\ \beta_{k+1} e_1 & 0 & Q_2 D_2 Q_2^T \end{pmatrix} = Q H Q^T, \quad (10.6.2)$$

where

$$H = \begin{pmatrix} \alpha_k & \beta_k l_1^T & \beta_{k+1} f_2^T \\ \beta_k l_1 & D_1 & 0 \\ \beta_{k+1} f_2 & 0 & D_2 \end{pmatrix}, \quad Q = \begin{pmatrix} 0 & Q_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & Q_2 \end{pmatrix}.$$

and $l_1 = Q_1^T e_{k-1}$ is the last row of Q_1 and $f_2 = Q_2^T e_1$ is the first row of Q_2 . Hence,

the matrix T is reduced to the bordered diagonal form

$$H = \begin{pmatrix} z_1 & z_2 & \cdots & z_n \\ z_2 & \alpha_2 & & \\ \vdots & & \ddots & \\ z_n & & & \alpha_n \end{pmatrix} \quad (10.6.3)$$

by the orthogonal similarity transformation Q . The matrix H is also called a **symmetric arrowhead matrix**. If $H = U\Lambda U^T$ is the spectral decomposition of H , then the spectral decomposition of T equals

$$T = (QU)\Lambda(QU)^T. \quad (10.6.4)$$

To compute the eigensystems of T_1 and T_2 , the splitting in (10.6.1) can be applied recursively until the original tridiagonal matrix T has been reduced to a desired number of small subproblems. Then the above relations may be applied from the bottom up to glue the eigensystems together.

We now give an algorithm for computing the eigenvalues and eigenvectors for the symmetric arrowhead matrix H . This problem is discussed in detail in Wilkinson [611, pp. 95–96]. It is no restriction to assume that $d_2 \leq d_3 \leq \cdots \leq d_n$, since this can be achieved by a symmetric permutation. We make the following observations:

- If $z_{ij} = 0$, then one eigenvalue equals d_j , and the degree of the secular equation is decreased by one.
- If $d_j = d_{j+1}$ for some j , $2 \leq j \leq n-1$, then it can be shown that one eigenvalue of H equals d_i , and again the degree of the secular equation may be reduced 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

$$P_{23} \begin{pmatrix} z_1 & 0 & z_3 \\ 0 & d_2 & 0 \\ z_3 & 0 & d_3 \end{pmatrix} P_{23} = \begin{pmatrix} z_1 & z_3 & 0 \\ z_3 & d_3 & 0 \\ 0 & 0 & d_2 \end{pmatrix} = \begin{pmatrix} H' & 0 \\ 0 & d_2 \end{pmatrix}.$$

Clearly d_2 is an eigenvalue and we can work with the deflated matrix H' .

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 $G_{23}d_2IG_{23}^T = d_2G_{23}G_{23}^T = d_2I$, 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 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 above 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 $\det(H - \lambda I)$ along the first row (see Section 7.1.3) the characteristic polynomial of H is

$$\det(H - \lambda I) = (z_1 - \lambda) \prod_{i=2}^n (d_i - \lambda) - \sum_{j=2}^n z_j^2 \prod_{i \neq j}^n (d_i - \lambda).$$

If λ is an eigenvalue the corresponding eigenvector satisfies the linear system $(H - \lambda_i I)x = 0$. Setting $x_1 = -1$, 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 10.6.1.

The eigenvalues of the arrowhead matrix H satisfy the interlacing property

$$\lambda_1 \leq d_2 \leq \lambda_2 \leq \cdots \leq d_n \leq \lambda_n,$$

and the secular equation

$$\phi(\lambda) = \lambda - z_1 + \sum_{j=2}^n \frac{z_j^2}{d_j - \lambda} = 0. \quad (10.6.5)$$

For each eigenvalue λ_i of H , a corresponding normalized eigenvector is $u_i = \tilde{u}_i / \|\tilde{u}_i\|_2$, where

$$u_i = \left(-1, \frac{z_2}{d_2 - \lambda_i}, \dots, \frac{z_n}{d_n - \lambda_i} \right)^T, \quad \|\tilde{u}_i\|_2^2 = \left(1 + \sum_{j=2}^n \frac{z_j^2}{(d_j - \lambda_i)^2} \right). \quad (10.6.6)$$

The roots of the secular equation are simple and isolated in an interval (d_i, d_{i+1}) where $f(\lambda)$ is monotonic and smooth. Although Newton's method could be used to find the roots it may not be suitable, since the function f has poles at d_2, \dots, d_n . A zero finder based on rational osculatory interpolation with guaranteed quadratic convergence was developed by Bunch et al. [82]. Even this can sometimes require too many iterations and an improved algorithm of Ren-Cang Li [407] is to be preferred.

The main work in the updating is to form the matrix product $X = QU$ in (10.6.4). Since Q is essentially a block 2×2 matrix of order n the work in forming X is approximately n^3 flops. As in recursive Cholesky factorization (see Section 7.3.2) at next lower level we have two subproblems which each takes $1/2^3$ as much work, so the number of flops roughly reduced by a factor of four for each stage. Thus, the total work in the divide and conquer method equals

$$n^3 (1 + 1/4 + 1/4^2 + \cdots) = n^3 / (1 - 1/4) = 4n^3 / 3 \text{ flops.}$$

Also, these flops are spent in matrix multiplication and can use BLAS 3 subroutines. What if only eigenvalues are wanted?

While the eigenvalues are always well conditioned with respect to small perturbations, the eigenvectors can be extremely sensitive in the presence of close eigenvalues. Then computing the eigenvectors using (10.6.6) will not give eigenvectors which are accurately orthogonal unless extended precision is used.

In practice, the formula for the eigenvectors in Lemma 10.6.1 cannot be used directly. The reason for this is that we can only compute an approximation $\hat{\lambda}_i$ to λ_i . Even if $\hat{\lambda}_i$ is very close to λ_i , the approximate ratio $z_j/(d_j - \hat{\lambda}_i)$ can be very different from the corresponding exact ratio. These errors may lead to computed eigenvectors of T which are numerically not orthogonal. An ingenious solution to this problem has been found, which involves modifying the vector z rather than increasing the accuracy of the $\hat{\lambda}_i$; see Gu and Eisenstat [291, 1975]. The resulting algorithm seems to outperform the QR algorithm even on single processor computers.

Because of the overhead in the divide and conquer algorithm it is less efficient than the QR algorithm for matrices of small dimension. A suitable value to switch has been found to be $n = 25$.

A Divide and Conquer Algorithm for the SVD

The computation of the bidiagonal singular value decomposition can also be speeded up by using a divide and conquer algorithm. Such an algorithm was given by Jessup and Sorensen [353] and later improved by Gu and Eisenstat [289]. Given the upper bidiagonal matrix $B \in \mathbf{R}^{n \times n}$ this is recursively divided into subproblems as follows:

$$B = \begin{pmatrix} q_1 & r_1 & & & \\ & q_2 & r_2 & & \\ & & \ddots & \ddots & \\ & & & q_{n-1} & r_{n-1} \\ & & & & q_n \end{pmatrix} = \begin{pmatrix} B_1 & 0 \\ q_k e_k^T & r_k e_1^T \\ 0 & B_2 \end{pmatrix} \quad (10.6.7)$$

where $B_1 \in \mathbf{R}^{n_1 \times (n_1+1)}$ and $B_2 \in \mathbf{R}^{n_2 \times n_2}$, $k = n_1 + 1$, and $n_1 + n_2 = n - 1$. Given the SVDs of B_1 and B_2 ,

$$B_1 = Q_1 (D_1 \ 0) W_1^T, \quad B_2 = Q_2 D_2 W_2^T,$$

and substituting into (10.6.7) we get

$$B = \begin{pmatrix} Q_1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & Q_2 \end{pmatrix} \begin{pmatrix} D_1 & 0 & 0 \\ q_k l_1^T & q_k \lambda_1 & r_k f_2^T \\ 0 & 0 & D_2 \end{pmatrix} \begin{pmatrix} W_1 & 0 \\ 0 & W_2 \end{pmatrix}^T, \quad (10.6.8)$$

where $(l_1^T \ \lambda_1) = e_k^T W_1$ is the last row of W_1 and $f_2^T = e_1^T W_2$ is the first row of W_2 .

After a symmetric permutation of block rows 1 and 2 in the middle matrix it has the form

$$M = \begin{pmatrix} q_k \lambda_1 & q_k l_1^T & r_k f_2^T \\ 0 & D_1 & 0 \\ 0 & 0 & D_2 \end{pmatrix},$$

If the SVD of this matrix is $M = X\Sigma Y^T$, then the SVD of B is

$$B = Q\Sigma W^T, \quad Q = \begin{pmatrix} 0 & Q_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & Q_2 \end{pmatrix} X, \quad W = \begin{pmatrix} W_1 P & 0 \\ 0 & W_2 \end{pmatrix} Y, \quad (10.6.9)$$

where P is the permutation matrix that permutes the last column into first position.

The middle matrix in (10.6.8) has the form

$$M = \begin{pmatrix} z_1 & z_2 & \cdots & z_n \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix} = D + e_1 z^T, \quad (10.6.10)$$

where $D = \text{diag}(d_1, d_2, \dots, d_n)$, $d_1 \equiv 0$, contains the elements in D_1 and D_2 . Here d_1 is introduced to simplify the presentation. We further assume that

$$0 = d_1 \leq d_2 \leq d_3 \leq \dots \leq d_n,$$

which can be achieved by a row and column permutation.

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 the matrix

$$M^T M = X\Sigma^2 X^T = D^2 + z e_1^T e_1 z^T = D^2 + z z^T.$$

This matrix has the same form as in Theorem 10.2.12 with $\mu = 1$. Further, $My_i = \sigma_i x_i$, which shows that so if y_i is a right singular vector then My_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 .

Lemma 10.6.2.

Let the SVD of the matrix in (10.6.10) 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 value satisfy the interlacing property

$$0 = d_1 < \sigma_1 < d_2 < \sigma_2 < \cdots < d_n < \sigma_n < d_n + \|z\|_2$$

where $z = (z_1, \dots, z_n)^T$. The secular equation is

$$f(\sigma) = 1 + \sum_{k=1}^n \frac{z_k^2}{d_k^2 - \sigma^2} = 0.$$

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

$$\tilde{y}_i = \left(\frac{z_1}{d_1^2 - \sigma_i^2}, \dots, \frac{z_n}{d_n^2 - \sigma_i^2} \right), \quad \tilde{x}_i = \left(-1, \frac{d_2 z_2}{d_2^2 - \sigma_i^2}, \dots, \frac{d_n z_n}{d_n^2 - \sigma_i^2} \right). \quad (10.6.11)$$

and

$$\|\tilde{y}_i\|_2^2 = \left(\sum_{j=1}^n \frac{z_j^2}{(d_j^2 - \sigma_i^2)^2} \right), \quad \|\tilde{x}_i\|_2^2 = \left(1 + \sum_{j=2}^n \frac{(d_j z_j)^2}{(d_j^2 - \sigma_i^2)^2} \right). \quad (10.6.12)$$

In the divide and conquer algorithm for computing the SVD of B this process is recursively applied to B_2 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 since, unlike B , B_1 is not a square matrix.

The secular equation can be solved efficiently and accurately by the algorithm of Ren-Cang Li. 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 singular vectors which are accurately orthogonal without using extended precision a similar approach as used for obtaining orthogonal eigenvectors can be used; see Gu and Eisenstat [290].

10.6.2 Spectrum Slicing

Using Sylvester's law of inertia (see Theorem 7.3.9) a simple and important method called **spectrum slicing** can be developed for counting the eigenvalues greater than a given real number τ of a Hermitian matrix A . In the following, we treat the real symmetric case, but the method is easily generalized to the Hermitian case. The following theorem is a direct consequence of Sylvester's law of inertia.

Theorem 10.6.3.

Assume that symmetric Gaussian elimination can be carried through for $A - \tau I$ yielding the factorization (cf. (7.3.2))

$$A - \tau I = LDL^T, \quad D = \text{diag}(d_1, \dots, d_n), \quad (10.6.13)$$

where L is a unit lower triangular matrix. Then $A - \tau I$ is congruent to D , and hence the number of eigenvalues of A greater than τ equals the number of positive elements $\pi(D)$ in the sequence d_1, \dots, d_n .

Example 10.6.1.

The LDL^T factorization

$$A - 1 \cdot I = \begin{pmatrix} 1 & 2 & \\ 2 & 2 & -4 \\ & -4 & -6 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -2 & \\ & & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 & \\ 1 & 1 & 2 \\ & 1 & \end{pmatrix}.$$

shows that the matrix A has two eigenvalues greater than 1.

The LDL^T factorization may fail to exist if $A - \tau I$ is not positive definite. This will happen for example if we choose the shift $\tau = 2$ for the matrix in Example 10.6.1. Then $a_{11} - \tau = 0$, and the first step in the factorization cannot be carried out. A closer analysis shows that the factorization will fail, if and only if τ equals an eigenvalue to one or more of the $n - 1$ leading principal submatrices of A . If τ is chosen in a small interval around each of these values, big growth of elements occurs and the factorization may give the wrong count. In such cases one should perturb τ by a small amount (**how small?**) and restart the factorization from the beginning.

For the special case when A is a symmetric tridiagonal matrix the procedure outlined above becomes particularly efficient and reliable. Here the factorization is $T - \tau I = LDL^T$, where L is unit lower bidiagonal and $D = \text{diag}(d_1, \dots, d_n)$. The remarkable fact is that if we only take care to avoid over/underflow then *element growth will not affect the accuracy of the slice*.

Algorithm 10.4. Tridiagonal Spectrum Slicing.

Let T be the tridiagonal matrix (10.5.1). Then the number π of eigenvalues greater than a given number τ is generated by the following algorithm:

```

 $d_1 := \alpha_1 - \tau;$ 
 $\pi := \text{if } d_1 > 0 \text{ then } 1 \text{ else } 0;$ 
 $\text{for } k = 2 : n$ 
     $d_k := (\alpha_k - \beta_k(\beta_k/d_{k-1})) - \tau;$ 
     $\text{if } |d_k| < \sqrt{\omega} \text{ then } d_k := \sqrt{\omega};$ 
     $\text{if } d_k > 0 \text{ then } \pi := \pi + 1;$ 
 $\text{end}$ 

```

Here, to prevent breakdown of the recursion, a small $|d_k|$ is replaced by $\sqrt{\omega}$ where ω is 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 computing initially β_k^2 , which however may cause unnecessary over/underflow. Assuming that no over/underflow occurs Algorithm 10.6.2 is backward stable. A round-off error analysis shows that the computed values \bar{d}_k satisfy exactly

$$\bar{d}_k = fl((\alpha_k - \beta_k(\beta_k/\bar{d}_{k-1})) - \tau)$$

$$\begin{aligned}
&= \left(\left(\alpha_k - \frac{\beta_k^2}{\bar{d}_{k-1}} (1 + \epsilon_{1k})(1 + \epsilon_{2k}) \right) (1 + \epsilon_{3k}) - \tau \right) (1 + \epsilon_{4k}) \quad (10.6.14) \\
&\equiv \alpha'_k - \tau - (\beta'_k)^2 / \bar{d}_{k-1}, \quad k = 1 : n,
\end{aligned}$$

where $\beta_1 = 0$ and $|\epsilon_{ik}| \leq u$. Hence, the computed number $\bar{\pi}$ is the exact number of eigenvalues greater than τ of a matrix A' , where A' has elements satisfying

$$|\alpha'_k - \alpha_k| \leq u(2|\alpha_k| + |\tau|), \quad |\beta'_k - \beta_k| \leq 2u|\beta_k|. \quad (10.6.15)$$

This is a very satisfactory backward error bound. It has been improved even further by Kahan [362, 1966], who shows that the term $2u|\alpha_k|$ in the bound can be dropped, see also Problem 10.6.1. Hence, the eigenvalues found by bisection differ by a factor 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 $2u$. This is obviously a very satisfactory result.

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

$$\pi(D_l) \geq k, \quad \pi(D_u) < k$$

so that λ_k lies in the interval $[\tau_l, \tau_u]$. We can then use p steps of the bisection (or multisection) method (see Volume I, Section 6.1.1) to locate λ_k in an interval of length $(\tau_u - \tau_l)/2^p$. From Geršgorin's theorem it follows that all the eigenvalues of a tridiagonal matrix are contained in the union of the intervals

$$\alpha_i \pm (|\beta_i| + |\beta_{i+1}|), \quad i = 1 : n$$

where $\beta_1 = \beta_{n+1} = 0$.

Using the bound in Theorem 10.2.13 it follows that the bisection error in each computed eigenvalue is bounded by $|\bar{\lambda}_j - \lambda_j| \leq \|A' - A\|_2$, where from (10.3.15), using the improved bound by Kahan, and the inequalities $|\tau| \leq \|A\|_2$, $|\alpha_k| \leq \|A\|_2$ it follows that

$$|\bar{\lambda}_j - \lambda_j| \leq 5u\|A\|_2. \quad (10.6.16)$$

This shows that the absolute error in the computed eigenvalues is always small. If some $|\lambda_k|$ is small it may be computed with poor *relative* precision. In some special cases (for example, tridiagonal, graded matrices) even very small eigenvalues are determined to high relative precision by the elements in the matrix.

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. However, if A is a banded matrix and only few eigenvalues are to be determined then the Band Cholesky Algorithm 7.4.3 can be used to slice the spectrum. It is then necessary to monitor the element growth in the factorization. We finally mention that the technique of spectrum slicing is also applicable to the computation of selected singular values of a matrix and to the generalized eigenvalue problem

$$Ax = \lambda Bx,$$

where A and B are symmetric and B or A positive definite, see Section 10.7.

The method of spectrum slicing can also be used for computing selected singular values of a bidiagonal matrix B . We proceed by applying Algorithm 10.6.2 for spectrum slicing to the special tridiagonal matrix T (10.5.23) with zeros on the main diagonal. We recall that after a symmetric permutation of rows and columns this matrix equals

$$C = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}, \quad B = \begin{pmatrix} q_1 & r_2 & & & \\ & q_2 & r_3 & & \\ & & \ddots & \ddots & \\ & & & q_{n-1} & r_n \\ & & & & q_n \end{pmatrix} \in \mathbf{R}^{n \times n}. \quad (10.6.17)$$

Therefore the eigenvalues of T are $\pm\sigma_i$, $i = 1 : n$. Taking advantage of the zero diagonal this algorithm can be simplified so that one slice requires only of the order $2n$ flops. Given the elements q_1, \dots, q_n and r_2, \dots, r_n of T in (10.5.23), the following algorithm generates the number π of singular values of T greater than a given value $\sigma > 0$.

Algorithm 10.5. Singular Values by Spectrum Slicing.

Let T be the tridiagonal matrix (10.5.1). Then the number π of eigenvalues greater than a given number σ is generated by the following algorithm:

```

 $d_1 := -\sigma;$ 
 $flip := -1;$ 
 $\pi := \text{if } d_1 > 0 \text{ then } 1 \text{ else } 0;$ 
 $\text{for } k = 2 : 2n$ 
 $flip := -flip;$ 
 $\text{if } flip = 1 \text{ then } \beta = q_{k/2}$ 
 $\text{else } \beta = r_{(k+1)/2};$ 
 $\text{end}$ 
 $d_k := -\beta(\beta/d_{k-1}) - \tau;$ 
 $\text{if } |d_k| < \sqrt{\omega} \text{ then } d_k := \sqrt{\omega};$ 
 $\text{if } d_k > 0 \text{ then } \pi := \pi + 1;$ 
 $\text{end}$ 

```

Spectrum slicing algorithm for computing singular values has been analyzed by Fernando [199]. and shown to provide high relative accuracy also for tiny singular values.

10.6.3 Jacobi Methods

One of the oldest methods for solving the eigenvalue problem for real symmetric (or Hermitian) matrices is **Jacobi's⁵⁶ method**. Because it is at least three times slower than the QR algorithm it fell out of favor for a period. However, Jacobi's method is easily parallelized and sometimes more accurate; see [152].

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 *all eigenvalues of symmetric positive definite matrices with uniformly better relative accuracy, first reduces the matrix to tridiagonal form*. Note that, although the QR algorithm is backward stable (see Section 10.5), high relative accuracy can only be guaranteed for the larger eigenvalues (those near $\|A\|$ in magnitude).

The Jacobi method solves the eigenvalue problem for $A \in \mathbf{R}^{n \times n}$ by employing a sequence of similarity transformations

$$A_0 = A, \quad A_{k+1} = J_k^T A_k J_k \quad (10.6.18)$$

such that the sequence of matrices A_k , $k = 1, 2, \dots$ tends to a diagonal form. For each k , J_k is chosen as a plane rotations $J_k = G_{pq}(\theta)$, defined by a pair of indices (p, q) , $p < q$, called the pivot pair. The angle θ is chosen so that the off-diagonal elements $a_{pq} = a_{qp}$ are reduced to zero, i.e., by solving a 2×2 subproblems. We note that only the entries in rows and columns p and q of A will change, and since symmetry is preserved only the upper triangular part of each A needs to be computed.

To construct the Jacobi transformation J_k we consider the symmetric 2×2 eigenvalue problem for the principal submatrix A_{pq} formed by rows and columns p and q . For simplicity of notation we rename $A_{k+1} = A'$ and $A_k = A$. Hence, we want to determine $c = \cos \theta$, $s = \sin \theta$ so that

$$\begin{pmatrix} l_p & 0 \\ 0 & l_q \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}. \quad (10.6.19)$$

Equating the off-diagonal elements we obtain (as $a_{pq} = a_{qp}$)

$$0 = (a_{pp} - a_{qq})cs + a_{pq}(c^2 - s^2), \quad (10.6.20)$$

which shows that the angle θ satisfies

$$\tau \equiv \cot 2\theta = (a_{qq} - a_{pp})/(2a_{pq}), \quad a_{pq} \neq 0. \quad (10.6.21)$$

The two diagonal elements a_{pp} and a_{qq} are transformed as follows,

$$\begin{aligned} a'_{pp} &= c^2 a_{pp} - 2csa_{pq} + s^2 a_{qq} = a_{pp} - ta_{pq}, \\ a'_{qq} &= s^2 a_{pp} + 2csa_{pq} + c^2 a_{qq} = a_{qq} + ta_{pq}. \end{aligned}$$

⁵⁶Carl Gustaf Jacob Jacobi (1805–1851), German mathematician. Jacobi joined the faculty of Berlin university in 1825. Like Euler, he was a prolific calculator, who drew a great deal of insight from immense algorithmical work. His method for computing eigenvalues was published in 1846; see [351].

where $t = \tan \theta$. We call this a **Jacobi transformation**. The following stopping criterion should be used:

$$\text{if } |a_{ij}| \leq \text{tol} (a_{ii}a_{jj})^{1/2}, \text{ set } a_{ij} = 0, \quad (10.6.22)$$

where tol is the relative accuracy desired.

A stable way to perform a Jacobi transformation is to first compute $t = \tan \theta$ as the root of smallest modulus to the quadratic equation $t^2 + 2\tau t - 1 = 0$. This choice ensures that $|\theta| < \pi/4$, and can be shown to minimize the difference $\|A' - A\|_F$. In particular, this will prevent the exchange of the two diagonal elements a_{pp} and a_{qq} , when a_{pq} is small, which is critical for the convergence of the Jacobi method. The transformation (10.6.19) is best computed by the following algorithm.

Algorithm 10.6.

Jacobi transformation matrix ($a_{pq} \neq 0$):

```
[c, s, lp, lq] = jacobi(app, apq, aqq)
τ = (aqq - app)/(2apq);
t = sign(τ)/(|τ| + √(1 + τ2));
c = 1/√(1 + t2); s = t · c;
lp = app - tapq;
lq = aqq + tapq;
end
```

The computed transformation is applied also to the remaining elements in rows and columns p and q of the full matrix A . These are transformed for $j \neq p, q$ according to

$$\begin{aligned} a'_{jp} &= a'_{pj} = ca_{pj} - sa_{qj} = a_{pj} - s(a_{qj} + ra_{pj}), \\ a'_{jq} &= a'_{qj} = sa_{pj} + ca_{qj} = a_{qj} + s(a_{pj} - ra_{qj}). \end{aligned}$$

where $r = s/(1 + c) = \tan(\theta/2)$. (The formulas are written in a form, due to Rutishauser [512, 1971], which reduces roundoff errors.)

If symmetry is exploited, then one Jacobi transformation takes about $4n$ flops. Note that an off-diagonal element made zero at one step will in general become nonzero at some later stage. The Jacobi method will also destroy the band structure if A is a banded matrix.

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, we note that the Frobenius norm of a matrix is invariant under multiplication from left

or right with an orthogonal matrix. Therefore, since $a'_{pq} = 0$ we have

$$(a'_{pp})^2 + (a'_{qq})^2 = a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2.$$

We also have that $\|A'\|_F^2 = \|A\|_F^2$, and it follows that

$$S(A') = \|A'\|_F^2 - \sum_{i=1}^n (a'_{ii})^2 = S(A) - 2a_{pq}^2.$$

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 since

$$2a_{pq}^2 \geq S(A_k)/N, \quad N = n(n-1)/2,$$

we have $S(A_{k+1}) \leq (1-1/N)S(A_k)$. 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 has been shown that ultimately the rate of convergence is quadratic, so that for k large enough, we have $S(A_{k+1}) < cS(A_k)^2$ for some constant c . The iterations are repeated until $S(A_k) < \delta\|A\|_F$, where δ is a tolerance, which can be chosen equal to the unit roundoff u . From the Bauer–Fike Theorem 10.2.4 it then follows that the diagonal elements of A_k then 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. In a **cyclic Jacobi method**, the $N = \frac{1}{2}n(n-1)$ off-diagonal elements are instead annihilated in some predetermined order, each element being 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 $|a_{pq}|$ is smaller than some threshold; see Forsythe and Henrici [214, 1960]. To ensure a good rate of convergence this 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

$$\begin{array}{cccccc} (1, 2), & (1, 3), & \dots & (1, n) \\ & (2, 3), & \dots & (2, n) \\ & \dots & \dots & \dots \\ & & & (n-1, n) \end{array} \tag{10.6.23}$$

which is cyclically repeated. About $2n^3$ flops per sweep is required. In practice, with the cyclic Jacobi method not more than about 5 sweeps are needed to obtain eigenvalues of more than single precision accuracy even when n is large. The number of sweeps grows approximately as $O(\log n)$, and 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 can easily be obtained in the Jacobi method by computing the product of all the 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

$$X_k = X_{k-1} J_k, \quad k = 1, 2, \dots \quad (10.6.24)$$

In each transformation the two columns (p, q) of X_{k-1} is rotated, which requires $4n$ flop. Hence, in each sweep an additional $2n$ flops is needed, which doubles the operation count for the method.

The Jacobi method is very suitable for parallel computation since several noninteracting 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 the $n/2$ Jacobi transformations can be performed simultaneously. A sweep needs at least $n - 1$ such parallel steps. Several parallel schemes which uses this minimum number of steps have been constructed. These can be illustrated in the $n = 8$ case by

$$(p, q) = \begin{aligned} & (1, 2), (3, 4), (5, 6), (7, 8) \\ & (1, 4), (2, 6), (3, 8), (5, 7) \\ & (1, 6), (4, 8), (2, 7), (3, 5) \\ & (1, 8), (6, 7), (4, 5), (2, 3) \\ & (1, 7), (8, 5), (6, 3), (4, 2) \\ & (1, 5), (7, 3), (8, 2), (6, 4) \\ & (1, 3), (5, 2), (7, 4), (8, 6) \end{aligned}$$

The rotations associated with *each row* of the above 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.

10.6.4 Jacobi Methods for Computing the SVD

Jacobi-type methods for computing the SVD $A = U\Sigma V^T$ of a matrix were developed in the 1950's. There are cases for which these algorithm compute the smaller singular values more accurately than any algorithm based on a preliminary bidiagonal reduction.

We first outline an algorithm for computing the SVD of an 2×2 upper triangular matrix. Even though this seems to be an easy task, care must be taken to avoid underflow and overflow.

Lemma 10.6.4.

The singular values of the upper triangular 2×2 matrix

$$R = \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}$$

are given by

$$\sigma_{1,2} = \frac{1}{2} \left| \sqrt{(r_{11} + r_{22})^2 + r_{12}^2} \pm \sqrt{(r_{11} - r_{22})^2 + r_{12}^2} \right|, \quad (10.6.25)$$

The largest singular value is computed using the plus sign, and the smaller is then obtained from

$$\sigma_2 = |r_{11}r_{22}|/\sigma_1. \quad (10.6.26)$$

Proof. The eigenvalues of the matrix

$$S = R^T R = \begin{pmatrix} r_{11}^2 & r_{11}r_{12} \\ r_{11}r_{12} & r_{12}^2 + r_{22}^2 \end{pmatrix}.$$

are $\lambda_1 = \sigma_1^2$ and $\lambda_2 = \sigma_2^2$. 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 (10.6.25). \square

The formulas above should not be used precisely as written, since it is necessary to guard against overflow and underflow. The following program computes singular values and orthogonal left and right singular vectors

$$\begin{pmatrix} c_u & s_u \\ -s_u & c_u \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix} \begin{pmatrix} c_v & -s_v \\ s_v & c_v \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}. \quad (10.6.27)$$

of a 2×2 upper triangular matrix with $|r_{11}| \geq |r_{22}|$:

```
[c_u, s_u, c_v, s_v, sigma1, sigma2] = svd(r11, r12, r22)
l = (|r11| - |r22|)/|r11|;
m = r12/r11; t = 2 - l;
s = sqrt(t^2 + m^2); r = sqrt(l^2 + m^2);
a = (s + r)/2;
sigma1 = |r11|a; sigma2 = |r22|/a;
t = (1 + a)(m/(s + t) + m/(r + l));
l = sqrt(t^2 + 4);
c_v = 2/l; s_v = -t/l;
c_u = (c_v - s_v*m)/a; s_u = s_v*(r22/r11)/a;
end
```

Even this program can be made to fail. A Fortran program that always gives high *relative accuracy* in the singular values and vectors has been developed by

Demmel and Kahan. It was briefly mentioned in [150] but not listed there. A listing of the Fortran program and a sketch of its error analysis is found in an appendix of [22].

There are two different ways to generalize the Jacobi method for the SVD problem. We assume that $A \in \mathbf{R}^{n \times n}$ is a square nonsymmetric matrix. This is no restriction, since we can first compute the QR factorization

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

and then apply the Jacobi SVD method to R . In the **two-sided Jacobi SVD** algorithm for the SVD of A (Kogbetliantz [379]) the elementary step consists of two-sided Givens transformations

$$A' = J_{pq}(\phi) A J_{pq}^T(\psi), \quad (10.6.28)$$

where $J_{pq}(\phi)$ and $J_{pq}(\psi)$ 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 $J_{pq}(\phi)$ and $J_{pq}(\psi)$ 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 of nonnegative diagonal elements is no restriction, since we can change the sign of these by pre-multiplication with an orthogonal matrix $\text{diag}(\pm 1, \pm 1)$.

Since the Frobenius norm is invariant under orthogonal transformations 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.

At first a drawback of the above algorithm seems to be that it works all the time on a full $m \times n$ unsymmetric matrix. However, if a proper cyclic rotation strategy is used, then at each step the matrix will be essentially triangular. 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 twosided Jacobi SVD method.

In the **one-sided Jacobi SVD** algorithm Givens transformations are used to find an orthogonal matrix V such that the matrix AV has orthogonal columns. Then

$AV = U\Sigma$ and the SVD of A 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 columns:

$$(\hat{a}_p, \hat{a}_q) = (a_p, a_q) \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad p < q. \quad (10.6.29)$$

The parameters c, s are determined so that the rotated columns are orthogonal, or equivalently so that

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} \|a_p\|_2^2 & a_p^T a_q \\ a_q^T a_p & \|a_q\|_2^2 \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}^T$$

is diagonal. This 2×2 symmetric eigenproblem can be solved by a Jacobi transformation, where the rotation angle is determined by (cf. (10.6.21))

$$\tau \equiv \cot 2\theta = (\|a_q\|_2^2 - \|a_p\|_2^2)/(2a_q^T a_p), \quad a_q^T a_p \neq 0.$$

Alternatively, we can first compute the 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 2×2 SVD $R = U\Sigma V^T$. then 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 (10.6.29).

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. Hence, the rate of convergence is ultimately quadratic, also for multiple singular values. Note that the one-sided Jacobi SVD will by construction have U orthogonal to working accuracy, but loss of orthogonality in V may occur. Therefore, the columns of V should be reorthogonalized using a Gram–Schmidt process at the end.

The one-sided method can be applied to a general real (or complex) matrix $A \in \mathbf{R}^{m \times n}$, $m \geq n$, but an initial QR factorization should be performed to speed up convergence. If this is performed with *row and column pivoting*, then high relative accuracy can be achieved for matrices A that are *diagonal scalings of a well-conditioned matrix*, that is which can be decomposed as

$$A = D_1 B D_2,$$

where D_1 , D_2 are diagonal and B well-conditioned. It has been demonstrated that if pre-sorting the rows after decreasing norm $\|a_{i,:}\|_\infty$ and then using column pivoting only gives equally good results. 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 noninteracting 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, and a sweep needs at least $n - 1$ such parallel steps. In practice, with the cyclic Jacobi method not 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 $O(\log n)$.

The alternative algorithm for the SVD of 2×2 upper triangular matrix below always gives high *relative accuracy* in the singular values and vectors, has been developed by Demmel and Kahan, and is based on the relations in Problem 10.6.5.

Review Questions

- 6.1** What is the asymptotic speed of convergence for the classical Jacobi method? Discuss the advantages and drawbacks of Jacobi methods compared to the QR algorithm.
- 6.2** There are two different Jacobi-type methods for computing the SVD were developed. What are they called? What 2×2 subproblems are they based on?
- 6.3** (a) Describe the method of spectrum slicing for determining selected eigenvalues of a real symmetric matrix A .
 (b) How is the method of spectrum slicing applied for computing singular values?

Problems

- 6.1** Implement Jacobi's algorithm, using the stopping criterion (10.6.22) with $\text{tol} = 10^{-12}$. 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 steps are used?

6.2 Suppose 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 at a certain step of the Jacobi algorithm. Estimate the eigenvalues of \tilde{A} as accurately as possible using the Geršgorin circles with a suitable diagonal transformation, see Problem 10.3.3.

6.3 Jacobi-type methods can also be constructed for 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 equation (10.6.21) 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.)

6.4 Let $A \in \mathbf{C}^{2 \times 2}$ be a given matrix, and U a unitary matrix of the form in Problem 10.6.3. Determine U so that the matrix $B = U^{-1}AU$ becomes upper triangular, that is, a Schur decomposition of A . 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 .

6.5 (a) Use one Givens rotation to transform the matrix

$$A = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix},$$

to tridiagonal form.

(b) Compute the largest eigenvalue of A , using spectrum slicing on the tridiagonal form derived in (a). Then compute the corresponding eigenvector.

6.6 Show that (10.6.13) can be written

$$\hat{d}_k = \alpha_k - \frac{\beta_k^2}{\hat{d}_{k-1}} \frac{(1 + \epsilon_{1k})(1 + \epsilon_{2k})}{(1 + \epsilon_{3,k-1})(1 + \epsilon_{4,k-1})} - \frac{\tau}{(1 + \epsilon_{3k})}, \quad k = 1 : n,$$

where we have put $\bar{d}_k = \hat{d}_k(1 + \epsilon_{3k})(1 + \epsilon_{4k})$, and $|\epsilon_{ik}| \leq u$. Conclude that since $\text{sign}(\hat{d}_k) = \text{sign}(\bar{d}_k)$ the computed number \bar{n} is the exact number of eigenvalues a tridiagonal matrix A' whose elements satisfy

$$|\alpha'_k - \alpha_k| \leq u|\tau|, \quad |\beta'_k - \beta_k| \leq 2u|\beta_k|.$$

- 6.7** To compute the SVD of a matrix $A \in \mathbf{R}^{m \times 2}$ we can first reduce A to upper triangular form by a QR decomposition

$$A = (a_1, a_2) = (q_1, q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}.$$

Then, as outlined in Golub and Van Loan [277, 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. Derive the details of this algorithm!

- 6.8** Show that if Kogbetliantz's method is applied to a triangular matrix then after one sweep of the row cyclic algorithm (10.6.23) an upper (lower) triangular matrix becomes lower (upper) triangular.
- 6.9** In the original divide and conquer algorithm by Cuppen [127] for symmetric tridiagonal matrices a different splitting is used than in Section 10.6.1. The matrix is split into two smaller matrices T_1 and T_2 as follows:

$$T = \left(\begin{array}{ccc|c} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \beta_3 & \\ \ddots & \ddots & \ddots & \\ & \beta_k & \alpha_k & \beta_{k+1} \\ \hline & & \beta_{k+1} & \alpha_{k+1} & \beta_{k+2} \\ & & & \ddots & \ddots & \ddots \\ & & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & & \beta_n & \alpha_n & \end{array} \right) \\ = \begin{pmatrix} T_1 & \beta_{k+1} e_k e_1^T \\ \beta_{k+1} e_1 e_k^T & T_2 \end{pmatrix}.$$

- (a) Let \tilde{T}_1 be the matrix T_1 with the element α_k replaced by $\alpha_k - \beta_{k+1}$ and \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 10.2.12 to develop a divide and conquer algorithm for the eigenproblem of T .

10.7 Generalized Eigenvalue Problems

10.7.1 Canonical Forms

Matrix eigenvalue problems often have the form of a **generalized eigenvalue problem**. Given a matrix pair A and B of order n one wants to find scalars λ

and nonzero vectors x so that

$$Ax = \lambda Bx. \quad (10.7.1)$$

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**.⁵⁷ 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

$$p(\lambda) = \det(A - \lambda B) = 0. \quad (10.7.2)$$

In contrast to the standard eigenvalue problem the characteristic polynomial $p(\lambda)$ can have degree less than n . 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 .

Example 10.7.1.

The characteristic equation

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}.$$

is $\det(A - \lambda B) = 1 - \lambda = 0$. There is one eigenvalue $\lambda = \infty$ corresponding to the eigenvector e_1 . This corresponds to the zero eigenvalues of the pencil $B - \lambda A$.

For the ordinary eigenvalue problem eigenvalues are preserved under similarity transformations. The corresponding transformations for the generalized eigenvalue problem are called **equivalence transformations**.

Definition 10.7.1.

Let (A, B) be a matrix pencil and let S and T be nonsingular matrices. Then the two 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 $\det(S) \det(T) \neq 0$. Further, the eigenvectors of two equivalent pencils are related.

If A and B are real symmetric, then symmetry is preserved under congruence transformations in which $S = T^T$. The two pencils are then said to be **congruent**. Of particular interest are orthogonal congruence transformations, $S = Q^T$ and

⁵⁷The word “pencil” comes from optics and geometry, and is used for any one parameter family of curves, matrices, etc.

$T = Q$, where U is a unitary. Such transformations are stable since they preserve the 2-norm,

$$\|Q^T A Q\|_2 = \|A\|_2, \quad \|Q^T B Q\|_2 = \|B\|_2.$$

There is a canonical forms for regular matrix pencils that corresponds the Jordan canonical form (see Theorem 10.1.11). We state this without proof.

Theorem 10.7.2 (Kronecker's Canonical Form).

Let $A - \lambda B \in \mathbf{C}^{n \times n}$ be a regular matrix pencil. Then there are nonsingular matrices $X, Z \in \mathbf{C}^{n \times n}$, such that $X^{-1}(A - \lambda B)Z = \hat{A} - \lambda \hat{B}$, where

$$\begin{aligned}\hat{A} &= \text{diag}(J_{m_1}(\lambda_1), \dots, J_{m_s}(\lambda_s), I_{m_{s+1}}, \dots, I_{m_t}), \\ \hat{B} &= \text{diag}(I_{m_1}, \dots, I_{m_s}, J_{m_{s+1}}(0), \dots, J_{m_t}(0)).\end{aligned}\tag{10.7.3}$$

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 $\sum_{i=1}^t m_i = n$.

As for 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 are may be ill conditioned. Of more computational interest is the generalization of the Schur decomposition (Theorem 10.1.13), which can be obtained by a unitary transformation.

Theorem 10.7.3 (Generalized Schur Decomposition).

Let $A - \lambda B \in \mathbf{C}^{n \times n}$ be a regular matrix pencil. Then there exist unitary matrices U and V so that

$$U^H A V = T_A, \quad U^H B V = T_B,\tag{10.7.4}$$

are upper triangular. The eigenvalues of the pencil are the ratios of the diagonal elements of T_A and T_B .

Proof. Stewart [551, 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 $Av \neq 0$, set $u = Av/\|Av\|_2$. Let $U = (u \ U_\perp)$ and $V = (v \ V_\perp)$ be unitary. Then

$$(u, U_\perp)^H A (v, V_\perp) = \begin{pmatrix} u^H A v & u^H A V_\perp \\ U_\perp^H A v & U_\perp^H A V_\perp \end{pmatrix} = \begin{pmatrix} \sigma_1 & s_{12}^H \\ 0 & \hat{A} \end{pmatrix}.$$

Here the (2,1)-block of the matrix is zero since Av is orthogonal to U_\perp . Similarly

$$(u, U_\perp)^H B (v, V_\perp) = \begin{pmatrix} u^H B v & u^H B V_\perp \\ U_\perp^H B v & U_\perp^H B V_\perp \end{pmatrix} = \begin{pmatrix} \tau_1 & t_{12}^H \\ 0 & \hat{B} \end{pmatrix}.$$

Here the (2,1)-block of the matrix is zero since if $Bv \neq 0$ it is 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 2×2 diagonal blocks corresponding to complex conjugate eigenvalues.

Suppose that in the generalized Schur form (10.7.4)

$$\text{diag}(T_A) = \text{diag}(\sigma_1, \dots, \sigma_n), \quad \text{diag}(T_B) = \text{diag}(\tau_1, \dots, \tau_n).$$

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 matrix pencil $A - \lambda B$ correspond to zero eigenvalues of $B - \mu A$ and vice versa. It is often convenient to represent the eigenvalue 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. Since the role of A and B are interchangeable it is natural to require that the measure is invariant under reciprocation. The **chordal metric**

$$\xi(\lambda, \mu) = \frac{|\lambda - \mu|}{\sqrt{1 + \lambda^2}\sqrt{1 + \mu^2}} \quad (10.7.5)$$

is the appropriate choice. If the eigenvalues are represented by (α, β) and (γ, δ) , respectively, then the distance in the chordal metric (10.7.5) becomes

$$\xi = \frac{|\alpha\delta - \beta\gamma|}{\sqrt{\alpha^2 + \beta^2}\sqrt{\gamma^2 + \delta^2}}. \quad (10.7.6)$$

This formula is valid in general, even when β or δ equals zero.

10.7.2 Reduction to Standard Form

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. However, it may be possible to find a shift μ such that $B + \gamma A$ is well conditioned. Then if μ is an eigenvalue of the standard eigenvalue problem

$$(B + \gamma A)^{-1}Ax = \mu x, \quad \lambda = \mu/(1 + \gamma\mu).$$

By symmetry we could also shift the matrix A and work with the matrix pair $(A + \gamma_1 B)$ and $(B + \gamma_2 A)$.

If B is nonsingular and y^H a left eigenvector of the standard eigenvalue problem for $B^{-1}A$, then the Rayleigh quotient is

$$\rho = \frac{y^H B^{-1} Ax}{y^H x} = \frac{z^H Ax}{z^H Bx},$$

where $z^H = y^H B^{-1}$, is a left eigenvector of $Ax = \lambda B$. If B is nonsingular, we can use the pair of numbers $(z^H Ax, z^H Bx)$ to represent the Rayleigh quotient.

Of particular interest is the case when the problem can be reduced to a symmetric 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 10.7.4.

Let (A, B) be a Hermitian pair and B positive definite. Then there is a nonsingular matrix X such that

$$X^H AX = \Lambda, \quad X^H BX = I \quad (10.7.7)$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ are the eigenvalues of the pencil $A - \lambda B$.

Proof. Since B is positive definite its eigenvalues are real positive, and we can write $B = V\Sigma^2 V^H$. Let $B^{1/2} = V\Sigma^{1/2} V^H$ be the positive definite square root and let $B^{-1/2}AB^{-1/2}$ have the spectral decomposition $U\Lambda U^H$. Then $X = B^{-1/2}U$ 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 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. For a proof see Parlett [478, Sec. 15-2] (cf. also Problem 10.7.1). Thus, even if A and B are Hermitian the generalized eigenvalue problems embodies all the difficulties of the unsymmetric standard eigenvalue problem.

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 $s = \sin \phi$, $c = \cos \phi$. This shows that the rotated pencil $(A_\phi, B_\phi) = (cA - sB, sA + cB)$ has the same eigenvectors as (A, B) , and that the eigenvalues are related by

$$\lambda_\phi = \frac{c\lambda - s}{s\lambda + c}, \quad \lambda = \frac{s + c\lambda_\phi}{c - s\lambda_\phi}. \quad (10.7.8)$$

For the reduction to a Hermitian standard eigenvalue problem to be applicable, it suffices that B_ϕ is positive definite for some real ϕ .

Definition 10.7.5.

*The Hermitian pair (A, B) is a **definite pair** if the **Crawford number***

$$\gamma(A, B) = \min_{\substack{x \in \mathbb{C}^n \\ \|x\|_2=1}} |x^H(A + iB)x| \equiv \min_{\substack{x \in \mathbb{C}^n \\ \|x\|_2=1}} \sqrt{(x^H Ax)^2 + (x^H Bx)^2} > 0. \quad (10.7.9)$$

The generalized eigenproblem $Ax = \lambda Bx$ is definite if (A, B) is definite.

Note that $x^H(A + iB)x$, $x \in \mathbf{C}^n$ is the field of values of the matrix $A + iB$ and that $x^H Ax$ and $x^H Bx$ are real numbers. Further, since

$$A_p hi + iB_\phi = e^{i\phi}(A + iB),$$

the Crawford number is the same for any rotated pair.

Theorem 10.7.6 ([552, Theorem VI.1.18]).

Let (A, B) be a definite pair. Then there exists a real number ϕ , such that the matrix $B_\phi - sA + cB$ 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 10.7.5. \square

Unfortunately, determining whether a given matrix pair is definite is not easy. An algorithm by Crawford and Moon [125] for finding such a linear combination has been improved and tested by Guo, Higham, and Tisseur [295].

For a definite eigenvalue problem perturbation bounds for the eigenvalues and eigenvectors can be derived. The Crawford number $\gamma(A, B)$ plays a key role in these error bounds. 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 a number of perturbation theorems for Hermitian matrices can be generalized to definite problems; see [552, Sec. VI.3].

10.7.3 Methods for Generalized Eigenvalue Problems

In many important applications $A, B \in \mathbf{R}^{n \times n}$ are symmetric and B is positive definite. In this case the problem can be treated by the explicit reduction to be reduced to a standard eigenvalue problem for a real matrix. Compute the Cholesky factorization, preferable with complete pivoting,

$$P^T BP = LD^2L^T, \quad (10.7.10)$$

where 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

$$C = D^{-1}L^{-1}P^T APL^{-T}D^{-1}, \quad y = DL^T P^T x. \quad (10.7.11)$$

Any method for solving the real symmetric eigenvalue problem, e.g. the QR algorithm can be applied to (10.7.11). If instead A is positive definite a similar reduction applies. In both cases the eigenvalues of the generalized eigenvalue problem $Ax = \lambda Bx$ will be real. Moreover, the eigenvectors can be chosen to be B -orthogonal

$$x_i^T B x_j = 0, \quad i \neq j.$$

This generalizes the standard symmetric case for which $B = I$.

Computing the Cholesky decomposition $B = LL^T$ and forming $C = (L^{-1}A)L^{-T}$ takes about $5n^3/12$ flops if symmetry is used, see Wilkinson and Reinsch, Contribution II/10, [614]. If eigenvectors are not wanted, then the transform matrix L need not be saved.

When B is ill-conditioned then the eigenvalues λ may vary widely in magnitude, and a small perturbation in B can correspond to large perturbations in the eigenvalues. Surprisingly, well-conditioned eigenvalues are often given accurately in spite of the ill-conditioning of B . Typically, L will have elements in its lower part. This will produce a matrix $(L^{-1}A)L^{-T}$ which is graded so that the large elements appear in the lower right corner. Hence, a reduction to tridiagonal form should work from bottom to top and the QL-algorithm should be used.

[Wilkinson and Reinsch [614, p. 310]]

Example 10.7.2.

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 $O(10^4)$. 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.

There are a number of related problems involving A and B , which can also be reduced to standard form; see Martin and Wilkinson [427]. For example, $ABx = \lambda x$ is equivalent to

$$(L^T A L)y = \lambda y, \quad y = L^T x. \tag{10.7.12}$$

If A and B are symmetric band matrices and $B = LL^T$ positive definite, then although L inherits the bandwidth of A the matrix $C = (L^{-1}A)L^{-T}$ will in general be a full matrix. Hence, in this case it may not be practical to form C . Crawford [124] has devised an algorithm for reduction to standard form which interleaves orthogonal transformations in such way that the matrix C retains the bandwidth of A , see Problem 10.7.2.

The power method and inverse iteration can both be extended to the generalized eigenvalue problems. Starting with some q_0 with $\|q_0\|_2 = 1$, these iterations

now become

$$\begin{aligned} B\hat{q}_k &= Aq_{k-1}, \quad q_k = \hat{q}_k / \|\hat{q}_k\|, \\ (A - \sigma B)\hat{q}_k &= Bq_{k-1}, \quad q_k = \hat{q}_k / \|\hat{q}_k\|, \quad k = 1, 2, \dots \end{aligned}$$

respectively. Note that $B = I$ gives the iterations in equations (10.6.21) and (10.6.24).

The Rayleigh Quotient Iteration also extends to the generalized eigenvalue problem: For $k = 0, 1, 2, \dots$ compute

$$(A - \rho(q_{k-1})B)\hat{q}_k = Bq_{k-1}, \quad q_k = \hat{q}_k / \|q_k\|_2, \quad (10.7.13)$$

where the (generalized) Rayleigh quotient of x is

$$\rho(x) = \frac{x^H Ax}{x^H Bx}.$$

In the symmetric definite case the Rayleigh Quotient Iteration has asymptotically cubic convergence and the residuals $\|(A - \mu_k B)x_k\|_{B^{-1}}$ decrease monotonically.

The Rayleigh Quotient method is advantageous to use when A and B have band structure, since it does not require an explicit reduction to standard form. The method of spectrum slicing can be used to count eigenvalues of $A - \lambda B$ in an interval.

Theorem 10.7.7.

Let $A - \sigma B$ have the Cholesky factorization

$$A - \mu B = LDL^T, \quad D = \text{diag}(d_1, \dots, d_n),$$

where L is unit lower triangular. If B is positive definite then the number of eigenvalues of A greater than μ equals the number of positive elements $\pi(D)$ in the sequence d_1, \dots, d_n .

Proof. The proof follows from Sylvester's Law of Inertia (Theorem 7.3.9 and the fact that by Theorem 10.7.2 A and B are congruent to D_A and D_B with $\Lambda = D_A D_B^{-1}$). \square

The **QZ algorithm** by Moler and Stewart [442] is a generalization of the implicit QR algorithm. It does not preserve symmetry and is therefore more expensive than the special algorithms for the symmetric case.

In the first step of the QZ algorithm the matrix A a sequence of equivalence transformations is used to reduce A to upper Hessenberg form and simultaneously B to upper triangular form. Note that this corresponds to a reduction of AB^{-1} to upper Hessenberg form. This can be performed using standard Householder transformations and Givens rotations as follows. This step begins by finding an orthogonal matrix Q 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

$$Q^T(A, B)Z = (Q^T AZ, Q^T BZ) = (H_A, R_B).$$

The elements in $Q^T A$ are zeroed starting in the first column working from bottom up. This process is then repeated on the columns $2 : n$. Infinite eigenvalues, which correspond to zero diagonal elements of R_B can be eliminated at this step.

After the initial transformation the implicit shift QR algorithm is applied to $H_A R_B^{-1}$, but without forming this 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 A is eliminated by pre-multiplication by a plane rotation:

$$\begin{array}{c} \rightarrow \begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix}, \quad \rightarrow \begin{pmatrix} b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ \hat{b} & b & b & b & b \end{pmatrix}. \\ \rightarrow \end{array}$$

This introduces a new nonzero element \hat{b} in the $(5, 4)$ position in B , which is zeroed by a post-multiplication by a plane rotation

$$\begin{array}{c} \downarrow \downarrow \quad \downarrow \downarrow \\ \begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix}, \quad \begin{pmatrix} b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ \hat{b} & b & b & b & b \end{pmatrix}. \\ \end{array}$$

All remaining steps in the reduction of A to upper Hessenberg form are similar. The complete reduction requires about $34n/3$ flops. If eigenvectors are to be computed, the product of the post-multiplications 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 2×2 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 that in the implicit QR algorithm and will not be described here. The first Householder transformation is determined by the shift. When (H_A, R_B) is pre-multiplied by this new nonzero elements are introduced. This “bulge” is chased down the matrix by pre- and post-multiplication by Householder and plane rotations until the upper Hessenberg and triangular forms are restored.

The matrix H_A will converge to upper triangular form and the eigenvalues of $A - \lambda B$ will be obtained as ratios of diagonal elements of the transformed H_A and R_B . For a more detailed description of the algorithm see Stewart [543, Chapter 7.6].

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.

10.7.4 Quadratic and Structured Eigenvalue Problems

The quadratic eigenvalue problem (QEP) is to find scalars λ and nonzero vectors x such

$$Q(\lambda)x = 0, \quad Q(\lambda) = \lambda^2 M + \lambda C + K, \quad (10.7.14)$$

where $M, C, K \in \mathbb{C}^{nn}$. This problem 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 $q(t), f(t)$ are vectors of order n . Such equations governs electrical and mechanical oscillations. The matrix M represents the mass, matrix K the resistance to displacements (stiffness) and the matrix C the damping. The vector $f(t)$ is the external force.

It arises, e.g., in the dynamical analysis of mechanical systems, in acoustics, and in the linear stability analysis of fluid mechanics. The characteristic equation

$$\det(C(\lambda)) = 0$$

will have degree $2n$ and the eigenvalue problem (10.7.14) 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.

Some properties of special classes of QEP can be shown.

- If M is nonsingular, then there are $2n$ finite eigenvalues.
- If M, C , and K are real or Hermitian then $Q(\lambda)$ is self-adjoint $Q(\lambda) = Q(\bar{\lambda})^H$ for all $\lambda \in \mathbb{C}$. Then the eigenvalues are real or come in pairs $\lambda, \bar{\lambda}$.
- 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, K are Hermitian positive semidefinite, then $\Re(\lambda) \leq 0$.

Many methods for solving QEP starts with a linearization, i.e., a reduction to a generalized eigenvalue problem $Ax - \lambda Bx = 0$. This can be done by making the substitution $u = \lambda x$ and rewriting the equation $(\lambda^2 M + \lambda C + K)x = 0$ as $\lambda Mu + Cu + Kx = 0$, or

$$\begin{pmatrix} 0 & I \\ -K & -C \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} - \lambda \begin{pmatrix} I & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} = 0.$$

More generally, in the first companion form we set

$$A = \begin{pmatrix} 0 & N \\ -K & -C \end{pmatrix}, \quad B = \begin{pmatrix} N & 0 \\ 0 & M \end{pmatrix}. \quad (10.7.15)$$

where N is any nonsingular matrix. The second companion form is

$$A = \begin{pmatrix} -K & 0 \\ 0 & N \end{pmatrix}, \quad B = \begin{pmatrix} C & M \\ N & 0 \end{pmatrix}. \quad (10.7.16)$$

Many eigenvalue problems that occur in applications have some form of symmetry, which implies that the spectrum has certain properties. We have already seen that a Hermitian matrix, $A^H = A$, has real eigenvalues. Unless such a structure is preserved by the numerical method, useless results may be produced.

We now present some other important structured eigenvalue problems. A **symplectic matrix** if it satisfies the condition $A^HJA = J$. Let J be the $2n \times 2n$ skew-symmetric matrix

$$J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}. \quad (10.7.17)$$

Note that $J^2 = I$ and therefore the matrix J has determinant equal to ± 1 . It can be shown that $\det(J) = +1$ and its inverse equals $J^{-1} = J^T = -J$. A $2n \times 2n$ matrix H is called J -Hermitian or **Hamiltonian**⁵⁸ if $(JH)^H = JH$. If

$$H = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

is real with $n \times n$ blocks, then H is Hamiltonian provided that B and C are symmetric and $A + D^T = 0$. In the vector space of all $2n \times 2n$ matrices, Hamiltonian matrices form a $2n^2 + n$ vector subspace. Some further properties of Hamiltonian matrices are

- The transpose of a Hamiltonian matrix is Hamiltonian.
- The trace of a Hamiltonian matrix is zero.

Eigenvalue problems for Hamiltonian matrices occur, e.g., in linear quadratic optimal control problems. The eigenvalues of a Hamiltonian matrix are symmetric about the imaginary axis in the complex plane, i.e., they occur in pairs $\lambda, -\bar{\lambda}$ and the corresponding eigenvectors are related by

$$Hx = \lambda x \Rightarrow (Jx)^H H = -\bar{\lambda}(Jx)^H. \quad (10.7.18)$$

A $2n \times 2n$ (real or complex) matrix M is called J -orthogonal or **symplectic** if $M^TJM = J$ and J -unitary if $M^HJM = J$. Since $M^TJM^T = JJ^T = I$ it follows

⁵⁸William Rowan Hamilton (1805–1865) Irish mathematician, professor at Trinity College, Dublin. He made important contributions to classical mechanics, optics, and algebra. His greatest contributions 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.

that $(MJ^T)^{-1} = M^T J$. Hence, every symplectic matrix is invertible and its inverse is given by

$$M^{-1} = J^{-1} M^T J.$$

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 eigenvalues of a symplectic matrix occur in pairs $\lambda, 1/\lambda$ and the eigenvectors are related by

$$Mx = \lambda x \Rightarrow (Jx)^T M = 1/\lambda (Jx)^T. \quad (10.7.19)$$

A symplectic vector space is a $2n$ -dimensional vector space equipped with a skew-symmetric bilinear form. Hamiltonian and symplectic matrices are related. The exponential of a Hamiltonian matrix is symplectic and the logarithm of a symplectic matrix is Hamiltonian. Special methods for solving Hamiltonian and symplectic eigenvalue problems are given in [420, 198].

10.7.5 Generalized Singular Value Decomposition

Let $A \in \mathbf{R}^{m \times n}$ and $B \in \mathbf{R}^{p \times n}$ be two matrices with the same number of columns. The **generalized singular value decomposition** (GSVD) of A and B is related to the generalized eigenvalue problem

$$A^T A x = \lambda B^T B x. \quad (10.7.20)$$

The GSVD has applications to, e.g., constrained least squares problems. As in the case of the SVD, the formation of $A^T A$ and $B^T B$ should be avoided. In the theorems below we assume for notational convenience that $m \geq n$. The GSVD was first proposed by Van Loan [588]. A form more suitable for numerical computation was suggested by Paige and Saunders [468]. The implementation used in LAPACK is described in [26] and [22].

Theorem 10.7.8 (The Generalized Singular Value Decomposition (GSVD)).

Let $A \in \mathbf{R}^{m \times n}$, $m \geq n$, and $B \in \mathbf{R}^{p \times n}$ be given matrices. Assume that

$$\text{rank}(M) = k \leq n, \quad M = \begin{pmatrix} A \\ B \end{pmatrix}.$$

Then there exist orthogonal matrices $U_A \in \mathbf{R}^{m \times m}$ and $U_B \in \mathbf{R}^{p \times p}$ and a matrix $Z \in \mathbf{R}^{k \times n}$ of rank k such that

$$U_A^T A = \begin{pmatrix} D_A \\ 0 \end{pmatrix} Z, \quad U_B^T B = \begin{pmatrix} D_B & 0 \\ 0 & 0 \end{pmatrix} Z, \quad (10.7.21)$$

where

$$D_A = \text{diag}(\alpha_1, \dots, \alpha_k), \quad D_B = \text{diag}(\beta_1, \dots, \beta_q), \quad q = \min(p, k).$$

Further, we have

$$\begin{aligned} 0 \leq \alpha_1 \leq \dots \leq \alpha_k \leq 1, \quad 1 \geq \beta_1 \geq \dots \geq \beta_q \geq 0, \\ \alpha_i^2 + \beta_i^2 = 1, \quad i = 1, \dots, q, \quad \alpha_i = 1, \quad i = q+1, \dots, k, \end{aligned}$$

and the singular values of Z equal the nonzero singular values of M .

Proof. We now give a constructive proof of Theorem 10.7.8 using the CS decomposition. Let the SVD of M be

$$M = \begin{pmatrix} A \\ B \end{pmatrix} = Q \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} P^T,$$

where Q and P are orthogonal matrices of order $(m+p)$ and n , respectively, and

$$\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_k), \quad \sigma_1 \geq \dots \geq \sigma_k > 0.$$

Set $t = m + p - k$ and partition Q and P as follows:

$$Q = \left(\underbrace{\begin{matrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{matrix}}_k \right) \}_{p}^m, \quad P = \left(\underbrace{\begin{matrix} P_1 \\ P_2 \end{matrix}}_t \right)_{n-k}.$$

Then the SVD of M can be written

$$\begin{pmatrix} A \\ B \end{pmatrix} P = \begin{pmatrix} AP_1 & 0 \\ BP_1 & 0 \end{pmatrix} = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} (\Sigma_1 \ 0). \quad (10.7.22)$$

Now let

$$Q_{11} = U_A \begin{pmatrix} C \\ 0 \end{pmatrix} V^T, \quad Q_{21} = U_B \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} V^T$$

be the CS decomposition of Q_{11} and Q_{21} . Substituting this into (10.7.22) we obtain

$$\begin{aligned} AP &= U_A \begin{pmatrix} C \\ 0 \end{pmatrix} V^T (\Sigma_1 \ 0), \\ BP &= U_B \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} V^T (\Sigma_1 \ 0), \end{aligned}$$

and (10.7.21) follows with

$$D_A = C, \quad D_B = S, \quad Z = V^T (\Sigma_1 \ 0) P^T.$$

Here $\sigma_1 \geq \dots \geq \sigma_k > 0$ are the singular values of Z . \square

When $B \in \mathbf{R}^{n \times n}$ is square and nonsingular the GSVD of A and B corresponds to the SVD of AB^{-1} . However, when A or B is ill-conditioned, then 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.

10.7.6 Pseudospectra of Nonnormal Matrices

A matrix $A \in \mathbf{C}^{n \times n}$ is normal if $A^H A = AA^H$. Normal matrices have a complete set of orthogonal eigenvectors and hence the eigenvalue problem is perfectly conditioned. For a nonnormal matrix the transformation to diagonal form may be very

ill-conditioned or a complete set of linearly independent eigenvectors may not exist. In applications one often works with a family of matrices indexed by a physical parameter such that the nonnormality increases unboundedly as the parameter approaches some limit. For such problems it may not be reasonable to analyze the problem in terms of eigenvalues and eigenvectors. The concept of **pseudospectra** of a matrix is a tool to deal with such problems.

A complex number z is an eigenvalue of a matrix $A \in \mathbf{C}^{n \times n}$ if and only if the matrix $zI - A$ is singular. The **resolvent**

$$R(z) = (zI - A)^{-1} \quad (10.7.23)$$

regarded as function of z has singularities at the eigenvalues of A . Away from these eigenvalues the resolvent is an analytic function of z .

Definition 10.7.9. For $\epsilon > 0$ the ϵ -**pseudospectrum** of a matrix $A \in \mathbf{C}^{n \times n}$ is the subset of the complex plane

$$\Lambda_\epsilon = \{z \in \mathbf{C} \mid \|(zI - A)^{-1}\|_2 \geq \epsilon^{-1}\} \quad (10.7.24)$$

If A is normal, then Λ_ϵ equals the set of points in \mathbf{C} at a distance less than or equal to ϵ . For a nonnormal matrix it can be much larger.

Other equivalent definitions are

$$\Lambda_\epsilon = \{z \in \mathbf{C} \mid z \in \Lambda(A + E), \|E\|_2 \leq \epsilon\} \quad (10.7.25)$$

Thus z belongs to the pseudospectrum of A if and only if it is in the spectrum of some perturbed matrix $(A + \Delta A)$ with $\|\Delta A\|_2 \leq \epsilon$.

More suited to computation is the definition

$$\Lambda_\epsilon = \{z \in \mathbf{C} \mid \sigma_{min}(zI - A) \leq \epsilon\}. \quad (10.7.26)$$

A straightforward way to compute the pseudospectrum Λ_ϵ is to evaluate $\sigma_{min}(zI - A)$ for points z on a grid in the complex plane using a standard QR-SVD algorithm (see Section 10.5.3) and then plot level curves. Alternatively, an iterative algorithm could be used for computing the square root of the smallest eigenvalue of $(zI - A)^H(zI - A)$. This can be much faster than computing the entire SVD for each grid point. If an LU factorization of the matrix $zI - A$ is available it is better to compute the largest eigenvalue of $((zI - A)^H(zI - A))^{-1}$.

Another method to obtain an approximation to Λ_ϵ is to compute the eigenvalues of a sequence of perturbed matrices $(A + E)$, with $\|E\|_2 \leq \epsilon$. This yields a number of pseudo-eigenvalues of A and plotting these gives a discrete approximation of the pseudospectrum. However, this requires the computation of the full spectrum of a sequence of nonnormal matrices, which is a hard problem.

Closely related to pseudospectra is the concept of the **field of values** of a matrix.

Definition 10.7.10.

The field of values $F(A)$ of a matrix A is the set of all possible Rayleigh quotients

$$F(A) = \{z^H A z \mid z \in \mathbb{C}^n, \|z\|_2 = 1\}. \quad (10.7.27)$$

$F(A)$ is a closed convex set that contains the convex hull of the eigenvalues of A . 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

$$\rho(x) = \sum_{i=1}^n \lambda_i |\xi_i|^2 / \sum_{i=1}^n |\xi_i|^2,$$

that is, any point in $F(A)$ is a weighted mean of the eigenvalues of A . Thus, for a normal matrix the field of values 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 field of values equals a segment of the imaginary axis.

Associated with the field of values is the **numerical radius**

$$r(A) = \max\{|w| \mid w \in F(A)\}, \quad (10.7.28)$$

In general the field of values of a matrix A may contain complex values even if its eigenvalues are real. For a detailed presentation of the field of values see [339, Chapter 2]. A MATLAB routine for computing the field of values is included in the matrix computation toolbox by Higham [329]. 1.0”,

The numerical radius does not differ much from $\|A\|_2$ since it can be shown to satisfy

$$\frac{1}{2} \|A\|_2 \leq r(A) \leq \|A\|_2. \quad (10.7.29)$$

For a diagonalizable matrix $A = XDX^{-1}$ it holds that

$$\frac{r(A)}{\rho(A)} \leq \kappa_2(X), \quad (10.7.30)$$

where $\rho(A)$ is the spectral radius. Thus, if the numerical radius is much bigger than the spectral radius, the matrix A is far from normal.

The **numerical abscissa**

$$\alpha(A)_2 = \max_{z \in F(A)} \Re z. \quad (10.7.31)$$

where $F(A)$ is the field of values of A , This is trivially at least as large as the **spectral abscissa**

$$\alpha_F(A) = \max_{i=1:n} \Re(\lambda_i). \quad (10.7.32)$$

Review Questions

- 7.1** What is meant by a regular matrix pencil? Give examples of a singular pencil, and a regular pencil that has an infinite eigenvalue.
- 7.2** Formulate a generalized Schur decomposition. Show that the eigenvalues of the pencil are easily obtained from the canonical form.
- 7.3** Let A and B be real symmetric matrices, and B also positive definite. Show that there is a congruence transformation that diagonalizes the two matrices simultaneously. How is the Rayleigh Quotient iteration generalized to this type of eigenvalue problems, and what is its order of convergence?
- 7.4** Let $(M\lambda^2 + C\lambda + k)x = 0$, be a quadratic eigenvalue problem, where M , C , and K are matrices in $\mathbf{C}^{n \times n}$
- This eigenvalue problem is related to a differential equation. Which?
 - Give a linearization of the eigenvalue problem, which reduces it to a generalized eigenvalue problem $Ax = \lambda Bx$, for some matrices A , B of order $2n$.

Problems

- 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 both real and symmetric.

- 7.2** Show that the chordal metric

$$\frac{|\lambda - \mu|}{\sqrt{1 + \lambda^2}\sqrt{1 + \mu^2}}$$

has the following geometric interpretation. Let S be a circle with diameter one 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 on the circle. Define $\tilde{\mu}$ similarly. Then the chordal distance equals the shorter chord between the points $\tilde{\lambda}$ and $\tilde{\mu}$.

- 7.4** (a) Show that

$$\xi(\lambda, \infty) = \frac{1}{\sqrt{1 + \lambda^2}}.$$

- (b) Show that when $|\lambda|, |\mu| \leq 1$ the chordal metric has the property that

$$\xi(\lambda, \infty) \leq |\lambda - \mu| \leq 2\xi(\lambda, \infty).$$

7.5 Let A and B be 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 $L = \Lambda_1 \Lambda_2 \cdots \Lambda_n$, where Λ_{m_k} differs from the unit matrix only in the k th column.
- (b) Consider the recursion

$$A_1 = A, \quad A_{k+1} = Q_k \Lambda_k^{-1} A_k \Lambda_k^{-T} Q_k^T, \quad k = 1 : n.$$

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.)

10.8 Matrix Functions

10.8.1 Convergence of Matrix Power Series

Let $f(z) = \sum_{k=0}^{\infty} a_k z^k$ be an infinite power series, where z is a complex number. Associated with this there is a **circle of convergence** in the z -plane with radius r and center at the origin, such that the series converges absolutely for any $z < r$ and diverges for any $z > r$. (The radius can be 0 or ∞ in extreme cases.) In the interior of the convergence circle, formal operations such as term-wise differentiation and integration with respect to z are valid. To this power series there corresponds an infinite **matrix power series**

$$f(A) = \sum_{k=0}^{\infty} a_k A^k. \quad (10.8.1)$$

Such power series are simple examples of matrix functions. An important question is: when is the matrix power series (10.8.4) convergent? To answer this question we need the following result.

Theorem 10.8.1.

Given a matrix $A \in \mathbf{R}^{n \times n}$ with spectral radius $\rho(A)$. Then for every $\epsilon > 0$, there exists a nonsingular matrix $X = X(\epsilon)$ such that the consistent matrix norm

$$\|A\|_X = \|X^{-1}AX\|_{\infty} = \rho + \epsilon. \quad (10.8.2)$$

Proof. If A is diagonalizable, then we can take X to be the diagonalizing transformation $\Lambda = X^{-1}AX$. Then $\|A\|_X = \|\Lambda\|_{\infty} = \rho$.

In the general case, let $U^H AU = T = \Lambda + N$ be a Schur decomposition of A , where U is unitary and $N = (\eta_{ij})$ strictly upper triangular. Let $X = UD$, where

$$D = \text{diag } (1, \delta, \delta^2, \dots, \delta^{n-1}), \quad \delta \leq \frac{\epsilon}{n-1} \min_{\substack{i=1:n \\ j=i+1:n}} |\eta_{ij}|.$$

Then $X^{-1}AX = D^{-1}TD = \Lambda + D^{-1}ND$, where

$$D^{-1}ND = \begin{pmatrix} 0 & \eta_{12}\delta & \eta_{13}\delta^2 & \cdots & \eta_{1n}\delta^{n-1} \\ 0 & 0 & \eta_{23}\delta & \cdots & \eta_{2n}\delta^{n-2} \\ & & 0 & \cdots & \eta_{3n}\delta^{n-3} \\ & & & \ddots & \vdots \\ & & & & 0 \end{pmatrix}.$$

By the choice of δ we have

$$\|X^{-1}AX\|_\infty \leq \rho(A) + (n-1)\delta \max(\eta_{ij}) \leq \rho(A) + \epsilon. \quad (10.8.3)$$

□

Note that $\|A\|/\kappa(X) \leq \|A\|_X \leq \kappa(X)\|A\|$. If A is diagonalizable, then for every natural number n , we have $\|A^n\|_X \leq \|A\|_X^n = \rho(A)^n$. Hence

$$\|A^n\|_p \leq \kappa(X)\|A^n\|_X \leq \kappa(X)\rho^n.$$

In the general case, the same holds, if ρ and X are replaced by, respectively, $\rho + \epsilon$, $X(\epsilon)$.

In some applications to partial differential equations, one needs to apply this kind of theorem to a *family of matrices* instead of just one individual matrix. This leads to the famous *matrix theorems of Kreiss*, see [386].

We now return to the question of convergence of a matrix power series.

Theorem 10.8.2.

If the infinite power series $f(z) = \sum_{k=0}^{\infty} a_k z^k$ has radius of convergence r , then the matrix power series

$$f(A) = \sum_{k=0}^{\infty} a_k A^k. \quad (10.8.4)$$

converges if $\rho < r$, where $\rho = \rho(A)$ is the spectral radius of A . If $\rho > r$, then the matrix series diverges; the case $\rho = r$ is a “questionable case”.

Proof. By Theorem 7.1.16 the matrix series $\sum_{k=0}^{\infty} a_k A^k$ converges if the series $\sum_{k=0}^{\infty} |a_k| \|A^k\|$ converges. By Theorem 10.8.1, for any $\epsilon > 0$ there is a matrix norm depending on A and ϵ such that $\|A\|_X = \rho + \epsilon$. If $\rho < r$ then we can choose r_1 such that $\rho(A) \leq r_1 < r$, and we have

$$\|A^k\|_X \leq \|A\|_X^k \leq (\rho + \epsilon)^k = O(r_1^k).$$

Here $\sum_{k=0}^{\infty} |a_k| r_1^k$ converges, and hence $\sum_{k=0}^{\infty} |a_k| \|A^k\|$ converges. If $\rho > r$, let $Ax = \lambda mx$ with $|\lambda| = \rho$. Then $A^k x = \lambda^k x$, and since $\sum_{k=0}^{\infty} a_k \lambda^k$ diverges $\sum_{k=0}^{\infty} a_k A^k$ cannot converge. □

It follows from this theorem that a sufficient condition for $\lim_{n \rightarrow \infty} A^n = 0$ is that $\rho(A) < 1$. Note that this is an *asymptotic result, and the behavior of A^n* . For small values of n is better predicted by the **numerical radius**

$$r(A) = \max_{z^H z=1} |z^H A z|, \quad (10.8.5)$$

which satisfies

$$\frac{1}{2} \|A\|_2 \leq r(A) \leq \|A\|_2. \quad (10.8.6)$$

For a diagonalizable matrix $A = X^{-1} \Lambda X$ it holds that

$$\frac{r(A)}{\rho(A)} \leq \frac{\|A\|_2}{\|D\|_2} \leq \kappa_2(X).$$

Thus, the numerical radius can be much bigger than the spectral radius if the eigenvector basis is ill conditioned.

10.8.2 Analytic Matrix Functions

Given a matrix $A \in \mathbf{C}^{n \times n}$ and a complex number z the resolvent operator is defined as $R(z) = (zI - A)^{-1}$. Regarded as function of z the resolvent has singularities at the eigenvalues of A . Away from these eigenvalues the resolvent is an analytic function of z .

For an analytic function f with Taylor expansion $f(z) = \sum_{k=0}^{\infty} a_k z^k$, we define the matrix function by

$$f(A) = \sum_{k=0}^{\infty} a_k A^k. \quad (10.8.7)$$

(Note that this is not the same as a matrix-valued function of a complex variable.) Alternatively, using the resolvent operator

$$f(A) = \int_{\Gamma} (zI - A)^{-1} f(z) dz \quad (10.8.8)$$

where Γ is any contour enclosing the spectrum of A .

If the matrix A is diagonalizable, $A = X \Lambda X^{-1}$, we define

$$f(A) = X \text{diag}(f(\lambda_1), \dots, f(\lambda_n)) X^{-1} = X f(\Lambda) X^{-1}. \quad (10.8.9)$$

This expresses the matrix function $f(A)$ in terms of the function f evaluated at the spectrum of A and is often the most convenient way to compute $f(A)$.

For the case when A is not diagonalizable we first give an explicit form for the k th power of a Jordan block $J_m(\lambda) = \lambda I + N$. Since $N^j = 0$ for $j \geq m$ we get using the binomial theorem

$$J_m^k(\lambda) = (\lambda I + N)^k = \lambda^k I + \sum_{p=1}^{\min(m-1, k)} \binom{k}{p} \lambda^{k-p} N^p, \quad k \geq 1.$$

Since an analytic function can be represented by its Taylor series we are led to the following definition:

Definition 10.8.3.

Suppose that the analytic function $f(z)$ is regular for $z \in D \subseteq \mathbf{C}$, where D is a simply connected region, which contains the spectrum of A in its interior. Let

$$A = X J X^{-1} = X \operatorname{diag}(J_{m_1}(\lambda_1), \dots, J_{m_t}(\lambda_t)) X^{-1}$$

be the Jordan canonical form of A . We then define

$$f(A) = X \operatorname{diag}\left(f(J_{m_1}(\lambda_1)), \dots, f(J_{m_t}(\lambda_t))\right) X^{-1}. \quad (10.8.10)$$

where the analytic function f of a Jordan block is

$$\begin{aligned} f(J_{m_k}) &= f(\lambda_k)I + \sum_{p=1}^{m_k-1} \frac{1}{p!} f^{(p)}(\lambda_k) N^p \\ &= \begin{pmatrix} f(\lambda_k) & f'(\lambda_k) & \cdots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & \ddots & & \vdots \\ & \ddots & f'(\lambda_k) & \\ & & & f(\lambda_k) \end{pmatrix}. \end{aligned} \quad (10.8.11)$$

If A is diagonalizable, $A = X^{-1} \Lambda X$, then for the exponential function we have,

$$\|e^A\|_2 = \kappa(X) e^{\alpha(A)},$$

where $\alpha(A) = \max_i \Re \lambda_i$ is the **spectral abscissa** of A and $\kappa(X)$ denotes the condition number of the eigenvector matrix. If A is normal, then V is orthogonal and $\kappa(V) = 1$.

One can show that for every non-singular matrix T it holds

$$f(T^{-1} A T) = T^{-1} f(A) T. \quad (10.8.12)$$

With this definition, the theory of analytic functions of a matrix variable closely follows the theory of a complex variable. If $\lim_{n \rightarrow \infty} f_n(z) = f(z)$ for $z \in D$, 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 can be obtained, which shows that Definition 10.8.3 is consistent with the more restricted definition (by a power series) given in Theorem 10.8.2.

Theorem 10.8.4.

All identities which hold for analytic functions of one complex variable z for $z \in D \subset \mathbf{C}$, where D is a simply connected region, also hold for analytic functions of one matrix variable A if the spectrum of A is contained in the interior of D . The identities also hold if A has eigenvalues on the boundary of D , provided these are not defective.

Example 10.8.1.

We have, for example,

$$\begin{aligned}\cos^2 A + \sin^2 A &= I, \quad \forall A; \\ \ln(I - A) &= -\sum_{n=1}^{\infty} \frac{1}{n} A^n, \quad \rho(A) < 1; \\ \int_0^{\infty} e^{-st} e^{At} dt &= (sI - A)^{-1}, \quad \operatorname{Re}(\lambda_i) < \operatorname{Re}(s);\end{aligned}$$

For two arbitrary analytic functions f and g which satisfy the condition of Definition, it holds that $f(A)g(A) = g(A)f(A)$. However, when several non-commutative matrices are involved, one can no longer use the usual formulas for analytic functions.

Example 10.8.2.

$e^{(A+B)t} = e^{At} e^{Bt}$ 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

$$e^{(A+B)t} = \sum_{n=0}^{\infty} \frac{t^n}{n!} (A + B)^n.$$

The difference between the coefficients of $t^2/2$ in the two expressions is

$$(A + B)^2 - (A^2 + 2AB + B^2) = BA - AB \neq 0, \quad \text{if } BA \neq AB.$$

Conversely, if $BA = AB$, then it follows by induction that the binomial theorem holds for $(A + B)^n$, and the two expressions are equal.

An alternative definition of matrix functions, due to Sylvester 1883, can be obtained by using polynomial interpolation. Denote by $\lambda_1, \dots, \lambda_t$ the distinct eigenvalues of A and let m_k be the index of λ_k , that is, the order of the largest Jordan block containing λ_k . Then $f(A) = p(A)$, where p is the unique Hermite interpolating polynomial of degree less than $\sum_{k=1}^t m_k$ that satisfies the interpolating conditions

$$p^{(i)}(\lambda_k) = f^{(j)}(\lambda_k), \quad j = 0 : m_k, \quad i = 1 : t. \quad (10.8.13)$$

The function is said to be defined on the spectrum of A if all the derivatives in (10.8.13) exist. Note that the eigenvalues are allowed to be complex. The proof is left to Problem 10.8.10. Formulas for complex Hermite interpolation are given in Volume I, Section 4.3.2.

Note also that if $f(z)$ is analytic inside the closed contour C , and if the whole spectrum of A is inside C , the Cauchy integral definition is (cf. Problem 7.8.11)

$$\frac{1}{2\pi i} \int_C (zI - A)^{-1} f(z) dz = f(A). \quad (10.8.14)$$

For a composite function $f(t) = g(h(t))$ it holds that $f(A) = g(h(A))$, provided that the right hand side is defined.

If a transformation $B = XAX^{-1}$ can be found such that $f(B)$ is easily evaluated then the relation $f(A) = X^{-1}f(B)X$ can be used for numerical computation of $f(A)$. However, it is important that X is not too ill-conditioned. For a Hermitian matrix A there is unitary X such that B is diagonal and the evaluation of $f(B)$ is trivial.

A general matrix A can be reduced to Schur form $A = QTQ^H$, where T is upper triangular. Then $f(T)$ is again upper triangular since it is a polynomial in T . The diagonal elements of $F = f(T)$ are then equal to $f_{ii} = f(t_{ii})$. In the **Schur–Parlett method** the off-diagonal elements are computed using a recurrence relation; see [475]. Since F is a polynomial in T these matrices commute, i.e., $TF - FT = 0$. Equating the (i, j) th element, $i < j$, on both sides of this relation gives

$$\sum_{k=i}^j (t_{ik}f_{kj} - f_{ik}t_{kj}) = 0.$$

Taking out the first and last terms in the sum, this can be rearranged into

$$f_{ij}(t_{ii} - t_{jj}) = t_{ij}(f_{ii} - f_{jj}) + \sum_{k=i+1}^{j-1} (f_{ik}t_{kj} - t_{ik}f_{kj}). \quad (10.8.15)$$

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 $i < k < j$. For $j = 1 + 1$ the sum is empty and

$$f_{i,i+1} = t_{i,i+1} \frac{f_{ii} - f_{i+1,i+1}}{t_{i,i+1} - t_{i+1,i+1}}, \quad i = 2, n$$

gives the elements in the first superdiagonal of F . In the **Schur–Parlett method** the recurrence (10.8.15) is used to compute the off-diagonal elements in $f(T)$ one superdiagonal at a time in $2n^3/3$ flops.

When $t_{ii} = t_{jj}$ for some $i \neq j$, Parlett's recurrence breaks down and it can give inaccurate results when T has close eigenvalues. Then a block version of the Schur–Parlett method can be used; see Higham and Davies [137]. First a Schur decomposition $A = QTQ^T$ is computed. The block triangular Schur form is then reordered in block triangular form so that eigenvalues within each diagonal block T_{ii} are close and those of separate blocks are well separated. The function $F = f(T)$ will then have a similar block structure. If the diagonal blocks $F_{ii} = f(T_{ii})$ are evaluated using some other technique, then the off-diagonal blocks in F can then be computed from the Sylvester equations

$$F_{ii}T_{ij} - T_{ij}F_{jj} = T_{ii}F_{ij} - F_{ij}T_{jj} - \sum_{k=i+1}^{j-1} (F_{ik}T_{kj} - T_{ik}F_{kj}), \quad i \neq j. \quad (10.8.16)$$

These equations have unique solutions and an algorithm for computing F_{ij} is outlined in the proof of Theorem 10.1.19.

10.8.3 Matrix Exponential and Logarithm

The **matrix exponential** e^{At} , where A is a constant matrix, can be defined by the series expansion

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \dots$$

This series converges for all A and t since the radius of convergence of the power series $\sum_{k=0}^{\infty} \|A\|^k t^k / k!$ is infinite. The series can thus be differentiated everywhere and

$$\frac{d}{dt}(e^{At}) = A + A^2t + \frac{1}{2!}A^3t^2 + \dots = Ae^{At}.$$

Hence, $y(t) = e^{At}c \in \mathbf{R}^n$ solves the initial value problem for the linear system of ordinary differential equations with constant coefficients

$$dy(t)/dt = Ay(t), \quad y(0) = c. \quad (10.8.17)$$

Such systems occurs in many physical, biological, and economic processes. Similarly, the functions $\sin(z)$, $\cos(z)$, $\log(z)$, can be defined for matrix arguments from their Taylor series representation.

The matrix exponential and its qualitative behavior has been studied extensively. A wide variety of methods for computing e^A have been proposed; see Moler and Van Loan [441]. Consider the 2 by 2 upper triangular matrix

$$A = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}.$$

The exponential of this matrix is

$$e^{tA} = \begin{cases} \begin{pmatrix} e^{\lambda t} & \alpha \frac{e^{\lambda t} - e^{\mu t}}{\lambda - \mu} \\ 0 & e^{\mu t} \end{pmatrix}, & \text{if } \lambda \neq \mu, \\ \begin{pmatrix} e^{\lambda t} & \alpha t e^{\lambda t} \\ 0 & e^{\mu t} \end{pmatrix}, & \text{if } \lambda = \mu \end{cases}. \quad (10.8.18)$$

When $|\lambda - \mu|$ is small, but not negligible neither of these two expressions are suitable, since severe cancellation will occur in computing the divided difference in the (1,2)-element in (10.8.18). When the same type of difficulty occurs in non-triangular problems of larger size the cure is by no means easy!

To study the growth of the matrix exponential the **logarithmic norm** (see Ström [559])

$$\mu(A) = \lim_{\epsilon \downarrow 0} \frac{\|I + \epsilon A\| - 1}{\epsilon} \quad (10.8.19)$$

can be used. Note that $\mu(A)$ can be a negative number. In other respects the properties re similar to those of a norm, for example

$$\mu(\gamma A) = |\gamma| \mu(A) \quad \text{if } \gamma > 0, \quad (10.8.20)$$

$$|\mu(A)| \leq \|A\|, \quad (10.8.21)$$

$$\mu(A + B) \leq \mu(A) + \mu(B) \leq \mu(A) + \|B\|. \quad (10.8.22)$$

For the 2-norm $\mu(A)$ equals the largest eigenvalue of $(A + A^H)/2$

$$\mu_2(A) = \max \frac{1}{2} \lambda(A + A^H), \quad (10.8.23)$$

which can be shown to equal the numerical abscissa $\alpha(A)_2 = \max_{z \in F(A)} \Re z$.

The behavior of $\|e^{tA}\|_2$ may be very different in the initial, transient, and asymptotic phase. The asymptotic behavior depends on $\mu(A)$. In particular, $\lim_{t \rightarrow \infty} \|e^{tA}\| = 0$ if and only if $\mu(A) < 0$. The logarithmic norm $\mu(A)$ (or spectral abscissa) determines the growth of e^{tA} for small positive t .

Example 10.8.3. Consider the matrix

$$A = \begin{pmatrix} -1 & 4 \\ 0 & -2 \end{pmatrix}.$$

Since $\max\{-1, -2\} = -1 < 0$ it follows that $\lim_{t \rightarrow \infty} e^{tA} = 0$. In Figure 10.8.1 $\|e^{tA}\|_2$ is plotted as a function of t . The curve has a **hump** illustrating that as t increases some of the elements in e^{tA} first increase before they start to decay. The logarithmic norm is the largest eigenvalue of the symmetric matrix

$$B = \frac{1}{2}(A + A^T) = \begin{pmatrix} -1 & 2 \\ 2 & -2 \end{pmatrix},$$

which equals $\mu(A) = (3 + \sqrt{17})/2 \approx 3.562$, which correctly predicts the initial growth of $\|e^{tA}\|_2$.

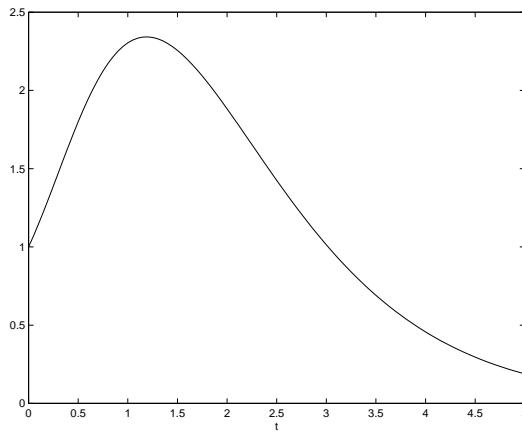


Figure 10.8.1. $\|e^{tA}\|$ as a function of t for the matrix in Example 10.8.3.

One of the best methods to compute e^A , the method of scaling and squaring, uses the fundamental relation

$$e^A = (e^{A/m})^m, \quad m = 2^s$$

of the exponential function. Here the exponent s is chosen so that $e^{A/m}$ can be reliably computed, e.g. from a Taylor or Padé approximation. Then $e^A = (e^{A/m})^{2^s}$ can be formed by squaring the result s times.

Instead of the Taylor series it is advantageous to use the diagonal Padé approximation of e^x ; see Volume I, page 349.

$$r_{m,m}(z) = \frac{P_{m,m}(z)}{Q_{m,m}(z)} = \frac{\sum_{j=0}^m p_j z^j}{\sum_{j=0}^n q_j z^j}, \quad (10.8.24)$$

which are known explicitly for all m . We have

$$p_j = \frac{(2m-j)! m!}{(2m)!(m-j)!j!}, \quad q_j = (-1)^j p_j, \quad j = 0 : m. \quad (10.8.25)$$

with the error

$$e^z - \frac{P_{m,m}(z)}{Q_{m,m}(z)} = (-1)^k \frac{(m!)^2}{(2m)!(2m+1)!} z^{2m+1} + O(z^{2m+2}). \quad (10.8.26)$$

Note that $P_{m,m}(z) = Q_{m,m}(-z)$, which reflects the property that $e^{-z} = 1/e^z$. The coefficients satisfy the recursion

$$p_0 = 1, \quad p_{j+1} = \frac{m-j}{(2m-j)(j+1)} p_j, \quad j = 0 : m-1. \quad (10.8.27)$$

To evaluate a diagonal Padé approximant of even degree m we can 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 forming A^2, A^4, \dots, A^{2m} . Then $Q_{2m}(A) = U - V$ needs no extra matrix multiplications. For an approximation of odd degree $2m+1$ we write

$$\begin{aligned} P_{2m+1,2m+1}(A) &= A(p_{2m+1} A^{2m} + \cdots + p_3 A^2 + p_1 I) \\ &\quad + p_{2m} A^{2m-2} + \cdots + p_2 A^2 + p_0 I = U + V. \end{aligned}$$

This can be evaluated with the same number of matrix multiplications and $Q_{2m+1}(A) = -U + V$. For example, the function `expm` in MATLAB uses a scaling such that $2^{-s} \|A\| < 1/2$ and a diagonal Padé approximant of degree $2m = 6$

$$P_{6,6}(z) = 1 + \frac{1}{2}z + \frac{5}{44}z^2 + \frac{1}{66}z^3 + \frac{1}{792}z^4 + \frac{1}{15840}z^5 + \frac{1}{665280}z^6.$$

The final division $P_{k,m}(A)/Q_{m,m}(A)$ is performed by solving

$$Q_{m,m}(A)r_{m,m}(A) = P_{m,m}(A)$$

for $r_{m,m}(A)$ using Gaussian elimination.

It can be shown ([441, Appendix A]) that then $r_{mm}(2^{-s}A)^{2^s} = e^{A+E}$, where

$$\frac{\|E\|}{\|A\|} < 2^3(2^{-s}\|A\|)^{2m} \frac{(m!)^2}{(2m)!(2m+1)!}.$$

For s and m chosen as in MATLAB this gives $\|E\|/\|A\| < 3.4 \cdot 10^{-16}$, which is close to the unit roundoff in IEEE double precision $2^{-53} = 1.11 \cdot 10^{-16}$. Note that this backward error result does not guarantee an accurate result. If the problem is inherently sensitive to perturbations the error can be large.

The analysis does not take roundoff errors in the squaring phase into consideration. This is the weak point of this approach. We have

$$\|A^2 - fl(A^2)\| \leq \gamma_n \|A\|^2, \quad \gamma_n = \frac{nu}{1-nu}$$

but since possibly $\|A^2\| \ll \|A\|^2$ this is not satisfactory and shows the danger in matrix squaring. If a higher degree Padé approximation is chosen then the number of squarings can be reduced. Choices suggested in the literature (N. J. Higham [331]) are $m = 8$, with $2^{-s}\|A\| < 1.5$ and $m = 13$, with $2^{-s}\|A\| < 5.4$.

The logarithm of a matrix A is a solution to the matrix equation $e^X = A$. The following example shows that there may be non-primary solutions (see Gantmacher).

$$e^B = I, \quad B = \begin{pmatrix} 0 & 2\pi & 0 \\ -2\pi & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Let $A \in \mathbf{C}^{n \times n}$ have no eigenvalues on \Re^- . Then there exists a unique principal logarithm $X = \log(A)$ such that

$$-\pi < \Im \lambda(X) < \pi.$$

For any integer $p > 0$, $X^{1/p}$ is the unique X such that $X^p = A$ and

$$-\pi/p < \arg(\lambda(X)) < \pi/p.$$

An efficient method to compute the $\log(A)$ is the method of **inverse scaling and squaring** due to Kenney and Laub [374]. In this the matrix

$$I + X = A^{1/2^k}$$

is computed by repeatedly taking square roots of A until X is sufficiently small. Then a diagonal Padé approximant is used to compute $\log(I + X)$. The result is finally obtained from the identity

$$\log A = 2^k \log A^{1/2^k} = 2^k \log(I + X). \quad (10.8.28)$$

The method can be applied to the original matrix A , but this requires square roots of full matrices. More efficiently, the method is applied to the triangular Schur factor of A or diagonal blocks within the Schur–Parlett method.

Padé approximants of $\log(1+x)$ can be obtained from the continued fraction expansion

$$\ln(1+x) = \frac{x}{1+} \frac{x}{2+} \frac{x}{3+} \frac{2^2x}{4+} \frac{2^2x}{5+} \dots \quad (10.8.29)$$

The first few approximants are

$$r_{11} = x \frac{2}{2+x}, \quad r_{22} = x \frac{6+3x}{6+6x+x^2}, \quad r_{33} = x \frac{60+60x+11x^2}{60+90x+36x^2+3x^3}.$$

These Padé approximants can be evaluated, e.g., by Horner's scheme. Several other potentially more efficient methods are investigated by Higham [327].

These approximations contain both odd and even terms and unlike the exponential function there is no symmetry between the nominator and denominator that can be used to reduce the work. Using the identity

$$\ln(1+x) = \ln\left(\frac{1+z}{1-z}\right), \quad z = \frac{x/2}{1+x/2}, \quad (10.8.30)$$

we can instead use the continued fraction expansion in z ,

$$\begin{aligned} \frac{1}{2z} \ln\left(\frac{1+z}{1-z}\right) &= \frac{1}{1-} \frac{z^2}{3-} \frac{2^2z^2}{5-} \frac{3^2z^2}{7-} \dots \frac{n^2z^2}{2n+1} \dots \\ &= 1 + \frac{z^2}{3} + \frac{z^4}{5} + \dots \end{aligned}$$

This contains only even terms. The convergents of this expansion gives the Padé approximants

$$\begin{aligned} s_{01} &= \frac{3}{3-z^2}, & s_{11} &= \frac{15+4z^2}{3(5-3z^2)}, & s_{12} &= \frac{5(21-11z^2)}{3(35-30z^2+3z^4)}, \\ s_{22} &= \frac{945-735z^2+64z^4}{15(63-70z^2+15z^4)}. \end{aligned}$$

Here the diagonal approximants s_{mm} are most interest. For example, the approximation s_{22} matches the Taylor series up to the term z^8 and the error is approximately equal to the term $z^{10}/11$. Note that the denominators are the Legendre polynomials in $1/z$; see Volume I, Example 3.5.6

To evaluate $\ln(I+X)$, we first compute

$$Z = \frac{1}{2}(I + \frac{1}{2}X)^{-1}X, \quad Z^2 = Z * Z,$$

using an LU factorization of $(I + \frac{1}{2}X)$. The nominator and denominator polynomials in the Padé approximants are then evaluated, by Horner's scheme, e.g., for s_{22}

$$P(Z^2) = 63I - 49Z^2 + (64/15)Z^4, \quad Q(Z^2) = 63I - 70Z^2 + 15Z^4.$$

Finally, the quotient $Q(Z^2)^{-1}P(Z^2)$ is computed by performing an LU factorization of $Q(Z^2)$.

10.8.4 Matrix Square Root and the Polar Decomposition

Any matrix X such that $X^2 = A$, where $A \in \mathbf{C}^{n \times n}$ is called a **square root** of A and denoted by $X = A^{1/2}$. Unlike a square root of a scalar, the square root of a matrix may not exist. For example, it is easy to verify that the matrix

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

can have no square root; see Problem 10.8.8.

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.

There is a **principal square root** of particular interest, namely the one whose eigenvalues lie in the right half plane. The principal square root, when it exists, is a polynomial in the original matrix.

Lemma 10.8.5.

Assume that $A \in \mathbf{C}^{n \times n}$ has no eigenvalues on the closed negative axis \Re^- . Then there exists a unique square root $X = A^{1/2}$, such that all the eigenvalues of X lie in the open right half plane and X is a primary matrix function of A . This matrix X is called a principal square root of A .

We assume in the following that the condition in Lemma 10.8.5 is satisfied. When A is symmetric positive definite the principal square root is the unique symmetric positive definite square root.

If X_k is an approximation to $X = A^{1/2}$ and we put $X = X_k + E_k$, then

$$A = (X_k + E_k)^2 = X_k^2 + X_k E_k + E_k X_k + E_k^2.$$

Ignoring the term E_k^2 gives

$$X_{k+1} = X_k + E_k, \quad X_k E_k + E_k X_k = A - X_k^2. \quad (10.8.31)$$

In general this iteration is expensive, since at each step a Sylvester equation has to be solved for the correction E_k . But if the initial approximation X_0 is a polynomial in A , then all following X_k will also commute with A . Then the iteration (10.8.31) simplifies to

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}A). \quad (10.8.32)$$

This Newton iteration is quadratically convergent, but numerically unstable. Because of rounding errors the computed approximations will not commute with A and this simple iteration will not work unless A is extremely well conditioned.

Several stable variants of the Newton iteration have been suggested. For the Denman–Beavers [153] iteration

$$X_{k+1} = \frac{1}{2}(X_k + Y_k^{-1}), \quad Y_{k+1} = \frac{1}{2}(Y_k + X_k^{-1}), \quad (10.8.33)$$

with initial conditions $X_0 = A$, $Y_0 = I$, it holds that

$$\lim_{k \rightarrow \infty} X_k = A^{1/2}, \quad \lim_{k \rightarrow \infty} Y_k = A^{-1/2}.$$

This iteration needs two matrix inversions per iteration. It is mathematically equivalent to (10.8.31) and hence also quadratically convergent.

We now describe a method to compute the principal square root based on the Schur decomposition $A = QSQ^H$. If U is an upper triangular square root of S , then $X = QUQ^H$ is a square root of A . If A is a normal matrix then $S = \text{diag}(\lambda_i)$ and we can just take $U = \text{diag}(\lambda_i^{1/2})$. Otherwise, from the relation $S = U^2$, we get

$$s_{ij} = \sum_{k=i}^j u_{ik} u_{kj}, \quad i \leq j. \quad (10.8.34)$$

This gives a recurrence relation for determining the elements in U . For the diagonal elements in U we have

$$u_{ii} = s_{ii}^{1/2}, \quad i = 1 : n, \quad (10.8.35)$$

and further

$$u_{ij} = \left(s_{ij} - \sum_{k=i+1}^{j-1} u_{ik} u_{kj} \right) / (u_{ii} + u_{jj}). \quad i < j. \quad (10.8.36)$$

Hence, the elements in U can be determined computed from (10.8.36), for example, one diagonal at a time. Since whenever $s_{ii} = s_{jj}$ we take $u_{ii} = u_{jj}$ this recursion does not break down. (Recall we assumed that at most one diagonal element of S is zero.)

If we let \bar{U} be the computed square root of S then it can be shown that

$$\bar{U}^2 = S + E, \quad \|E\| \leq c(n)u(\|S\| + \|U\|^2),$$

where u is the unit roundoff and $c(n)$ a small constant depending on n . If we define

$$\alpha = \|A^{1/2}\|^2/\|A\|,$$

then we have

$$\|E\| \leq c(n)u(1 + \alpha)\|S\|.$$

We remark that for real matrices an analogue algorithm can be developed, which uses the real Schur decomposition and only employs real arithmetic.

An iterative method for computing the inverse, the **Schulz iteration** [524]

$$X_{k+1} = X_k(2I - AX_k) = (2I - AX_k)X_k. \quad (10.8.37)$$

This is an analogue to the iteration $x_{k+1} = x_k(2 - ax_k)$, for computing the inverse of a scalar. It can be shown that if $X_0 = \alpha_0 A^T$ and $0 < \alpha_0 < 2/\|A\|_2^2$, then $\lim_{k \rightarrow \infty} X_k = A^{-1}$. Convergence can be slow initially, but is ultimately quadratic,

$$E_{k+1} = E_k^2, \quad E_k = I - AX_k \quad \text{or} \quad I - X_k A.$$

Since about $2\log_2 \kappa_2(A)$ (see [538, 1974]) iterations are needed for convergence it cannot in general compete with direct methods for dense matrices. However, a few steps of the iteration (10.8.37) can be used to improve an approximate inverse. Another use is as an inner iteration in the iteration for the matrix square root.

The p th root of a matrix, denoted by $A^{1/p}$, and its inverse $A^{-1/p}$, can be similarly defined. The principal p th root is the unique matrix which satisfies $X^p = A$ and whose eigenvalues lie in the segment $\{z \mid -\pi/p < \arg(z) < \pi/p\}$. Newton's iteration becomes

$$X_{k+1} = \frac{1}{p}((p-1)X_k + X_k^{1-p}A). \quad (10.8.38)$$

This iteration converges (in exact arithmetic) quadratically if $X_0 = I$ and all eigenvalues of A lie in the union of the open positive real axis and the set

$$\{z \in \mathbf{C} \mid \Re z > 0 \text{ and } |z| \leq 1\}. \quad (10.8.39)$$

The Newton iteration (10.8.38) can be rewritten as (see Iannazzo [345, 346])

$$X_{k+1} = X_k \left(\frac{(p-1)I + M_k}{p} \right), \quad M_{k+1} = \left(\frac{(p-1)I + M_k}{p} \right)^{-p} M_k, \quad (10.8.40)$$

with initial values equal to $X_0 = I$ and $M_0 = A$.

The Polar Decomposition

Several matrix functions can be expressed in terms of the square root. In the polar decomposition introduced in Section 8.1.5) of a matrix $A \in \mathbf{C}^{m \times n}$ is the factors can be expressed

$$A = PH, \quad P = (A^H A)^{1/2}, \quad P = A(A^T A)^{-1/2}. \quad (10.8.41)$$

The unitary factor P in the polar decomposition can be written in the form $P = e^{iH}$, where H is a Hermitian matrix; see Gantmacher [227, p. 278]. Thus, the polar decomposition can be regarded as a generalization to matrices of the complex number representation $z = re^{i\theta}$, $r \geq 0$!

If $A \in \mathbf{R}^{n \times n}$ is a *real* matrix, then H is a symmetric positive semidefinite matrix and P orthogonal. Any real orthogonal matrix P , with $\det(P) = +1$, can be written as $P = e^K$, where K is a real skew-symmetric matrix, since

$$P^T = e^{K^T} = e^{-K} = P^{-1}.$$

The eigenvalues of a skew-symmetric matrix K lie on the imaginary axis and are mapped onto eigenvalues for Q on the unit circle by the mapping $Q = e^K$

Example 10.8.4.

Let $A \in \mathbf{R}^{3 \times 3}$ have the polar decomposition $A = QS$. Then there is a skewsymmetric matrix

$$K = \begin{pmatrix} 0 & k_{12} & k_{13} \\ -k_{12} & 0 & k_{23} \\ -k_{13} & -k_{23} & 0 \end{pmatrix}$$

such that $Q = e^K$.

We now consider iterative methods for computing the unitary factor P . The related Hermitian factor then is $H = P^H A$. When P is a computed factor then one should take

$$H = \frac{1}{2}(P^H A + (P^H A)^H)$$

to ensure that H is Hermitian. If A is sufficiently close to a unitary matrix, then a series expansion can be used to compute P . If $z = |z|e^{i\varphi}$ is a complex number, then

$$z/|z| = z(1 - (1 - |z|^2))^{-1/2} = z(1 - q)^{-1/2}, \quad q = 1 - |z|^2.$$

and expanding the right hand side in a Taylor series in q gives

$$z(1 - q)^{-1/2} = z\left(1 + \frac{1}{2}q + \frac{3}{8}q^2 + \dots\right).$$

The related matrix expansion is

$$P = A\left(I + \frac{1}{2}U + \frac{3}{8}U^2 + \dots\right), \quad U = I - A^H A, \quad (10.8.42)$$

This suggests the following iterative algorithm: Put $A_0 = A$, and for $k = 0, 1, 2, \dots$, compute

$$A_{k+1} = A_k\left(I + \frac{1}{2}U_k + \frac{3}{8}U_k^2 + \dots\right), \quad U_k = I - A_k^H A_k. \quad (10.8.43)$$

In particular, terminating the expansion (10.8.42) after the linear term we have

$$A_{k+1} = A_k\left(I + \frac{1}{2}U_k\right) = \frac{1}{2}A_k(3I - A_k^H A_k), \quad k = 0, 1, 2, \dots$$

The rate of convergence is quadratic. It can be shown (see [62]) that a sufficient condition for convergence is that for some consistent matrix norm $\|I - A^H A\| \leq 2$.

An alternative rational iterative algorithm for computing the polar decomposition is the Newton iteration. Assume that $A_0 = A$ is square and nonsingular, we compute

$$A_{k+1} = \frac{1}{2}(A_k + (A_k^H)^{-1}). \quad (10.8.44)$$

This iteration avoids the possible loss of information associated with the explicit formation of $A^H A$. It is globally convergent to P and asymptotically the convergence is quadratic. The corresponding scalar iteration is the Newton iteration for the square root of 1. The iteration (10.8.44) does not suffer from the instability noted before of the Newton square root iteration because the unit matrix commutes with any matrix.

A drawback with the iteration (10.8.44) is that it cannot directly be applied to a rectangular or singular matrix A . If it is rewritten in the form

$$A_{k+1} = \frac{1}{2}A_k(I + (A_k^H A_k)^{-1}), \quad (10.8.45)$$

we get a formula that can be applied in the rectangular and rank deficient case. Another possibility to treat a rectangular matrix is to first perform a QR factorization

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where R is nonsingular. If the polar decomposition of R is $R = PH$ then $A = (QP)H$ is the polar decomposition of A . Hence, we can apply the iteration (10.8.44) to R .

To analyze the iteration (10.8.44) we let the singular value decomposition of $A_0 = A$ be

$$A_0 = U\Sigma_0 V^H, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n).$$

Then

$$A_k = UD_k V^H, \quad D_k = \text{diag}(d_1^{(k)}, \dots, d_n^{(k)}),$$

where $D_0 = \Sigma$, and by (10.8.44)

$$D_{k+1} = \frac{1}{2}(D_k + (D_k)^{-1}), \quad k \geq 0.$$

Thus, (10.8.44) is equivalent to n decoupled scalar iterations

$$d_i^{(0)} = \sigma_i, \quad d_i^{(k+1)} = \frac{1}{2}(d_i^{(k+1)} + 1/d_i^{(k+1)}), \quad i = 1 : n. \quad (10.8.46)$$

From familiar relations for the Newton square root iteration we know that

$$\frac{d_i^{(k+1)} - 1}{d_i^{(k+1)} + 1} = \left(\frac{d_i^{(k)} - 1}{d_i^{(k)} + 1} \right)^2 = \dots = \left(\frac{\sigma_i - 1}{\sigma_i + 1} \right)^{2^{k+1}}. \quad i = 1 : n. \quad (10.8.47)$$

Note that the convergence depends on the spread of the singular values of A but is independent of n .

Initially the convergence of the iteration can be slow. It follows by inspecting (10.8.46) that singular values $\sigma_i \gg 1$ will initially be reduced by a factor of two in each step. Similarly singular values $\sigma_i \ll 1$ will in the first iteration be transformed into a large singular value and then be reduced by a factor of two. Convergence can be accelerated by using 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 is $A_0 = A$,

$$A_{k+1} = \frac{1}{2}(\gamma_k A_k + (\gamma_k A_k^H)^{-1}), \quad k = 0, 1, 2, \dots, \quad (10.8.48)$$

where γ_k are scale factors. The optimal scale factors are determined by the condition that $\gamma_k \sigma_1(A_k) = 1/(\gamma_k \sigma_n(A_k))$ or

$$\gamma_k = (\sigma_1(A_k) \sigma_n(A_k))^{-1/2}.$$

Since the singular values are not known we must use some cheaply computable approximation to these optimal scale factors. We have the estimate

$$\sigma_1(A) = \|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty} \leq \sqrt{n} \|A\|_2.$$

Defining

$$\alpha_k = \sqrt{\|A_k\|_1 \|A_k\|_\infty}, \quad \beta_k = \sqrt{\|A_k^{-1}\|_1 \|A_k^{-1}\|_\infty}$$

we use the scale factors

$$\gamma_k = (\alpha_k / \beta_k)^{-1/2}.$$

(see Higham [323] and Kenney and Laub [376])

10.8.5 The Matrix Sign Function

Assume that the matrix $A \in \mathbf{C}^{n \times n}$ has no eigenvalues on the imaginary axis. Let

$$A = X^{-1} J X, \quad J = \begin{pmatrix} J_+ & 0 \\ 0 & J_- \end{pmatrix}$$

be the Jordan canonical form arranged so that the n_1 eigenvalues of J_+ lie in the open right half-plane and the n_2 eigenvalues of J_- lie in the open left half-plane ($n_1 + n_2 = n$). Then the matrix **sign function** is defined as

$$\text{sign}(A) = X \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} X^{-1} \quad (10.8.49)$$

The sign function is related to the matrix square root by the identity

$$\text{sign}(A) = A(A^2)^{-1/2}.$$

The matrix sign function can be used for computing invariant subspaces for nonsymmetric matrices and their associated eigenvalues. and has many applications in control theory. It can be verified that the spectral projectors corresponding to the eigenvalues in the right and left plane are

$$P_{\pm} = \frac{1}{2}(I \pm \text{sign}(A)),$$

respectively. If the leading columns of an orthogonal matrix Q span the column space of P_+ then

$$Q^T A Q = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

which is a spectral decomposition of A , where $\lambda(A_{11})$ contains the eigenvalues of A in the right half plane.

Instead of computing $\text{sign}(A)$ from its definition (10.8.49) it is usually more efficient to use an iterative algorithm rich in matrix multiplications. We first observe that the scalar iteration

$$\mu_{k+1} = \frac{1}{2}(\mu_k + \mu_k^{-1})$$

can be interpreted as a Newton iteration for the square root of 1. The iteration converges with quadratic rate to 1 if $\Re(\mu_0) > 0$ and to -1 if $\Re(\mu_0) < 0$. If $S =$

$\text{sign}(A)$ is defined, then $S^2 = I$, i.e., S is **involutory**. and $S^{-1} = S$. This suggests using the Newton iteration for the square root of I

$$A_{k+1} = \frac{1}{2} (A_k + A_k^{-1}), \quad A_0 = A. \quad (10.8.50)$$

to compute the matrix sign function. This iteration is globally and quadratically convergent to $\text{sign}(A)$, provided A has no eigenvalues on the imaginary axis. By computing the sign function of a Möbius transformation of A the spectrum can be split along arbitrary lines or circles rather than the imaginary axis.

The iteration (10.8.50) can be analyzed using the eigendecomposition of A . The convergence is determined by the convergence of the eigenvalues to ± 1 . Ill-conditioning of a matrix A_k can destroy the convergence or cause mis-convergence. To analyze the conditioning of the matrix sign function we assume 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

$$\begin{aligned} \text{sign}(A) &= X\text{sign}(\text{diag}(A_{11}, A_{22}))X^{-1} \\ &= X\text{diag}(I, -I)X^{-1} = \begin{pmatrix} I & -2P \\ 0 & -I \end{pmatrix} \end{aligned}$$

The spectral projector corresponding to the eigenvalues of A_{11} is

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

Thus $\|R\|_2 = (1 + \|P\|_2^2)^{1/2}$, and for the solution of the Sylvester equation we have

$$\|P\|_2 \leq \frac{\|A_{12}\|_2}{\text{sep}(A_{11}, A_{22})}.$$

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 [23]).

10.8.6 Estimating Matrix Functionals

Let f be smooth function on a given real interval $[a, b]$ and consider the **matrix functional**

$$F(A) = u^T f(A)u, \quad (10.8.51)$$

where u is a given vector and $A \in \mathbf{R}^{n \times n}$ is a symmetric matrix. The evaluation of such a functional arises in many applications.

Example 10.8.5.

Let \bar{x} be an approximate solution of a linear system $Ax = b$. If $r = b - A\bar{x}$ is the residual vector then the error $e = x - \bar{x}$ can be expressed as $e = A^{-1}r$. Thus the 2-norm of the error equals

$$\|e\|_2 = e^T e = r^T A^{-2} r = r^T f(A) r, \quad f(x) = x^{-2}. \quad (10.8.52)$$

Therefore the problem of computing an upper bound for $\|e\|_2$ given the residual vector is a special case of estimating a matrix functional.

Since A is a real symmetric matrix it has a spectral decomposition

$$A = Q\Lambda Q^T,$$

where Q is an orthonormal matrix whose columns are the normalized eigenvalues of A and Λ is a diagonal matrix containing the eigenvalues

$$a = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n = b.$$

Then by definition of a matrix function we have

$$F(A) = u^T Q f(\Lambda) Q^T u = \sum_{i=1}^n f(\lambda_i) \mu_j^2, \quad Q^T u = (\mu_1, \mu_2, \dots, \mu_n)^T \quad (10.8.53)$$

In the following we assume that $\|u\|_2 = \|Q^T u\|_2 = 1$.

The last sum can be considered as a Riemann–Stieltjes integral

$$F(A) = \int_a^b f(t) d\mu(t),$$

where the measure μ is piecewise constant and defined by

$$\mu(t) = \begin{cases} 0, & \text{if } t < a; \\ \sum_{j=1}^i \mu_j^2, & \text{if } \lambda_i \leq t < \lambda_{i+1}; \\ \sum_{j=1}^n \mu_j^2, & \text{if } \lambda_n \leq t. \end{cases}$$

10.8.7 Finite Markov Chains

A finite **Markov chain**⁵⁹ 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 development 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 equal to

$$p_{ij} = \Pr\{X_t = s_j \mid X_{t-1} = s_i\}.$$

The matrix $P \in \mathbf{R}^{n \times n}$ of **transition probabilities** is nonnegative and must satisfy

$$\sum_{j=1}^n p_{ij} = 1, \quad i = 1 : n. \quad (10.8.54)$$

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 (10.8.54) as

$$Pe = e, \quad e = (1, 1, \dots, 1)^T. \quad (10.8.55)$$

Thus, e is a *right* eigenvector of P corresponding to the eigenvalue $\lambda = 1$ and

$$P^k e = P^{k-1}(Pe) = P^{k-1}e = \dots = e, \quad k > 1.$$

That is the matrix 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

$$p^T P = p^T, \quad p^T e = 1. \quad (10.8.56)$$

Hence, p is a *left* eigenvector of the transition matrix P corresponding to the eigenvalue $\lambda = 1 = \rho(P)$. Then p solves the singular homogeneous linear system

$$A^T p = 0, \quad \text{subject to } e^T p = 1, \quad A = I - P, \quad (10.8.57)$$

and p lies in the nullspace 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

⁵⁹Named after the Russian mathematician Andrej Andreevich Markov (1856–1922), who introduced them in 1908.

a simple eigenvalue of P and $\text{rank}(A) = n - 1$. Further, there is a unique positive eigenvector p with $\|p\|_1 = 1$, satisfying (10.8.56). 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 10.8.6.

Assume that a Markov chain has a positive transition matrix. Then, independent of the initial state vector, $\lim_{t \rightarrow \infty} p(t) = p$, where p spans the nullspace 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 10.8.6.

Consider a Markov chain with two states for which state 2 is always transformed into state 1 and state 1 into state 1. The corresponding transition matrix

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

with two eigenvalues of modulus $\rho(P)$: $\lambda_1 = 1$ and $\lambda_2 = -1$. Here P is symmetric and its left eigenvalue equals $p = (0.5, 0.5)^T$. However, 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 equal to the permutation matrix corresponding to a cyclic shift. Then P has m eigenvalues on the unit circle in the complex plane.

Many results in the theory of Markov chains can be phrased in terms of the so-called **group inverse** of the matrix $A = (I - P)$.

Definition 10.8.7.

The Drazin inverse of A is the unique matrix X satisfying the three equations

$$A^k X A = A, \quad X A X = X, \quad A X = X A. \quad (10.8.58)$$

It exists only if $k \geq \text{index}(A)$, where

$$\text{index}(A) = \min\{k \mid \text{rank}(A^{k+1}) = \text{rank}(A^k)\}. \quad (10.8.59)$$

The group inverse A^\ddagger of A is the unique matrix satisfying (10.8.58) for $k = 1$.

The two first identities in (10.8.58) say that Drazin inverse X is an $(1, 2)$ -inverse. The last identity says that X commutes with A . The group inverse of A exists if and only if the matrix A has index one, i.e. $\text{rank}(A^2) = \text{rank}(A)$.⁶⁰

⁶⁰It is known that if this condition holds then A belongs to a set that forms a multiplicative group under ordinary matrix multiplication.

This condition is satisfied for every transition matrix (Meyer [436, Theorem 2.1]). Further, we have

$$I - ep^T = AA^\ddagger.$$

(Note that AA^\ddagger is a projection matrix since $(ep^T)^2 = p^T e e p^T = ep^T$.)

Theorem 10.8.8 (Golub and Meyer [264]).

Let $A = I - P$, where P is the transition matrix of an ergodic Markov chain and consider the QR factorization of A . Then the R-factor is uniquely determined and has the form

$$R = \begin{pmatrix} U & -Ue \\ 0 & 0 \end{pmatrix}. \quad (10.8.60)$$

The stationary distribution p is given by the last column $q = Qe_n$ of Q and equals

$$p = q / \sum_{i=1}^n q_i. \quad (10.8.61)$$

Further, it holds that

$$A^\ddagger = (I - ep^T) \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T (I - ep^T), \quad (10.8.62)$$

where p is the stationary distribution of for P .

Proof.

That R has the form above follows from the fact that the first $n - 1$ columns of A are linearly independent and that

$$0 = Ae = \begin{pmatrix} U & u_n \\ 0 & 0 \end{pmatrix} e = \begin{pmatrix} Ue + u_n \\ 0 \end{pmatrix}$$

and thus $u_n = -Ue$. 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 it follows that $q > 0$ or $q < 0$ and (10.8.61) follows.

If we set

$$A^- = \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T,$$

then it can be verified that $AA^-A = A$. Using the definition of a group inverse

$$\begin{aligned} A^\ddagger &= A^\ddagger AA^\ddagger = A^\ddagger(AA^-A)A^\ddagger = (A^\ddagger A)A^-(AA^\ddagger) \\ &= (I - ep^T) \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T (I - ep^T), \end{aligned}$$

which proves the result. \square

For an ergodic chain define the matrix M of *mean first passage times*, where the element m_{ij} is 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

$$AX = ee^T - P \operatorname{diag}(X).$$

The mean first passage times matrix M can be expressed in terms of A^\ddagger as

$$M = I - A^\ddagger + ee^T \operatorname{diag}(A^\ddagger).$$

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 state s_i to s_j in a finite number of steps. If also s_j has access to s_i , 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

$$P = \begin{pmatrix} P_{11} & 0 & \dots & 0 \\ P_{21} & P_{22} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ P_{s1} & P_{s2} & \dots & P_{ss} \end{pmatrix} \quad (10.8.63)$$

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 the matrix P is irreducible then the corresponding matrix chain contains a single class of states.

Example 10.8.7. Suppose there is an epidemic in which every month 10% of those who are well become sick and of those who are sick 20% dies, 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!

We now describe a way to force a Markov chain to become irreducible.

Example 10.8.8 (Eldén).

Let $P \in \mathbf{R}^{n \times n}$ 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 since $e^T e = n$, 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 equal $1, \lambda_2, \lambda_3, \dots, \lambda_n$, 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 10.1.13) we define the orthogonal matrix $U = (u_1 \ U_2)$, where $u_1 = e/\sqrt{n}$. 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

$$\begin{aligned} U^T Q U &= U^T \left(\alpha P + (1 - \alpha) \frac{1}{n} ee^T \right) U \\ &= \alpha \begin{pmatrix} 1 & v^T \\ 0 & T \end{pmatrix} + (1 - \alpha) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & \alpha v^T \\ 0 & \alpha T \end{pmatrix}. \end{aligned}$$

Review Questions

- 8.1** Define the matrix function e^A . Show how this can be used to express the solution to the initial value problem $y'(t) = Ay(t)$, $y(0) = c$?
- 8.2** What can be said about the behavior of $\|A^k\|$, $k \gg 1$, in terms of the spectral radius and the order of the Jordan blocks of A ?
- 8.3** (a) Given a square matrix A . Under what condition does there exist a vector norm, such that the corresponding operator norm $\|A\|$ equals the spectral radius? If A is diagonalizable, mention a norm that has this property.
(b) What can you say about norms that come close to the spectral radius, when the above condition is not satisfied? What sets the limit to their usefulness?
- 8.4** Show that
- $$\lim_{t \rightarrow \infty} \frac{1}{t} \ln \|e^{At}\| = \max_{\lambda \in \lambda(A)} \operatorname{Re}(\lambda), \quad \lim_{t \rightarrow 0} \frac{1}{t} \ln \|e^{At}\| = \mu(A).$$
- 8.5** Under what conditions can identities which hold for analytic functions of complex variable(s) be generalized to analytic functions of matrices?
- 8.6** Show that if all off-diagonal elements of a matrix B are nonnegative, then $A = e^B > 0$.
Hint: For some $\alpha > 0$, $B = C - \alpha I$, where $C > 0$.
- 8.7** (a) Show that any permutation matrix is doubly stochastic.
(b) What are the eigenvalues of matrix

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}?$$

8.8 Suppose that P and Q are row stochastic matrices.

- (a) Show that $\alpha P + (1 - \alpha Q)$ is a row stochastic matrix.
 - (b) Show that PQ is a row stochastic matrix.
-

Problems and Computer Exercises

8.1 (a) Let $A \in \mathbf{R}^{n \times n}$, and consider the matrix polynomial

$$p(A) = a_0 A^n + a_1 A^{n-1} + \cdots + a_n I \in \mathbf{R}^{n \times n}.$$

Show that if $Ax = \lambda x$ then $p(\lambda)$ is an eigenvalue and x an associated eigenvector of $p(A)$.

(b) Show that the same is true in general for an analytic function $f(A)$. Verify (10.8.12). Also construct an example, where $p(A)$ has other eigenvectors in addition to those of A .

8.2 Show that the series expansion

$$(I - A)^{-1} = I + A + A^2 + A^3 + \dots$$

converges if $\rho(A) < 1$.

8.3 (a) Let $\|\cdot\|$ be a consistent matrix norm, and ρ denote the spectral radius. Show that

$$\lim_{k \rightarrow \infty} \|A^k\|^{1/k} = \rho(A).$$

(b) Show that

$$\lim_{t \rightarrow \infty} \frac{\ln \|e^{At}\|}{t} = \max_{\lambda \in \lambda(A)} \Re(\lambda).$$

Hint: Assume, without loss of generality, that A is in its Jordan canonical form.

8.4 Show that for the matrix norm $\|A\|_\infty$ the corresponding logarithmic norm equals

$$\mu(A) = \max_i \left(\Re(a_{ii}) + \sum_{j \neq i} |a_{ij}| \right).$$

8.5 Show that

$$e^A \otimes e^B = e^{B \oplus A},$$

where \oplus denotes the Kronecker sum.

8.6 (a) Show that if $A = \begin{pmatrix} \lambda_1 & 1 \\ 0 & \lambda_2 \end{pmatrix}$ and $\lambda_1 \neq \lambda_2$ then

$$f(A) = \begin{pmatrix} f(\lambda_1) & \frac{f(\lambda_1) - f(\lambda_2)}{\lambda_1 - \lambda_2} \\ 0 & f(\lambda_2) \end{pmatrix}.$$

Comment on the numerical use of this expression when $\lambda_2 \rightarrow \lambda_1$.

(b) For $A = \begin{pmatrix} 0.5 & 1 \\ 0 & 0.6 \end{pmatrix}$, show that $\ln(A) = \begin{pmatrix} -0.6931 & 1.8232 \\ 0 & 0.5108 \end{pmatrix}$.

8.7 (a) Compute e^A , where

$$A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix},$$

using the method of scaling and squaring. Scale the matrix so that $\|A/2^s\|_\infty < 1/2$ and approximate the exponential of the scaled matrix by a Padé approximation of order (4,4).

(b) Compute the eigendecomposition $A = X\Lambda X^{-1}$ and obtain $e^A = Xe^\Lambda X^{-1}$. Compare the result with that obtained in (a).

8.8 (a) The so-called Wilson matrix ([221, 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 a 2-norm condition number $\kappa_2(W) \approx 2984$. Perform 12 iterations with the simplified Newton method (10.8.32) with the starting value $X_0 = I$. Compute also the residual norm $\|X_k^2 - W\|_F$. For what value of k is the smallest residual norm obtained?

(b) Same as (a), but use the Denman–Beavers iteration (10.8.33).

8.9 (Higham [323]) (a) Let $A \in \mathbf{R}^{n \times n}$ 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) lead to an attractive algorithm for computing the square root of A . Suppose s steps of the iteration (10.8.44) is 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?

8.10 (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 ?

8.11 Show the relation

$$\text{sign} \begin{pmatrix} 0 & A \\ I & 0 \end{pmatrix} = \begin{pmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{pmatrix}. \quad (10.8.64)$$

8.12 Show that an analytic function of the matrix A can be computed by Newton's interpolation formula, i.e.,

$$f(A) = f(\lambda_1)I + \sum_{j=1}^{n^*} f(\lambda_1, \lambda_2, \dots, \lambda_j)(A - \lambda_1 I) \cdots (A - \lambda_j I)$$

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 .

8.13 We use the notation of Theorem 10.8.1. For a given n , show by an appropriate choice of ϵ that $\|A^n\|_p \leq C n^{m^*-1} \rho^n$, where C is independent of n . Then derive the same result from the Jordan Canonical form.

Hint: See the comment after Theorem 10.8.1.

8.14 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 C then, by (10.8.14) $\phi_C(A) = I$. What is $\phi_C(A)$, when only part of the spectrum (or none of it) is inside C ? Is it generally true that $\phi_C(A)^2 = \phi_C(A)$?

Hint: First consider the case, when A is a Jordan block.

8.15 Higher order iterations for the orthogonal factor in the polar decomposition based on Padé approximations are listed in [375]. A special case is the third order Halley's method, (see Vol. I, Section 6.3.4). which has been studied by Gander [223]. This leads to the iteration

$$A_{k+1} = A_k (3I + A_k^H A_k) (I + 3A_k^H A_k)^{-1}.$$

Implement this method and test it on ten matrices $A \in \mathbf{R}^{20 \times 10}$ with random elements uniformly distributed on $(0, 1)$. (In MATLAB such matrices are generated by the command $A = \text{rand}(20, 10)$.) How many iteration are needed to reduce $\|I - A_k^T A_k\|_F$ to the order of machine precision?

Notes and Further References

A still unsurpassed text on computational methods for the eigenvalue problem is Wilkinson [611, 1965]. A classical treatment of the symmetric eigenvalue is given by Parlett [478, 1980]. A more recent treatment of is found in Watkins [602]. The book by Stewart [551] is unsurpassed in its detailed and insightful coverage. Large scale eigenvalue problems are treated by Saad [516].

Wilkinson and Reinsch [614] contain detailed instructive discussions and programs. Further practical details on the implementation of eigenvalue algorithms can be found in documentations of the EISPACK and LAPACK software; see Smith et al. [536], B. S. Garbow et al. [229], and E. Anderson et al. [10]. An excellent source of information on algorithms for many classes of eigenvalue problems is Bai et al. [24]. Applications of spectral methods using MATLAB are considered by Trefethen [574]. Golub and van der Vorst [275] survey the developments in eigenvalue computations.

Section 10.1

A short elegant proof of the Jordan normal form has been given by Pták [493]. However, it involves concepts which are not elementary; see also comments by Boor [72].

For a discussion of the many delicate computational problems that are involved in the further reduction to Jordan normal form we refer to Golub and Wilkinson [278]. Kågström and Ruhe [360, 359] give an algorithm for the reduction.

The algebraic Riccati equation plays a fundamental role in system theory. It is the main tool for solving the linear regulator problem and to find the stationary Kalman filter. It is also used for computing the optimal controller in more modern optimal controllers. The importance of this subject has resulted in several texts devoted to the numerical solution of such matrix equations, e.g., Lancaster and Rodman [389] and Abou-Kandil et al.[2].

Section 10.2

Varga [595] studies the original results and newer extensions of Geršgorin's circle theorem. An invaluable source of results on matrix perturbation with many historical remarks is Stewart and Sun [552]. Perturbation theory of eigenvalues and eigenspaces, with an emphasize on Hermitian and normal matrices is Bhatia [51, 52].

The residual error bounds in this section only make use of the norms of certain residuals. A survey of more elaborate inclusion theorems for the Hermitian eigenvalue problem that have been developed are found in Parlett [476, Chap. 10].

Kato [371] uses resolvent techniques are used to treat many questions of matrix and operator theory. Stewart and Sun [552] gives a lucid treatise of matrix perturbation theory, with many historical comments and a very useful bibliography. 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 [408, 409] studies relative perturbations in eigenvalues and singular values.

An introduction to Grassmann and Stiefel manifolds is given by Edelman, Arias and Smith [184]. They give a framework of Newton algorithms with orthogonality constraints. Grassmann–Rayleigh quotient iteration for computing invariant

can achieve cubic convergence for symmetric problems, as shown in [3, 4]; see also Simonson [531].

Section 10.3

A fairly complete treatment of the power method is found in Wilkinson [611] and references therein. The inverse power method was proposed in 1944 by Wielandt [608]. 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.

Ostrowski has studied the convergence of Rayleigh quotient iteration for both the symmetric and unsymmetric case in a series of papers [459].

An analysis and a survey of inverse iteration for a single eigenvector is given by Ipsen [348]. A very careful Algol implementation of Rayleigh quotient iteration for symmetric matrices was given by Rutishauser [513]. The relation between simultaneous iteration and the QR algorithm and is explained in Watkins [604].

Newton-based methods are treated by Chatelin [107] and Demmel [144]. Several newer iteration methods for computing invariant subspaces rely on the quotient geometry of Grassmann manifolds. Edelman and T. Arias and S. T. Smith [184] give the framework of algorithms with orthogonality constraints. Such a Grassmann-Rayleigh quotient iterations for computing invariant subspaces is given in [3, 4].

Section 10.4

Braman, Byers and Mathias [76, 77] have developed a version of the QR algorithm that uses a large number of shifts in each QR step. For a matrix of order $n = 2000$ (say) they use of the order of $m = 100$ shifts computed from the current lower right-hand 100×100 principal submatrix. These eigenvalues are computed using a standard QR algorithm with $m = 2$. The 100 shifts are not applied all at once by chasing a large bulge. Instead they are used to perform 50 implicit QR iterations each of degree 2. These can be applied in tight formation allowing level 3 performance.

Section 10.5

A stable algorithm for computing the SVD based on an initial reduction to bidiagonal form was first sketched by Golub and Kahan in [265]. However, the QR algorithm is not treated in this paper. The adaption of the QR algorithm for computing the SVD directly from the reduced bidiagonal matrix was described by Golub [262]. The more or less final form of the QR algorithm for the SVD is given in Golub and Reinsch [273]. Chandrasekaran and Ipsen [106] gives an analysis of a class of QR algorithms for the SVD.

Section 10.6

The zero shift QR-SVD algorithm was introduced by Demmel and Kahan [150]. The implementation given here which uses two steps of the LR algorithm is slightly simpler and more economical. An extensive survey of cases when it is possible to compute singular values and singular vectors with high relative accuracy is given in [146].

The differential qd algorithm with shifts (dqds) is a new algorithm developed independently by Fernando and Parlett [200] and von Matt [597]. It is a sequential algorithm for computing eigenvalues of tridiagonal matrices or singular values of

bidiagonal matrices in $O(n^2)$ time; see also Dhillon and Parlett [162, 161]. It is faster but as accurate that the QR algorithm. Further developments are described by Parlett [477], Dhillon [159], and Parlett and Marques [479]. An implementation of the dqds algorithm (positive case) is given by Grosser and Lang [285]

In developing the differential qd algorithm Fernando and Parlett [200] make the interesting observation that the LR algorithm was initially developed by Rutishauser from the qd algorithm. From the LR algorithm came the QR algorithm and the zero shift QR–SVD algorithm. This finally inspired a square root free version from which the qd algorithm in a new implementation evolved.

The QR algorithm for the skew-symmetric eigenproblem is due to Ward and Gray [600]. Singer and Singer [532] give a qd algorithm for skew-symmetric matrices.

Section 10.7

The product eigenvalue problems is treated in Watkins [603]. Methods for the numerical treatment of quadratic eigenvalue problems are surveyed by Tisseur and Meerbergen [567]. Stewart and Sun [552, Sec. VI] treats perturbation theory for generalized eigenvalue problems. A survey of structured eigenvalue problems are given in [86]

The result that $F(A)$ is a convex set for an arbitrary matrix is nontrivial and was first published by Toeplitz [568]. Householder [343, Sec 3.3.2] gives a proof due to Hans Schneider (unpublished). The resolvent can also be defined for closed linear operators in a Banach space; see Trefethen [572, 573], Trefethen and Embree [576]. A MATLAB package called EigTool for computing pseudospectra has been developed by Wright [620]. This also provides a graphical interface to MATLAB’s built-in `eigs` routine (ARPACK). The software is available from <http://www.comlab.ox.ac.uk/pseudospectra/eigtool>.

Section 10.8

A brief history of matrix functions is given by Higham [332, Sec. 1.10]. The square root of 2×2 and 3×3 matrices were treated already in 1858 by Cayley. Soon after a general definition of $f(A)$ was introduced by Sylvester. Laguerre defined the exponential function from its power series in 1967. The definition of $f(A)$ by an interpolating polynomial was stated by Sylvester in 1883 for the case of distinct eigenvalues. The Cauchy integral representation was introduced by Frobenius 1996 and Poincaré 1999. Matrix functions are treated in Gantmacher [228, Chapter V] and Lancaster [390, 1985]. A comprehensive state of the art treatment is available in Higham [332]. Methods for computing A^α , $\log(A)$ and related matrix functions by contour integrals are analyzed in [302].

The logarithmic norm was introduced independently by Lozinskii [417] and Dahlquist [130] as a tool to study the growth of solutions to differential equations.

Chapter 11

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, 2000.

11.1 Classical Iterative Methods

11.1.1 Introduction

The methods discussed so far in this book for solving systems of linear equations $Ax = b$ have been direct methods based on matrix factorization. Disregarding rounding errors, direct methods are reliable and yield the solution in a predictable number of operations and storage use. Using sparse direct matrix factorizations systems of quite large size can be treated and this is often the methods of choice for industrial applications, e.g., structural engineering, which typically yield very ill-conditioned matrices, direct methods are still preferred.

Iterative methods start from an initial approximation, which is successively improved until a sufficiently accurate solution is obtained. An important feature of basic iterative methods is that they work directly with the original matrix A and only need extra storage for a few vectors. Since the matrix A is involved only in terms of matrix-vector products and there is usually no need even to store A . Iterative methods are particularly useful for solving *sparse linear systems*, which typically arise from discretization of equations. Since the LU factors of matrices in such applications would typically contain order of magnitudes more nonzero elements than A itself, direct methods may become far too costly (or even impossible) to use. This is true, in particular, three-dimensional multiphysics simulations, such as reservoir modeling, mechanical engineering, electric circuit simulations often involve tens of million equations and unknowns.

As iterative methods have evolved the distinction between direct and iterative methods has become less sharp. Often an iterative method is applied to a so called preconditioned version of the system, where each iteration step involves the solution of a simpler auxiliary systems by a direct method.

11.1.2 A Model Problem

The idea of solving systems of linear equations by iterative methods dates at least back to Gauss (1823). Before the age of high speed computers the iterative methods used were usually rather unsophisticated, so called, noncyclic **relaxation methods**. Each step was guided by the sizes of the components of the residual vector $r_k = b - Ax_k$ of the current approximate solution x_k . In the 1950s, when computers replaced desk calculators, an intense development of iterative methods started.

Early iterative methods were predominantly applied for solving discretized elliptic self-adjoint partial differential equations. 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 Ω , is frequently used as model problem for these methods. In Example 7.4.3 a simple finite difference approximation of Laplace equation is used, which gives rise to a system of linear equation $Ax = b$. The matrix A is symmetric positive definite with the block-tridiagonal form

$$A = \text{trid}(-1, 2, -1) = \begin{pmatrix} 2I + T & -I & & \\ -I & 2I + T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & 2I + T \end{pmatrix} \in \mathbf{R}^{n^2 \times n^2}, \quad (11.1.1)$$

where T is symmetric tridiagonal,

$$T = \text{trid}(-1, 2, -1) = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbf{R}^{n \times n}. \quad (11.1.2)$$

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 the $5n^2$ flops needed per iteration, e.g., with Jacobi's method.

The linear system arising from Poisson'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 $p \approx 5-10$ 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 $2p \cdot n^2$ flops or equivalently $2p$ flops per unknown. Using iterative methods which take advantage of the sparsity and other features does allow the efficient solution of such systems.

The disadvantage of direct methods becomes even more accentuated for three dimensional problems. For Laplace equation in the unit cube a similar analysis shows that for solving n^3 unknown we need n^7 flops and about n^5 storage. When n grows this quickly becomes infeasible. However, basic iterative methods still require only about $7n^3$ flops per iteration.

We still have not discussed the number of iterations needed to get acceptable accuracy. It turns out that this will depend on the condition number of the matrix. We now show that for the Laplace equation considered above this condition number will be about πh^{-2} , independent of the dimension of the problem.

Lemma 11.1.1.

Let $T = \text{trid}(c, a, b) \in \mathbf{R}^{(n-1) \times (n-1)}$ be a tridiagonal matrix with constant diagonals, and assume that a, b, c are real and $bc > 0$. Then the n eigenvalues of T are given by

$$\lambda_i = a + 2\sqrt{bc} \cos \frac{i\pi}{n}, \quad i = 1 : n - 1.$$

Further, the j th component of the eigenvector v_i corresponding to λ_i is

$$v_{ij} = \left(\frac{b}{c}\right)^{j/2} \sin \frac{ij\pi}{n}, \quad j = 1 : n - 1.$$

From Lemma 11.1.1 it follows that the eigenvalues of $T = \text{trid}(-1, 2, -1)$ are

$$\lambda_i = 2(1 + \cos(i\pi/n)), \quad i = 1 : n - 1,$$

and in particular

$$\lambda_{\max} = 2(1 + \cos(\pi/n)) \approx 4, \quad \lambda_{\min} = 2(1 - \cos(\pi/n)) \approx (\pi/n)^2.$$

We conclude that the spectral condition number of $T = \text{trid}(-1, 2, -1)$ is approximately equal to $\kappa(T) = 4n^2/\pi^2$.

The matrix $A = 4(I - L - U)$ in (11.1.1) can be written in terms of the Kronecker product (see Section 7.6.5)

$$A = (I \otimes T) + (T \otimes I),$$

i.e., A is the Kronecker sum of T and T . It follows that the $(n-1)^2$ eigenvalues of A are $(\lambda_i + \lambda_j)$, $i, j = 1 : n - 1$, and hence the condition number of A is the same as for T . The same conclusion can be shown to hold for a three dimensional problem.

The eigenvalues and eigenvectors of $C = A \otimes B$ can be expressed in terms of the eigenvalues and eigenvectors of A and B . Assume that $Ax_i = \lambda_i x_i$, $i = 1 : n$, and $By_j = \mu_j y_j$, $j = 1 : m$. Then, using Lemma 7.6.4 we obtain

$$(A \otimes B)(x_i \otimes y_j) = (Ax_i) \otimes (By_j) = \lambda_i \mu_j (x_i \otimes y_j). \quad (11.1.3)$$

This shows that the nm eigenvalues of $A \otimes B$ are $\lambda_i \mu_j$, $i = 1 : n$, $j = 1 : m$, and $x_i \otimes y_j$ are the corresponding eigenvectors. If A and B are diagonalizable, $A = X^{-1} \Lambda_1 X$, $B = Y^{-1} \Lambda_2 Y$, then

$$(A \otimes B) = (X^{-1} \otimes Y^{-1})(\Lambda_1 \otimes \Lambda_2)(X \otimes Y),$$

and thus $A \otimes B$ is also diagonalizable.

The matrix

$$(I_m \otimes A) + (B \otimes I_n) \in \mathbf{R}^{nm \times nm} \quad (11.1.4)$$

is the **Kronecker sum** of A and B . Since

$$\begin{aligned} [(I_m \otimes A) + (B \otimes I_n)](y_j \otimes x_i) &= y_j \otimes (Ax_i) + (By_j) \otimes x_i \\ &= (\lambda_i + \mu_j)(y_j \otimes x_i). \end{aligned} \quad (11.1.5)$$

the nm eigenvalues of the Kronecker sum equal the sum of all pairs of eigenvalues of A and B

11.1.3 Stationary Iterative Methods

We start by describing two classical iterative methods. Assume that A has nonzero diagonal entries, i.e., $a_{ii} \neq 0$, $i = 1 : n$. If A is symmetric positive definite this is necessarily the case. Otherwise, since A is nonsingular, the equations can always be reordered so that this is true. In component form the system can then be written

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j \right), \quad i = 1 : n. \quad (11.1.6)$$

In a (minor) step of **Jacobi's method** we pick one equation, say the i th, and adjust the i th component of $x^{(k)}$ so that this equation becomes exactly satisfied. Hence, given $x^{(k)}$ we compute

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^n a_{ij} x_j^{(k)}. \quad (11.1.7)$$

In the days of “hand” computation one picked an equation with a large residual $|r_i|$ and went through the equations in an irregular manner. This is less efficient when using a computer, and here one usually perform these adjustments for $i = 1 : n$, in a cyclic fashion. Jacobi's method is therefore also called the method of *simultaneous displacements*. Note that all components of x can be updated *simultaneously* and the result does not depend on the sequencing of the equations.

The method of successive displacements or **Gauss–Seidel's method**⁶¹ differs from the Jacobi method only by using new values $x_j^{(k+1)}$ as soon as they are available

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^n a_{ij} x_j^{(k)}, \quad i = 1, 2, \dots, n. \quad (11.1.8)$$

Here the components are *successively* updated and the sequencing of the equations will influence the result.

⁶¹It was noted by Forsythe that Gauss nowhere mentioned this method and Seidel never advocated using it!

Since each new value $x_i^{(k+1)}$ can immediately replace $x_i^{(k)}$ in storage the Gauss–Seidel method storage for unknowns is halved compared to Jacobi’s method. For both methods the amount of work required in each iteration step is comparable in complexity to the multiplication of A with a vector, i.e., proportional to the number of nonzero elements in A . By construction it follows that if $\lim_{k \rightarrow \infty} x^{(k)} = x$, then x satisfies (11.1.6) and therefore the system $Ax = b$.

In **Richardson’s method**,⁶² the next approximation is computed from

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots, \quad (11.1.9)$$

where $\omega > 0$ is a parameters to be chosen. It follows easily from (11.1.9) that the residual $r^{(k)} = b - Ax^{(k)}$ and error satisfy the recursions

$$r^{(k+1)} = (I - \omega_k A)r^{(k)}, \quad x^{(k+1)} - x = (I - \omega_k A)(x^{(k)} - x).$$

In the special case that $\omega = \omega_k$ for all k , Richardson’s method is a **stationary** iterative method and

$$x^{(k)} - x = (I - \omega A)^k(x^{(0)} - x).$$

If A has a nonzero diagonal it can be scaled to have all diagonal elements equal to 1. Then Richardson’s method with $\omega = 1$ is equivalent to **Jacobi’s method**.

The Jacobi, Gauss–Seidel, and the stationary Richardson methods are all special cases of a class of iterative methods, the general form of which is

$$Mx^{(k+1)} = Nx^{(k)} + b, \quad k = 0, 1, \dots. \quad (11.1.10)$$

Here

$$A = M - N \quad (11.1.11)$$

is a **splitting** of the matrix coefficient matrix A with M nonsingular.

If the iteration (11.1.10) converges, i.e., $\lim_{k \rightarrow \infty} x^{(k)} = x$, then $Mx = Nx + b$ and it follows from (11.1.11) that 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 matrix M . This is the case, for example, if M is chosen to be triangular.

Definition 11.1.2.

An iterative method of the form (11.1.10), or equivalently

$$x^{(k+1)} = Bx^{(k)} + c, \quad k = 0, 1, \dots, \quad (11.1.12)$$

is called a (one-step) **stationary iterative method**, and

$$B = M^{-1}N = I - M^{-1}A, \quad c = M^{-1}b. \quad (11.1.13)$$

is called the **iteration matrix**.

Subtracting the equation $x = Bx + c$ from (11.1.10), we obtain the recurrence formula

$$x^{(k+1)} - x = B(x^{(k)} - x) = \dots = B^{k+1}(x^{(0)} - x), \quad (11.1.14)$$

⁶²Lewis Fry Richardson (1881–1953) English mathematician, who was the first to use mathematical methods for weather prediction.

for the error in successive approximations.

Richardson's method (11.1.9) can, for fixed $\omega_k = \omega$, be written in the form (11.1.10) with the splitting $A = M - N$, with

$$M = (1/\omega)I, \quad N = (1/\omega)I - A.$$

To write the Jacobi and Gauss–Seidel methods in the form of one-step stationary iterative methods we introduce the **standard splitting**

$$A = D_A - L_A - U_A, \quad (11.1.15)$$

where $D_A = \text{diag}(a_{11}, \dots, a_{nn})$,

$$L_A = -\begin{pmatrix} 0 & & & & \\ a_{21} & 0 & & & \\ \vdots & \ddots & \ddots & & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \quad U_A = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \ddots & \ddots & & \vdots \\ 0 & a_{n-1,n} & & 0 \end{pmatrix}, \quad (11.1.16)$$

and L_A and U_A are strictly lower and upper triangular, respectively. Assuming that $D_A > 0$, we can also write

$$D_A^{-1}A = I - L - U, \quad L = D_A^{-1}L_A, \quad U = D_A^{-1}U_A. \quad (11.1.17)$$

With these notations the Jacobi method, (11.1.7), can be written $D_Ax^{(k+1)} = (L_A + U_A)x^{(k)} + b$ or

$$x^{(k+1)} = (L + U)x^{(k)} + c, \quad c = D_A^{-1}b. \quad (11.1.18)$$

The Gauss–Seidel method, (11.1.8), becomes $(D_A - L_A)x^{(k+1)} = U_Ax^{(k)} + b$, or equivalently

$$x^{(k+1)} = (I - L)^{-1}Ux^{(k)} + c, \quad c = (I - L)^{-1}D_A^{-1}b$$

Hence these methods are special cases of one-step stationary iterative methods, and correspond to the matrix splittings

$$\begin{aligned} \text{Jacobi:} \quad M &= D_A, & N &= L_A + U_A, \\ \text{Gauss–Seidel:} \quad M &= D_A - L_A, & N &= U_A, \end{aligned}$$

The iteration matrices for the Jacobi and Gauss–Seidel methods are

$$\begin{aligned} B_J &= D_A^{-1}(L_A + U_A) = L + U, \\ B_{GS} &= (D_A - L_A)^{-1}U_A = (I - L)^{-1}U. \end{aligned}$$

The basic iterative methods described so far can be generalized for block matrices A . Assume that A and b are partitioned conformally,

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

where the diagonal blocks A_{ii} are square and nonsingular. Using this partitioning we consider the splitting

$$A = D_A - L_A - U_A, \quad D_A = \text{diag}(A_{11}, A_{22}, \dots, A_{nn}),$$

where L_A and U_A are strictly lower and upper triangular. The block Jacobi method can then be written

$$D_A x^{(k+1)} = (L_A + U_A)x^{(k)} + b,$$

or, with x is partitioned conformally,

$$A_{ii} \left(x_i^{(k+1)} - x_i^{(k)} \right) = b_i - \sum_{j=1}^n A_{ij} x_j^{(k)}, \quad i = 1 : n.$$

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 11.1.1.

For the model problem in Section 11.1.3 the matrix A can naturally be written in the block form where the diagonal blocks $A_{ii} = 2I + T$ are tridiagonal and nonsingular, see (11.1.6). The resulting systems can be solved with little overhead. Note that here 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 also known as “line” methods and the other methods as “point” methods.

11.1.4 Convergence of Stationary Iterative Methods

The stationary iterative method (11.1.12) is called **convergent** if the sequence $\{x^{(k)}\}_{k=1,2,\dots}$ converges for *all initial vectors* $x^{(0)}$. It can be seen from (11.1.14) that of fundamental importance in the study of convergence of stationary iterative methods is conditions for a sequence of powers of a matrix B to converge to the null matrix.

Recall that the spectral radius of a matrix A is the nonnegative number

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i(A)|.$$

Obtaining the spectral radius of B is usually no less difficult than solving the linear system. The following important results holds:

Theorem 11.1.3. *A matrix $B \in \mathbf{R}^{n \times n}$ is said to be **convergent** if $\rho(B) < 1$. It holds that*

$$\lim_{k \rightarrow \infty} B^k = 0 \iff \rho(B) < 1. \quad (11.1.19)$$

Proof. We will show that the following four conditions are equivalent:

- (i) $\lim_{k \rightarrow \infty} B^k = 0,$
- (ii) $\lim_{k \rightarrow \infty} B^k x = 0, \quad \forall x \in \mathbf{C}^n,$
- (iii) $\rho(B) < 1,$
- (iv) $\|B\| < 1 \quad \text{for at least one matrix norm.}$

For any vector x we have the inequality $\|B^k x\| \leq \|B^k\| \|x\|$, which shows that (i) implies (ii). If $\rho(B) \geq 1$, then there is an eigenvector $x \in \mathbf{C}^n$ 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 10.8.1 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

From this theorem we deduce the following necessary and sufficient criterion for the convergence of a stationary iterative method.

Theorem 11.1.4. *The stationary iterative method $x^{(k+1)} = Bx^{(k)} + c$ is convergent for all initial vectors $x^{(0)}$ if and only if $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of B . A sufficient condition for convergence of the iterative method is that $\|B\| < 1$, for some matrix norm.*

Proof. From the recurrence (11.1.14) it follows that

$$x^{(k)} - x = B^k(x^{(0)} - x). \quad (11.1.20)$$

Hence $x^{(k)}$ converges for all initial vectors $x^{(0)}$ if and only if $\lim_{k \rightarrow \infty} B^k = 0$. The theorem now follows from Theorem 11.1.3. \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 nonnormality are discussed in Section 11.1.7.

Usually, we are not only interested in convergence, but also in the *rate of convergence*. Taking norms in (11.1.20) we get

$$\|e^{(k)}\| \leq \|B^k\| \|e^{(0)}\|.$$

where $e^{(k)} = x^{(k)} - x$ is the error at step k . On the 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 11.1.5. *For any consistent matrix norm it holds that*

$$\lim_{k \rightarrow \infty} (\|B^k\|)^{1/k} = \rho(B), \quad (11.1.21)$$

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

$$k \geq -\log \delta / R_k(B), \quad R_k(B) = -\frac{1}{k} \log \|B^k\|.$$

The above results motivate the following definitions:

Definition 11.1.6. *Assume that the iterative method $x^{(k+1)} = Bx^{(k)} + c$ is convergent. Then, for any consistent matrix norm $\|\cdot\|$,*

$$R_k(B) = -\frac{1}{k} \log \|B^k\|, \quad (\|B^k\| < 1). \quad (11.1.22)$$

is the average rate of convergence. Further, the asymptotic rate of convergence is

$$R_\infty(B) = \lim_{k \rightarrow \infty} R_k(B) = -\log \rho(B). \quad (11.1.23)$$

We now consider the convergence of the classical methods introduced in Section 11.1.3.

Theorem 11.1.7.

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 for $0 < \omega < 2/b$.

Proof. The iteration matrix of the stationary Richardson's method is $B = I - \omega A \in \mathbf{R}^{n \times n}$, with eigenvalues $\mu_i = 1 - \omega \lambda_i$. From the assumption $1 - \omega b \leq \mu_i \leq 1 - \omega a$, for all i . It follows that if $1 - \omega a < 1$ and $1 - \omega b > -1$, then $|\mu_i| < 1$ for all i and the method is convergent. Since $a > 0$ the first condition is satisfied for all $\omega > 0$, while the second is true if $\omega < 2/b$. \square

Assuming that $a = \lambda_{min}$ and $b = \lambda_{max}$. 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? It is left as an exercise to show that this optimal ω is that which satisfies $1 - \omega a = \omega b - 1$, i.e $\omega_{opt} = 2/(b+a)$. (*Hint:* Plot the graphs of $|1 - \omega a|$ and $|1 - \omega b|$ for $\omega \in (0, 2/b)$.) It follows that

$$\rho(B) = \frac{b-a}{b+a} = \frac{\kappa-1}{\kappa+1} = 1 - \frac{2}{\kappa+1},$$

where $\kappa = b/a$ is the condition number of A . Hence the optimal asymptotic rate of convergence is

$$R_\infty(B) = -\log\left(1 - \frac{2}{\kappa+1}\right) \approx 2/\kappa, \quad \kappa \gg 1. \quad (11.1.24)$$

is inversely proportional to κ . This illustrates a typical fact for iterative methods: *in general ill-conditioned systems require more work to achieve a certain accuracy!*

Theorem 11.1.8. *The Jacobi method is convergent if A is strictly rowwise diagonally dominant, i.e.,*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad i = 1 : n.$$

Proof. For the Jacobi method the iteration matrix $B_J = L + U$ has elements $b_{ij} = -a_{ij}/a_{ii}$, $i \neq j$, $b_{jj} = 0$, $i = j$. From the assumption it then follows that

$$\|B_J\|_\infty = \max_{1 \leq i \leq n} \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|/|a_{ii}| < 1,$$

which proves the theorem. \square

A similar result for strictly columnwise diagonally dominant matrices can be proved using $\|B_J\|_1$. A slightly stronger convergence result than in Theorem 11.1.8 is of importance in applications. (Note that, e.g., the matrix A in (11.1.6) is not strictly diagonal dominant.) For irreducible matrices (see Definition 7.1.3) the row sum criterion in Theorem 11.1.8 can be sharpened.

Theorem 11.1.9. *The Jacobi method is convergent if A is irreducible, and in*

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad i = 1 : n,$$

inequality holds for at least one row.

The column sum criterion can be similarly improved. The conditions in Theorem 11.1.8–11.1.9 are also sufficient for convergence of the Gauss–Seidel method for which $(I - L)B_{GS} = U$. Consider the strictly rowwise diagonally dominant and choose k so that

$$\|B_{GS}\|_\infty = \|B_{GS}^T\|_1 = \|B_{GS}^T e_k\|_1.$$

Then from $B_{GS}^T e_k = B_{GS}^T L^T e_k + U^T e_k$, we get

$$\|B_{GS}\|_\infty \leq \|B_{GS}\|_\infty \|L^T e_k\|_1 + \|U^T e_k\|_1.$$

Since A is strictly rowwise diagonally dominant we have $\|L^T e_k\|_1 + \|U^T e_k\|_1 \leq \|B_J\|_\infty < 1$, and 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 $\|B_{GS}\|_1$.

Example 11.1.2.

In Section 11.1.3 it was shown that the $(n - 1)^2$ eigenvalues of the matrix

$$A = (I \otimes T) + (T \otimes I)$$

arising from the model problem are equal to

$$(\lambda_i + \lambda_j), \quad i, j = 1 : n - 1, \quad \lambda_i = 2(1 + \cos(i\pi/n)).$$

It follows that the eigenvalues of the corresponding Jacobi iteration matrix $B_J = L + U = (1/4)(A - 4I)$ are

$$\mu_{ij} = \frac{1}{2}(\cos i\pi h + \cos j\pi h), \quad i, j = 1 : n - 1,$$

where $h = 1/n$ is the grid size. 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.⁶³ 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 $R_\infty(B_{GS}) \approx \pi^2 h^2$. This shows that for the model problem Gauss–Seidel’s method will converge asymptotically twice as fast as Jacobi’s method. However, for both methods the number of iterations required is proportional to $\kappa(A)$ for the model problem.

The rate of convergence of the basic Jacobi and Gauss–Seidel methods, as exhibited in the above example, is in general much too slow to make them of any practical use. In Section 11.1.5 we show that with a simple modification, the the rate of convergence of the Gauss–Seidel method for the model problem can be improved by a factor of n .

Many matrices arising from the discretization of partial differential equations have the following property:

Definition 11.1.10.

A matrix $A = (a_{ij})$ is an *M-matrix* if $a_{ij} \leq 0$ for $i \neq j$, A is nonsingular and $A^{-1} \geq 0$.

In particular the matrix arising from the model problem is a symmetric *M-matrix*. Such a matrix is also called a **Stieltjes** matrix.

⁶³This is one of the basic observations used in the multigrid method, which uses a sequence of different meshes to efficiently damp all frequencies.

Often the matrices M and N in the splitting (11.1.11) of the matrix A has special properties that can be used in the analysis. Of particular interest is the following property.

Definition 11.1.11.

*For a matrix $A \in \mathbf{R}^{n \times n}$, the splitting $A = M - N$ is a **regular splitting** if M is nonsingular, $M^{-1} \geq 0$ and $N \geq 0$.*

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 the matrix A has a positive diagonal and the off diagonal elements were non-negative. Clearly this ensures that the Jacobi and Gauss–Seidel methods both correspond to a regular splitting. For regular splittings several results comparing asymptotic rates of convergence can be obtained.

Theorem 11.1.12.

If $A = M - N$ is a regular splitting of the matrix A and $A^{-1} \geq 0$, then

$$\rho(M^{-1}N) = \frac{\rho(A^{-1}N)}{1 + \rho(A^{-1}N)} < 1. \quad (11.1.25)$$

Thus the iterative method (11.1.12) converges for any initial vector $x^{(0)}$.

Proof. See Varga [594, Theorem 3.13]. \square

11.1.5 Successive Overrelaxation Methods

It was noted early that a great improvement in the rate of convergence could be obtained by the simple means of introducing a **relaxation parameter** ω in the Gauss–Seidel method. The iteration then becomes

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}} r_i^{(k)}, \quad (11.1.26)$$

where $\omega > 1$ (over-relaxation) or $\omega < 1$ (under-relaxation). The optimal value of ω for an important class of problems was found in 1950 by Young⁶⁴. This led to the famous **Successive Over Relaxation** (SOR) method, which for a long time remained an important “workhorse” in scientific computing.

Using the standard splitting

$$A = D_A - L_A - U_A = D_A(I - L - U),$$

⁶⁴David M. Young (1922–2008) was one of the pioneers of modern scientific computing. His classic dissertation [622] 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.

introduced in Section 11.1.3 the SOR method can be written in matrix form as

$$x^{(k+1)} = x^{(k)} + \omega \left(c + Lx^{(k+1)} - (I - U)x^{(k)} \right), \quad (11.1.27)$$

where $c = D_A^{-1}b$, or after rearranging

$$(I - \omega L)x^{(k+1)} = [(1 - \omega)I + \omega U]x^{(k)} + \omega c.$$

The iteration matrix for SOR therefore is

$$B_\omega = (I - \omega L)^{-1}[(1 - \omega)I + \omega U]. \quad (11.1.28)$$

We now consider the convergence of the SOR method and first show that only values of ω , $0 < \omega < 2$ are of interest.

Lemma 11.1.13.

Let $B = L + U$ be any matrix with zero diagonal and let B_ω be given by (11.1.28). Then

$$\rho(B_\omega) \geq |\omega - 1|, \quad (11.1.29)$$

with equality only if all the eigenvalues of B_ω are of modulus $|\omega - 1|$. Hence the SOR method can only converge 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

The following theorem asserts that if A is a symmetric positive definite matrix, then the SOR method converges for $0 < \omega < 2$.

Theorem 11.1.14. *For a symmetric positive definite matrix A we have*

$$\rho(B_\omega) < 1, \quad \forall \omega, \quad 0 < \omega < 2.$$

Proof. We defer the proof to Theorem 11.2.4. \square

For an important class of matrices an explicit expression for the optimal value of ω can be given. We first introduce the class of matrices with **property A**.

Definition 11.1.15. *The matrix A is said to have property A if there exists a permutation matrix P such that PAP^T has the form*

$$\begin{pmatrix} D_1 & U_1 \\ L_1 & D_2 \end{pmatrix}, \quad (11.1.30)$$

where D_1, D_2 are diagonal matrices.

Equivalently, the matrix $A \in \mathbf{R}^{n \times n}$ 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 $i \in T, j \in S$. For example, for the tridiagonal matrix A

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}, \quad P^T AP = \begin{pmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{pmatrix}$$

has property A, and we can choose $S = \{1, 3\}$, $T = \{2, 4\}$. Permutation of column 1 and 4 followed by a similar row permutation will give a matrix of the form above.

Definition 11.1.16.

*A matrix A with the decomposition $A = D_A(I - L - U)$, D_A nonsingular, is said to be **consistently ordered** if the eigenvalues of*

$$J(\alpha) = \alpha L + \alpha^{-1}U, \quad \alpha \neq 0,$$

are independent of α .

A matrix of the form of (11.1.30) is consistently ordered. To show this we note that since

$$J(\alpha) = \begin{pmatrix} 0 & -\alpha^{-1}D_1^{-1}U_1 \\ -\alpha D_2^{-1}L_1 & 0 \end{pmatrix} = -\begin{pmatrix} I & 0 \\ 0 & \alpha I \end{pmatrix} J(1) \begin{pmatrix} I & 0 \\ 0 & \alpha^{-1}I \end{pmatrix},$$

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 the order $1, 3, 5, \dots, 2, 4, 6, \dots$

Theorem 11.1.17.

Let $A = D_A(I - L - U)$ be a consistently ordered matrix. Then, if 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

$$\mu = \frac{\lambda + \omega - 1}{\omega \lambda^{1/2}} \quad (11.1.31)$$

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 $\det(\lambda I - B_\omega) = 0$, or since $\det(I - \omega L) = 1$ for all ω , using (11.1.28)

$$\det[(I - \omega L)(\lambda I - B_\omega)] = \det[\lambda(I - \omega L) - (1 - \omega)I - \omega U] = 0.$$

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 since A is consistently ordered it follows that $\det(\mu I - (L + U)) = 0$, where μ given by (11.1.31). Hence μ is an eigenvalue of $L + U$. \square

If we put $\omega = 1$ in (11.1.31) we get $\lambda = \mu^2$. Since $\omega = 1$ corresponds to the Gauss–Seidel method it follows that $\rho(B_{GS}) = \rho(B_J)^2$, which means that Gauss–Seidel’s method converges twice as fast as Jacobi’s method. for all consistently ordered matrices A

We now state an important result due to Young [622].

Theorem 11.1.18.

Let A be a consistently ordered matrix, and assume that the eigenvalues μ of $B_J = L + U$ are real and $\rho_J = \rho(B_J) < 1$. Then the optimal relaxation parameter ω in SOR is given by

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho_J^2}}. \quad (11.1.32)$$

For this optimal value we have

$$\rho(B_{\omega_{opt}}) = \omega_{opt} - 1. \quad (11.1.33)$$

Proof. (See also Young [623, Section 6.2].) We consider, for a given value of μ in the range $0 < \mu \leq \rho(L + U) < 1$, the two functions of λ ,

$$f_\omega(\lambda) = \frac{\lambda + \omega - 1}{\omega}, \quad g(\lambda, \mu) = \mu \lambda^{1/2}.$$

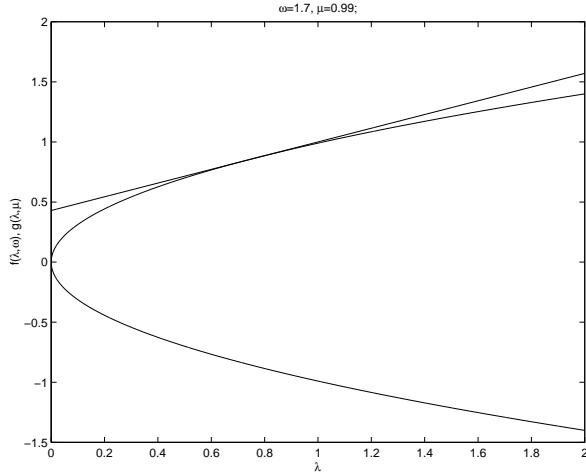


Figure 11.1.1. $f_\omega(\lambda)$ and $g(\lambda, \mu)$ as functions of λ ($\mu = 0.99$, $\omega = \omega_b = 1.7527$).

Here $f_\omega(\lambda)$ is a straight line passing through the points $(1, 1)$ and $(1-\omega, 0)$, and $g(\lambda, \mu)$ a parabola. The relation (11.1.31) can now be interpreted as the intersection of these two curves. For given μ and ω we get for λ the quadratic equation

$$\lambda^2 + 2\left((\omega - 1) - \frac{1}{2}\mu^2\omega^2\right)\lambda + (\omega - 1)^2 = 0. \quad (11.1.34)$$

which has two 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 Figure 11.2.1) Solving for the root $\omega \leq 2$ gives

$$\tilde{\omega} = \frac{1 - (1 - \mu^2)^{1/2}}{1/2\mu^2} = \frac{2}{1 + \sqrt{1 - \mu^2}}.$$

If $\omega > \tilde{\omega}$, equation (11.1.34) has two complex roots λ . By the relation between roots and coefficients in (11.1.34) these satisfy

$$\lambda_1\lambda_2 = (\omega - 1)^2.$$

and $|\lambda_1| = |\lambda_2| = \omega - 1$ for $1 < \tilde{\omega} < \omega < 2$. It follows that the minimum value of $\max_{i=1,2} |\lambda_i|$ occurs for $\tilde{\omega}$. 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

Table 11.1.1. Number of iterations needed to reduce the initial error by a factor of 10^{-3} for the model problem, as a function of $n = 1/h$.

n	10	20	50	100	200
Gauss-Seidel	69	279	1,749	6,998	27,995
SOR	17	35	92	195	413

Example 11.1.3.

By (11.1.32) for SOR $\omega_{opt} = 2/(1 + \sin \pi h)$, giving

$$\rho(B_{\omega_{opt}}) = \omega_{opt} - 1 = \frac{1 - \sin \pi h}{1 + \sin \pi h} \approx 1 - 2\pi h. \quad (11.1.35)$$

Note that when $\lim_{n \rightarrow \infty} \omega_{opt} = 2$.

$$R_\infty(B_{\omega_{opt}}) \approx 2\pi h,$$

which shows that for the model problem the number of iterations is proportional to n for the SOR method

In Table 10.1.1 we give the number of iterations required to reduce the norm of the initial error by a factor of 10^{-3} .

In practice, the number ρ_J is seldom known a priori, and its accurate determination would be prohibitively expensive. However, for some model problems the spectrum of the Jacobi iteration matrix is known. In the following we need the result:

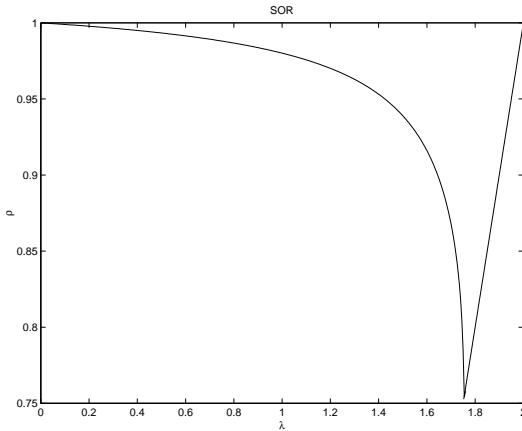


Figure 11.1.2. The spectral radius $\rho(B_\omega)$ as a function of ω ($\rho = 0.99$, $\omega_b = 1.7527$).

A simple scheme for estimating ω_{opt} is to initially perform a fixed number of iterations using $\omega = 1$, i.e., with the Gauss-Seidel method, and attempt to 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

$$\rho(B_J)^2 = \rho(B_{GS}) \approx \theta_n, \quad \theta_n = \|\delta^{(n+1)}\|_\infty / \|\delta^{(n)}\|_\infty,$$

An estimate of ω_{opt} is then obtained by substituting this value into (11.1.32). A closer analysis shows, however, that the number of iterations to obtain a good estimate of ω_{opt} 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 have been developed, see Young [623, p. 210].

In more complicated cases when ρ_J is not known, we have to estimate ω_{opt} in the SOR method. In Figure 11.2.2 we have plotted the spectral radius $\rho(B_\omega)$ 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_{opt} than to underestimate it.

As remarked above the iteration matrix B_ω of the SOR-method is **not** symmetric and its eigenvalues are not real. In fact, in case ω is chosen slightly larger than optimal (as recommended when ρ_J is not known) the extreme eigenvalues of B_ω lie on a circle in the complex plane. However, a symmetric version of SOR, the (**SSOR**) method of Sheldon (1955), can be constructed as follows. One iteration consists of two half iterations. 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. The SSOR method can be written in matrix form as

$$\begin{aligned} x^{(k+1/2)} &= x^{(k)} + \omega \left(c + Lx^{(k+1/2)} - (I - U)x^{(k)} \right), \\ x^{(k+1)} &= x^{(k+1/2)} + \omega \left(c + Ux^{(k+1)} - (I - L)x^{(k+1/2)} \right). \end{aligned}$$

This method is due to Sheldon [1955]. The iteration matrix for SSOR is

$$S_\omega = (I - \omega U)^{-1}[(1 - \omega)I + \omega L](I - \omega L)^{-1}[(1 - \omega)I + \omega U].$$

It can be shown that SSOR corresponds to a splitting with the matrix

$$M_\omega = \frac{\omega}{2 - \omega} \left(\frac{1}{\omega} D_A - L_A \right) D_A^{-1} \left(\frac{1}{\omega} D_A - U_A \right). \quad (11.1.36)$$

If A is symmetric, positive definite then so is M_ω . In this case the SSOR method is convergent for all $\omega \in (0, 2)$. A proof of this is obtained by a simple modification of the proof of Theorem 11.2.4.

In contrast to the SOR method, the rate of convergence of SSOR is not very sensitive to the choice of ω nor does it assume that A is consistently ordered. It

can be shown (see Axelsson [19]) that provided $\rho(LU) < 1/4$ a suitable value for ω is ω_b , where

$$\omega_b = \frac{2}{1 + \sqrt{2(1 - \rho_J)}}, \quad \rho(S_{\omega_b}) \leq \frac{1 - \sqrt{(1 - \rho_J)/2}}{1 + \sqrt{(1 - \rho_J)/2}}.$$

In particular, for the model problem in Section 11.1.3 it follows that

$$\rho(B_{\omega_b}) \leq \frac{1 - \sin \pi h/2}{1 + \sin \pi h/2} \approx 1 - \pi h.$$

This is half the rate of convergence for SOR with ω_{opt} .

Definition 11.1.19.

Consider the stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \quad k = 0, 1, \dots, \quad (11.1.37)$$

corresponding to the matrix splitting $A = M - N$, and with iteration matrix $B = I - M^{-1}A$. It is said to be **symmetrizable** if there is a nonsingular matrix W such that the matrix $W(I - B)W^{-1}$ is symmetric and positive definite.

Example 11.1.4.

If A is positive definite then in the standard splitting (11.1.15) $D_A > 0$, and hence the Jacobi method is symmetrizable with $W = D_A^{1/2}$. From (11.1.36) it follows that also the SSOR method is symmetrizable.

A sufficient condition for a method to be symmetrizable is that both A and the splitting matrix M are symmetric and positive definite, since then there is a matrix W such that $M = W^T W$, and

$$W(I - B)W^{-1} = WM^{-1}AW^{-1} = WW^{-1}W^{-T}AW^{-1} = W^{-T}AW^{-1},$$

which again is positive definite. For a symmetrizable method the matrix $M^{-1}A$ has real positive eigenvalues λ_i . Therefore Chebyshev acceleration can be used to improve the rate of convergence; see Section 11.1.6.

Block versions of the SOR, and SSOR methods can easily be developed. For SOR we have

$$A_{ii}(x_i^{(k+1)} - x_i^{(k)}) = \omega \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{(k+1)} - \sum_{j=i}^n A_{ij}x_j^{(k)} \right), \quad i = 1 : n.$$

(Taking $\omega = 1$ gives the Gauss–Seidel method.) Typically the rate of convergence is improved by a factor $\sqrt{2}$ compared to the point methods.

It can easily be verified that the SOR theory as developed in Theorems 11.1.14 and 11.1.17 are still valid in the block case. We have

$$B_\omega = (I - \omega L)^{-1}[(1 - \omega)I + \omega U],$$

where $L = D_A^{-1}L_A$ and $U = D_A^{-1}U_A$. 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 has spectral radius $\rho(B_J) < 1$. Then the optimal value of ω in the SOR method is given by (11.1.32). Note that with the block splitting any block-tridiagonal matrix

$$A = \begin{pmatrix} D_1 & U_1 & & & \\ L_2 & D_2 & U_2 & & \\ & L_3 & \ddots & \ddots & \\ & & \ddots & \ddots & U_{n-1} \\ & & & L_n & D_n \end{pmatrix},$$

is consistently ordered; for the point methods this was true only in case the block diagonal matrices D_i , $i = 1 : n$ were diagonal. In particular we conclude that with the block splitting the matrix A in (11.1.1) for the model problem is consistently ordered so the SOR theory applies.

11.1.6 Chebyshev Acceleration

The non-stationary Richardson iteration is

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots, \quad (11.1.38)$$

where $x^{(0)}$ is a given initial vector and $\omega_k > 0$, $k = 0, 1, 2, \dots$, are parameters (cf. (11.1.9)). For simplicity, we assume that $x^{(0)} = 0$. (This is no restriction, since otherwise we can make the change of variables $x := x - x^{(0)}$.) Then the residual vector $r^{(k)} = b - Ax^{(k)}$ can be written

$$r^{(k)} = \prod_{i=0}^{k-1} (I - \omega_i A)b = q_k(A)b, \quad q_k(z) = \prod_{i=0}^{k-1} (1 - \omega_i z). \quad (11.1.39)$$

It follows that $r^{(k)} \in \mathcal{K}_k(A, b)$. The **residual polynomial** q_k is of degree k with zeros $1/\omega_i$, $i = 0 : k-1$, and such that $q_k(0) = 1$. Any desired residual polynomial can be obtained by an appropriate choice of the parameters $\{\omega_i\}_{i=0}^{k-1}$. In the following we denote the set of residual polynomials q_k of degree $\leq k$ such that $q_k(0) = 1$ by Π_k^1 .

By choosing a sequence of suitable residual polynomials the rate of convergence of the iteration (11.1.38) can be improved. Because of the relation (11.1.39) this process is called **polynomial acceleration**. From (11.1.39) we get the estimate

$$\|r^{(k)}\|_2 = \|q_k(A)\|_2 \|b\|_2,$$

Assume that A is a Hermitian matrix with eigenvalues $\{\lambda_i\}_{i=1}^n$. Then $q_k(A)$ is Hermitian with eigenvalues $q_k(\lambda_i)_{i=1}^n$, and

$$\|q_k(A)\|_2 = \max_i |q_k(\lambda_i)|.$$

In general, the eigenvalues λ_i are unknown. However, on the basis of some assumption regarding the distribution of eigenvalues, we may be able to select a set S such

that $\lambda_i \in \mathcal{S}$, $i = 1 : n$. After k steps of the accelerated method the 2-norm of the residual is reduced by at least a factor of

$$\rho(q_k(A)) = \max_i |q_k(\lambda_i)| \leq \max_{\lambda \in \mathcal{S}} |q_k(\lambda)|.$$

Thus, finding a suitable residual polynomial q_k is reduced to the polynomial approximation problem

$$\min_{q \in \Pi_k^1} \max_{\lambda \in \mathcal{S}} |q(\lambda)|. \quad (11.1.40)$$

Chebyshev acceleration is based on the properties of Chebyshev polynomials;⁶⁵ see [132, Section .3.2.3]. The Chebyshev polynomials of the first kind are defined by

$$T_n(z) = \cos(n\phi), \quad z = \cos \phi. \quad (11.1.41)$$

The Chebyshev polynomials satisfy a three term recurrence relation (see Section 3.2.3)
 $T_0(z) = 1$, $T_1(z) = zT_0$,

$$T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z), \quad k \geq 1. \quad (11.1.42)$$

Hence, the leading coefficient of $T_n(x)$ is 2^{n-1} . From (11.1.41) it follows that $|T_n(z)| \leq 1$ for $z \in [-1, 1]$. Outside the interval $[-1, 1]$ the Chebyshev polynomials grow rapidly. Setting $w = e^{i\phi} = \cos \phi + i \sin \phi$, we have

$$z = \frac{1}{2}(w + w^{-1}),$$

and

$$T_k(z) = \frac{1}{2}(w^k + w^{-k}), \quad w = z \pm \sqrt{z^2 - 1}, \quad -\infty < z < \infty. \quad (11.1.43)$$

This is valid also for complex values of z .

The Chebyshev polynomials have the following important **minimax property**: Of all n th degree polynomials with leading coefficient 1, the polynomial $2^{1-n}T_n(x)$ has the smallest magnitude 2^{1-n} in $[-1, 1]$ (see [132, Lemma 3.2.4]). This makes them useful for convergence acceleration.

If the eigenvalues of A satisfy $0 < a \leq \lambda_i \leq b$, the relevant minimization problem is (11.1.40) with $\mathcal{S} = [a, b]$.

Theorem 11.1.20.

Let $[a, b]$ be a nonempty real interval and let $\gamma > b$ be a real scalar. Then the minimum

$$\min_{q \in \Pi_k^1} \max_{\lambda \in [a, b]} |q(\lambda)|. \quad (11.1.44)$$

is reached by the shifted and normalized Chebyshev polynomial

$$\hat{T}_k(\lambda) = T_k(z(\lambda))/T_k(z(0)), \quad (11.1.45)$$

⁶⁵Pafnuth 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 $T_n(x)$. He also made important contributions to probability theory and number theory.

where $\hat{T}_k(0) = 1$, and

$$z(\lambda) = \frac{b+a-2\lambda}{b-a} = \mu - \frac{2}{b-a}\lambda, \quad \mu = z(0) = \frac{b+a}{b-a} > 1. \quad (11.1.46)$$

The linear transformation (11.1.46) maps the interval $0 < a \leq \lambda < b$ onto $z \in [-1, 1]$. (Note that a is mapped onto $+1$ and b onto -1 .)

A stable way to implement Chebyshev acceleration is based on the three term recurrence relation (11.1.42). By (11.1.45) $T_k(z(\lambda)) = T_k(\mu)q_k(\lambda)$, and the residual polynomials q_k also satisfy a three term recurrence relation. Using (11.1.46) and substituting A for λ , we obtain from (11.1.42)

$$T_{k+1}(\mu)q_{k+1}(A) = 2\left(\mu I - \frac{2}{b-a}A\right)T_k(\mu)q_k(A) - T_{k-1}(\mu)q_{k-1}(A).$$

Multiplying by $(x^{(0)} - x)$ from the right, and using $q_k(A)(x^{(0)} - x) = x^{(k)} - x$, we obtain

$$T_{k+1}(\mu)(x^{(k+1)} - x) = 2\left(\mu I - \frac{2}{b-a}A\right)T_k(\mu)(x^{(k)} - x) - T_{k-1}(\mu)(x^{(k-1)} - x).$$

Subtracting (11.1.42) with $z = \mu$ it then follows that

$$T_{k+1}(\mu)x^{(k+1)} = 2\mu T_k(\mu)x^{(k)} + \frac{4T_k(\mu)}{b-a}A(x - x^{(k)}) - T_{k-1}(\mu)x^{(k-1)}.$$

Substituting $-T_{k-1}(\mu) = -2\mu T_k(\mu) + T_{k+1}(\mu)$ and dividing with $T_{k+1}(\mu)$ we get

$$x^{(k+1)} = x^{(k-1)} + \delta_k(b - Ax^{(k)}) + \omega_k(x^{(k)} - x^{(k-1)}), \quad k \geq 1,$$

where

$$\alpha = 2/(b+a), \quad \omega_k = 2\mu \frac{T_k(\mu)}{T_{k+1}(\mu)}, \quad \delta_k = \alpha \omega_k.$$

This is a three term recurrence for computing the accelerated approximate solution. A similar calculation for $k = 0$ gives $x^{(1)} = x^{(0)} + \alpha(b - Ax^{(0)})$, and we have derived the following algorithm:

Algorithm 11.1. *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

$$\mu = (b+a)/(b-a), \quad \alpha = 2/(b+a),$$

and $x^{(1)} = x^{(0)} + \alpha(b - Ax^{(0)})$. For $k = 1, 2, \dots$, compute

$$x^{(k+1)} = x^{(k-1)} + \omega_k(\alpha(b - Ax^{(k)}) + x^{(k)} - x^{(k-1)}), \quad (11.1.47)$$

where

$$\omega_0 = 2, \quad \omega_k = \left(1 - \frac{\omega_{k-1}}{4\mu^2}\right)^{-1}, \quad k \geq 1.$$

From (11.1.46) it follows that $\mu = (\kappa + 1)/(\kappa - 1)$, where $\kappa = b/a$ is an upper bound for the spectral condition number of A . Using (11.1.43)

$$w = \mu + \sqrt{\mu^2 - 1} = \frac{\kappa + 1}{\kappa - 1} + \frac{2\sqrt{\kappa}}{\kappa - 1} = \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} > 1,$$

$\rho(q_k(A)) \leq 1/T_k(\mu) > 2e^{-k\gamma}$, where after some simplification

$$\gamma = \log \left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} \right) > \frac{2}{\sqrt{\kappa}}. \quad (11.1.48)$$

(Verify the last inequality! See Problem 11.1.2.) Hence to reduce the error norm at least by a factor of $\delta < 1$ it suffices to perform k iterations, where

$$k > \frac{1}{2} \sqrt{\kappa} \log \frac{2}{\delta}, \quad (11.1.49)$$

Thus the number of iterations required for a certain accuracy for the Chebyshev accelerated method is proportional to $\sqrt{\kappa}$ rather than κ —a great improvement!

Chebyshev acceleration can be applied to any stationary iterative method (11.1.37). provided it is symmetrizable. This iteration corresponds to a matrix splitting $A = M - N$, and iteration matrix $B = M^{-1}N = I - M^{-1}A$. (In the simplest case we have $M = I$ and hence $B = I - A$.)

The eigenvalues of the iteration matrix of the SOR-method $B_{\omega_{opt}}$ are all complex and have modulus $|\omega_{opt}|$. Therefore, in this case convergence acceleration cannot be used. (A precise formulation is given in Young [623, p. 375].) However, Chebyshev acceleration can be applied to the Jacobi and SSOR methods, with

$$M_J = D_A, \quad M_\omega = \frac{\omega}{2 - \omega} \left(\frac{1}{\omega} D_A - L_A \right) D_A^{-1} \left(\frac{1}{\omega} D_A - U_A \right),$$

respectively, as well as block versions of these methods, often with a substantial gain in convergence rate.

Assuming that $M = I$ we obtain for the residual $\tilde{r}^{(k)} = b - A\tilde{x}^{(k)}$

$$\tilde{r}^{(k)} = A(x - \tilde{x}^{(k)}) = q_k(A)b, \quad q_k(\lambda) = p_k(1 - \lambda). \quad (11.1.50)$$

where we have used that $A = I - B$.

The main drawback of Chebyshev acceleration is that 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.

11.1.7 Effects of Nonnormality and Finite Precision

As shown by Lemma 11.1.5 the spectral radius $\rho(B)$ determines the *asymptotic* rate of growth of matrix powers. The norm $\|B\|$ will influence the *initial* behavior of the

powers B^k . However, even if $\rho(B) < 1$, $\|B\|$ can be arbitrarily large for a nonnormal matrix B .

By the Schur normal form any matrix A is unitarily equivalent to an upper triangular matrix. Therefore, it suffices to consider the case of an upper triangular 2×2 matrix

$$B = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}, \quad 0 < \mu \leq \lambda < 1. \quad (11.1.51)$$

Since $\rho(B) < 1$, this matrix is convergent, i.e., $\|B^k\| \rightarrow 0$ as $k \rightarrow \infty$. But if $\alpha \gg 1$ then $\|B\|_2 \gg \rho(B)$, and the spectral norm $\|B^k\|_2$ will initially increase with k . It is easily verified that

$$B^k = \begin{pmatrix} \lambda^k & \beta_k \\ 0 & \mu^k \end{pmatrix}, \quad (11.1.52)$$

where

$$\beta_k = \begin{cases} \alpha \frac{\lambda^k - \mu^k}{\lambda - \mu} & \text{if } \mu \neq \lambda; \\ \alpha k \lambda^{k-1} & \text{if } \mu = \lambda. \end{cases}$$

Clearly the off-diagonal element β_k will grow initially. In the case that $\lambda = \mu$ the maximum of $|\beta_k|$ will occur when $k \approx \lambda/(1 - \lambda)$. (See also Computer Exercise 1.) For matrices of larger dimension the initial increase of $\|B^k\|$ can be huge as shown by the following example:

Example 11.1.5.

Consider the iteration $x^{(k+1)} = Bx^{(k)}$, where $B \in \mathbf{R}^{20 \times 20}$ is the bidiagonal matrix

$$B = \begin{pmatrix} 0.5 & 1 & & & \\ & 0.5 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0.5 & 1 \\ & & & & 0.5 \end{pmatrix}, \quad x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Here $\rho(B) = 0.5$, and hence the iteration should converge to the exact solution of the equation $(I - B)x = 0$, which is $x = 0$. From Figure 11.1.2 it is seen that $\|x^{(n)}\|_2$ increases by more than a factor 10^5 until it starts to decrease after 35 iterations. Although in the long run the norm is reduced by about a factor of 0.5 at each iteration, large intermediate values of $x^{(n)}$ give rise to persistent rounding errors.

The curve in Figure 11.1.2 shows a large hump, i.e., $\|x^{(k)}\|_2$ increases substantially, even though the iteration eventually converges. In such a case cancellation will occur in the computation of the final solution, and a rounding error of size $u \max_k \|x^{(k)}\|_2$ remains, where u is the machine unit.

For the case when the iteration process is carried out in exact arithmetic, there is a complete and simple mathematical theory of convergence for iterates $x^{(k)}$ of stationary iterative methods. According to Theorem 11.1.3 there is asymptotic convergence for any $x^{(0)}$ if and only if $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of B . In finite precision arithmetic the convergence behavior is more complex

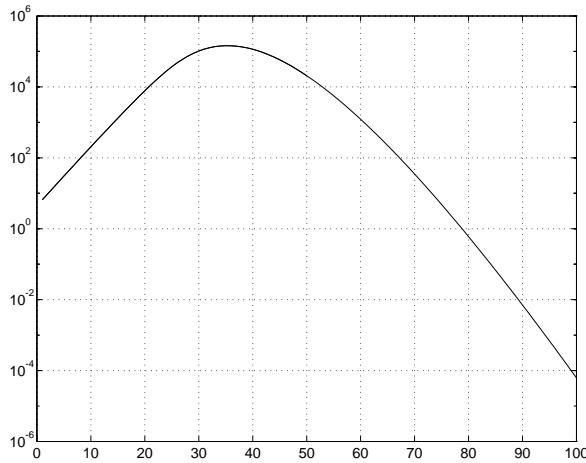


Figure 11.1.3. $\|x^{(n)}\|_2$, where $x^{(k+1)} = Bx^{(k)}$, and $x^{(0)} = (1, 1, \dots, 1)^T$.

and less easy to analyze. When B is nonnormal the iteration process can be badly affected by rounding errors. Moreover, asymptotic convergence in finite precision arithmetic is no longer guaranteed even when $\rho(B) < 1$ holds in exact arithmetic. 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 $B + \Delta B$. Hence even though $\rho(B) < 1$ it may be that $\rho(B + \Delta B)$ is larger than one. To have convergence in finite precision arithmetic we need a stronger condition to hold, e.g.,

$$\max \rho(B + E) < 1, \quad \|E\|_2 < u\|B\|_2,$$

where u is the machine precision. (Compare the discussion of pseudospectra in Section 10.7.6. The following rule of thumb has been suggested:

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!

It may be thought that iterative methods are less affected by rounding errors than direct solution methods, because in iterative methods one continues to work with the original matrix instead of modifying it. In Section 7.5 we showed that the total effect of rounding errors in Gaussian elimination with partial pivoting usually is equivalent to a perturbations in the elements of the original matrix of the order of machine roundoff. It is easy to verify that, in general, iterative methods cannot be expected to do much better than that.

Consider an iteration step with the Gauss–Seidel method 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 for δ_i , cf. Section 2.4.1. This can be interpreted that we have performed 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 of magnitude that 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 11.1.6.

(J. H. Wilkinson) Consider the (ill-conditioned) system $Ax = b$, where

$$A = \begin{pmatrix} 0.96326 & 0.81321 \\ 0.81321 & 0.68654 \end{pmatrix}, \quad b = \begin{pmatrix} 0.88824 \\ 0.74988 \end{pmatrix}.$$

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. Starting with $x_1 = 0.33116$, $x_2 = 0.70000$, the next approximation for x_1 is computed from the relation

$$x_1 = fl\left((0.88824 - 0.81321 \cdot 0.7)/0.96326\right) = 0.33116,$$

(working with five decimals). This would be an exact result if the element a_{11} was perturbed to be $0.963259\dots$, but no progress is made towards 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 to be made in each step are less than $0.5 \cdot 10^{-5}$.

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.

Normally a user would like to specify an absolute (or a relative) tolerance ϵ for the error, and stop as soon as

$$\|x - x^{(k)}\| \leq \epsilon \tag{11.1.53}$$

is satisfied for some suitable vector norm $\|\cdot\|$. However, such a criterion cannot in general be implemented since x is unknown. Moreover, if the system to be solved is illconditioned, then because of roundoff the criterion (11.1.53) may never be satisfied.

Instead of (11.1.53) one can use a test on the residual vector $r^{(k)} = b - Ax^{(k)}$, which is computable, and stop when

$$\|r^{(k)}\| \leq \epsilon(\|A\| \|x^{(k)}\| + \|b\|).$$

This is often replaced by the stricter criterion

$$\|r^{(k)}\| \leq \epsilon\|b\|, \quad (11.1.54)$$

but this may be difficult to satisfy in case $\|b\| \ll \|A\|\|x\|$. (To use $r^{(0)}$ instead of b in the criterion is not usually recommended, since 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 in the approximate solution. Since $x - x^{(k)} = A^{-1}r^{(k)}$, (11.1.54) only guarantees 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 7.5.9. The idea is then to compute the quantity

$$\omega = \max_i \frac{|r_i^{(k)}|}{(E|x^{(k)}| + f)_i}, \quad (11.1.55)$$

where $E > 0$ and $f > 0$, and stop when $\omega \leq \epsilon$. It then follows from Theorem 7.5.9 that $x^{(k)}$ is the exact solution to a perturbed linear system

$$(A + \delta A)x = b + \delta b, \quad |\delta A| \leq \omega E, \quad |\delta b| \leq \omega f.$$

We could in (11.1.55) take $E = \|A\|$ and $f = \|b\|$, which corresponds to componentwise backward errors. However, it can be argued that for iterative methods a more suitable choice is to use a normwise backward error by setting

$$E = \|A\|_\infty ee^T, \quad f = \|b\|_\infty e, \quad e = (1, 1, \dots, 1)^T.$$

This choice gives

$$\omega = \frac{\|r^{(k)}\|_\infty}{\|A\|_\infty \|x^{(k)}\|_1 + \|b\|_\infty}.$$

Review Questions

- 1.1** When is the matrix A reducible? Illustrate this property using the directed graph of A .
- 1.2** Let $A = D_A(I - L - U)$, where $D_A > 0$. When is A said to have “property A”. When is A consistently ordered? How are these properties related to the SOR method?
- 1.3** For the model problem the asymptotic rate of convergence for the classical iterative methods is proportional to h^p , where h is the mesh size. Give the value of p for Jacobi, Gauss–Seidel, SOR and SSOR. (For the last two methods it is assumed that the optimal ω is used.)

- 1.4** Consider an iterative method based on the splitting $A = M - N$. Give conditions on the eigenvalues of $M^{-1}A$ which are sufficient for Chebyshev acceleration to be used. Express the asymptotic rate of convergence for the accelerated method in terms of eigenvalue bounds.
- 1.5** Mention two types of linear systems, where iteration can be a good alternative to a direct method.
- 1.6** The standard discretization of Laplace equation on a square with Dirichlet boundary conditions leads to a certain matrix A . Give this matrix in its block triangular form.
- 1.7** (a) Give a sufficient condition for the convergence of the iterative method
- $$x^{(k+1)} = Bx^{(k)} + c.$$
- (b) How are B and c related to A and b in the Jacobi and Gauss–Seidel methods for the solution of $Ax = b$?
- 1.8** What iterative method can be derived from the splitting $A = M - N$? How is a symmetrizable splitting defined?
- 1.9** Does the condition $\rho(B) < 1$ imply that the error norm $\|x - x^{(k)}\|_2$ is monotonically decreasing for the iteration $x^{(k+1)} = Bx^{(k)} + c$. If not, give a counterexample.
- 1.10** Give at least two different criteria which are suitable for terminating an iterative method.

Problems and Computer Exercises

- 1.1** Let $A \in \mathbf{R}^{n \times n}$ be a given nonsingular matrix, and $X^{(0)} \in \mathbf{R}^{n \times n}$ an arbitrary matrix. Define a sequence of matrices by

$$X^{(k+1)} = X^{(k)} + X^{(k)}(I - AX^{(k)}), \quad k = 0, 1, 2, \dots$$

(a) Prove that $\lim_{k \rightarrow \infty} X^{(k)} = A^{-1}$ if and only if $\rho(I - AX^{(0)}) < 1$.

Hint: First show that $I - AX^{(k+1)} = (I - AX^{(k)})^2$.

(b) Use the iterations 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!

- 1.2** Show that if for a stationary iterative method $x^{(k+1)} = Bx^{(k)} + c$ it holds that $\|B\| \leq \beta < 1$, and

$$\|x^{(k)} - x^{(k-1)}\| \leq \epsilon(1 - \beta)/\beta,$$

then the error estimate $\|x - x^{(k)}\| \leq \epsilon$ holds.

1.3 Let B be the 2×2 matrix in (11.1.52), and take $\lambda = \mu = 0.99$, $\alpha = 4$. Verify that $\|B^k\|_2 \geq 1$ for all $k < 805$!

1.4 Let $B \in \mathbf{R}^{20 \times 20}$ be an upper bidiagonal matrix with diagonal elements equal to 0.025, 0.05, 0.075, ..., 0.5 and elements in the superdiagonal all equal to 5.

(a) Compute and plot $\eta_k = \|x^{(k)}\|_2 / \|x^{(0)}\|_2$, $k = 0 : 100$, where

$$x^{(k+1)} = Bx^{(k)}, \quad x^{(0)} = (1, 1, \dots, 1)^T.$$

Show that $\eta_k > 10^{14}$ before it starts to decrease after 25 iterations. What is the smallest k for which $\|x^{(k)}\|_2 < \|x^{(0)}\|_2$?

(b) Compute the eigendecomposition $B = X\Lambda X^{-1}$ and determine the condition number $\kappa = \|X\|_2 \|X^{-1}\|_2$ of the transformation.

1.5 (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 ?

1.6 The matrix A in (11.1.6) 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 $D_1(\alpha) = \text{diag}(1, \alpha, \dots, \alpha^{n-1})$, $D_{i+1}(\alpha) = \alpha D_i(\alpha)$, $i = 1 : n - 1$.

1.7 Verify the recursion for ω_k for the Chebyshev semi-iteration method.

1.8 Show that

$$\log((1+s)/(1-s)) = 2(s + s^3/3 + s^5/5 + \dots), \quad 0 \leq s < 1,$$

and use this result to prove (11.1.49).

1.9 Assume that A is symmetric indefinite with its eigenvalues contained in the union of two intervals of equal length,

$$\mathcal{S} = [a, b] \cup [c, d], \quad a < b < 0, \quad 0 < c < d,$$

where $d - c = b - a$. 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 $|b| \gg c$.

11.2 Projection Methods

11.2.1 Principles of Projection Methods

Consider a linear system $Ax = b$, where $A \in \mathbf{R}^{n \times n}$. Suppose we want to find an approximate solution \hat{x} in a subspace \mathcal{K} of dimension $m < n$. Then m independent conditions are needed to determine \hat{x} . One way to obtain these is by requiring that the residual $b - A\hat{x}$ is orthogonal to all vectors in a subspace \mathcal{L} of dimension m , i.e.,

$$\hat{x} \in \mathcal{K}, \quad b - A\hat{x} \perp \mathcal{L}. \quad (11.2.1)$$

Many important classes of methods can be interpreted as being projection methods in this general sense. If $\mathcal{K} = \mathcal{L}$, the conditions (11.2.1) are known as the **Galerkin conditions**⁶⁶ The generalization to $\mathcal{K} \neq \mathcal{L}$ is due to G. I. Petrov in 1946.

Example 11.2.1.

The truncated SVD solution $x_k = A_k^\dagger b$ (see Section 8.1.5) is the approximation corresponding to a projection method, where

$$x_k \in \mathcal{R}(V_k), \quad r_k \perp \in \mathcal{R}(U_k).$$

A matrix form of (11.2.1) can be obtained by introducing basis vectors in the two subspaces. Let $\mathcal{K} = \mathcal{R}(U)$, $\mathcal{L} = \mathcal{R}(V)$, where $U = (u_1, \dots, u_k)$, $V = (v_1, \dots, v_k)$. Then we can write (11.2.1) as

$$V^T(b - AUz) = 0, \quad z \in \mathbf{R}^k. \quad (11.2.2)$$

where $\hat{x} = Uz$. Hence \hat{x} is obtained by solving the reduced system

$$\hat{A}z = V^T b, \quad \hat{A} = V^T A U \in \mathbf{R}^{k \times k}. \quad (11.2.3)$$

We usually have $k \ll n$, and when k is small this system can be solved by a direct method.

Even though A is nonsingular the matrix \hat{A} may be singular. For example, take, $U = V = e_1$, and

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

⁶⁶Boris Galerkin (1871–1975) Russian mathematician. From 1908 he worked as a teacher in structural mechanics at the Saint Petersburg Polytechnical Institute. In 1915 he developed his method for solving boundary value problems for differential equations using a variational principle.

Then $\hat{A} = 0$. Note that the matrix A here is symmetric, but not positive definite. In two important special cases the matrix \hat{A} can be guaranteed to be nonsingular.

1. Let A by symmetric, positive definite (s.p.d.) and $\mathcal{L} = \mathcal{K}$. Then we can take $V = U$, and have $\hat{A} = U^T AU$. Clearly \hat{A} is s.p.d., and hence nonsingular. By (11.2.2)

$$\hat{x} = Uz, \quad z = (U^T AU)^{-1} U^T b.$$

2. Let A be nonsingular and $\mathcal{L} = A\mathcal{K}$, i.e., $y \in \mathcal{L}$, if and only if $y = Ax$, $x \in \mathcal{K}$. Then we get $\hat{A} = (AU)^T (AU)$ which is s.p.d. In this case

$$\hat{x} = Uz, \quad z = (U^T A^T AU)^{-1} U^T A^T b.$$

We now derive important optimality properties satisfied in these two special cases. For this purpose we first define a new inner product and norm related to a s.p.d. matrix A .

Definition 11.2.1. *For an s.p.d. matrix A we define a related A -inner product and A -norm, also called the **energy norm**⁶⁷, by*

$$(u, v)_A = u^T Av, \quad \|u\|_A = (u^T Au)^{1/2}, \quad (11.2.4)$$

It is easily verified that $\|u\|_A$ satisfies the conditions for a norm.

Lemma 11.2.2.

Let A be symmetric, positive definite matrix and $\mathcal{L} = \mathcal{K}$ ($V = U$). Then the energy norm of the error over all vectors $x \in \mathcal{K}$, is minimized by

$$\hat{x} = U(U^T AU)^{-1} U^T b,$$

i.e., \hat{x} solves the problem

$$\min_{x \in \mathcal{K}} \|x - x^*\|_A, \quad x^* = A^{-1}b. \quad (11.2.5)$$

Proof. By (11.2.2) \hat{x} satisfies $v^T(b - A\hat{x}) = 0$, $\forall v \in \mathcal{K}$. Let $\hat{e} = \hat{x} - x^*$ be the error in \hat{x} . Then for the error in $\hat{x} + v$, $v \in \mathcal{K}$ we have $e = \hat{e} + v$, and

$$\|e\|_A^2 = \hat{e}^T A \hat{e} + v^T A v + 2v^T A \hat{e}.$$

But here the last term is zero because $v^T A \hat{e} = v^T(A\hat{x} - b) = 0$. It follows that $\|e\|_A$ is minimum if $v = 0$. \square

A related result is obtained for the second case.

⁶⁷For some partial differential equation problems this name has a physical relevance.

Lemma 11.2.3.

Let A be nonsingular and consider the case $\mathcal{L} = A\mathcal{K}$, ($V = AU$). Then

$$\hat{x} = U(U^T A^T AU)^{-1}U^T A^T b$$

minimizes the 2-norm of the residual over all vectors $x \in \mathcal{K}$, i.e., \hat{x} solves

$$\min_{x \in \mathcal{K}} \|b - Ax\|_2. \quad (11.2.6)$$

Proof. Using $x = Uz$ we have $\|b - Ax\|_2 = \|b - AUz\|_2$, which is minimized when z satisfies the normal equations $U^T A^T AUz = U^T A^T b$. \square

In an iterative method *a sequence of projection steps* of the above form is taken. To cover this case we need to modify the above algorithms slightly, so that they can start from a given approximation x_k , $k = 1, 2, \dots$.⁶⁸ If we let $x = x_k + z$, then z satisfies the system $Az = r_k$, where $r_k = b - Ax_k$. In step k we now apply the above projection method to this system. Hence, we require that

$$z \in \mathcal{K}, \quad r_k - Az = b - A(x_k + z) \perp \mathcal{L}.$$

This gives the equations

$$r_k = b - Ax_k, \quad z = (V^T AU)^{-1}V^T r_k, \quad x_{k+1} = x_k + Uz. \quad (11.2.7)$$

for computing the new approximation x_{k+1} . A generic projection algorithm is obtained by starting from some x_0 (e.g., $x_0 = 0$), and repeatedly perform (11.2.7) for a sequence of subspaces \mathcal{L}_k and \mathcal{K}_k , $k = 1, 2, \dots$

11.2.2 The One-Dimensional Case

The simplest case of an iterative projection method is when the subspaces are one-dimensional, $\mathcal{L}_k = \text{span}(v_k)$, $\mathcal{K}_k = \text{span}(u_k)$, such that $v_k^T Au_k \neq 0$. Starting from some x_0 , in the k th step x_k is updated by

$$x_{k+1} = x_k + \alpha_k u_k, \quad \alpha_k = \frac{v_k^T r_k}{v_k^T Au_k}, \quad r_k = b - Ax_k. \quad (11.2.8)$$

By construction the new residual r_{k+1} is orthogonal to v_k . Note that r_{k+1} can be computed recursively from

$$r_{k+1} = r_k - \alpha_k Au_k. \quad (11.2.9)$$

This expression is obtained by multiplying $x_k = x_{k-1} + \alpha_{k-1} u_{k-1}$ by A and using the definition $r_k = b - Ax_k$. Since Au_{k-1} is needed for computing α_{k-1} using the recursive residual will save one matrix times vector multiplication.

⁶⁸In the rest of this chapter we will use vector notations and x_k will denote the k th approximation and not the k th component of x .

If A is s.p.d. and we take $v_k = u_k$, the above formulas become

$$r_k = b - Ax_k, \quad \alpha_k = \frac{u_k^T r_k}{u_k^T A u_k}, \quad x_{k+1} = x_k + \alpha_k u_k. \quad (11.2.10)$$

In this case x_{k+1} minimizes the quadratic functional

$$\phi(x) = \|x - x^*\|_A^2 = (x - x^*)^T A (x - x^*) \quad (11.2.11)$$

for all vectors of the form $x_k + \alpha_k u_k$. The vectors u_k are often called **search directions**. Expanding the function $\phi(x_k + \alpha u_k)$ with respect to α , we obtain

$$\phi(x_k + \alpha u_k) = \phi(x_k) - \alpha u_k^T (b - Ax_k) + \frac{1}{2} \alpha^2 u_k^T A u_k. \quad (11.2.12)$$

Taking $\alpha = \omega \alpha_k$ where α_k is given by (11.2.10), we obtain

$$\phi(x_k + \omega \alpha_k u_k) = \phi(x_k) - \rho(\omega) \frac{(u_k^T r_k)^2}{u_k^T A u_k}, \quad \rho(\omega) = \frac{1}{2} \omega(2 - \omega), \quad (11.2.13)$$

which is a quadratic function of ω . In a projection step ($\omega = 1$) the line $x_k + \alpha u_k$ is tangent to the ellipsoidal level surface $\phi(x) = \phi(x_{k+1})$, and $\phi(x_k + \alpha_k u_k) < \phi(x_k)$ provided that $u_k^T r_k \neq 0$. More generally, if $u_k^T r_k \neq 0$ we have from symmetry that

$$\phi(x_k + \omega \alpha_k u_k) < \phi(x_k), \quad 0 < \omega < 2.$$

For the error in $x_{k+1} = x_k + \omega \alpha_k u_k$ we have

$$\hat{x} - x_{k+1} = \hat{x} - x_k - \omega \frac{u_k^T r_k}{u_k^T A u_k} u_k = \left(I - \omega \frac{u_k u_k^T}{u_k^T A u_k} A \right) (\hat{x} - x_k).$$

This shows that the error in each step is transformed by a linear transformation $\hat{x} - x_{k+1} = B(\omega)(\hat{x} - x_k)$.

Example 11.2.2.

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

$$x_k := x_k - \hat{\alpha} e_i, \quad e_i^T (b - A(x_k - \hat{\alpha} e_i)) = 0,$$

where e_i is the i th unit vector. Hence the Gauss–Seidel method is equivalent to a sequence of one-dimensional modifications where the search directions are chosen equal to the unit vectors in cyclic order $e_1, \dots, e_n, e_1, \dots, e_n, \dots$

This interpretation can be used to prove convergence for the Gauss–Seidel (and, more generally, the SOR method) for the case when A is s.p.d..

Theorem 11.2.4.

If A is symmetric, positive definite then the SOR method converges for $0 < \omega < 2$, to the unique solution of $Ax = b$. In particular the Gauss–Seidel method, which corresponds to $\omega = 1$, converges.

Proof. In a minor step using search direction e_i the value of ϕ will decrease unless $e_i^T(b - Ax_k) = 0$, i.e., unless x_k satisfies the i th equation. A major step consists of a sequence of n minor steps using the search directions e_1, \dots, e_n . Since each minor step effects a linear transformation of the error $y_k = \hat{x} - x_k$, in a major step it holds that $y_{k+1} = By_k$, for some matrix B . Here $\|By_k\|_A < \|y_k\|_A$ unless y_k is unchanged in all minor steps, $i = 1, \dots, n$, which would imply that $y_k = 0$. Therefore if $y_k \neq 0$, then $\|By_k\|_A < \|y_k\|_A$, and thus $\|B\|_A < 1$. It follows that

$$\|B^n y_0\|_A \leq \|B\|_A^n \|y_0\|_A \rightarrow 0 \text{ when } n \rightarrow \infty,$$

i.e., the iteration converges.

If we define the minor step as $x := \omega \hat{a} e_i$, where ω is a fix relaxation factor, the convergence proof also holds. (We may even let ω vary with i , although the proof assumes that ω for the same i has the same value in all major steps.) This shows that the SOR method is convergent and by Theorem 11.1.4 this is equivalent to $\rho(B_\omega) < 1$, $0 < \omega < 2$. \square

We make two remarks about the convergence proof. First, it also holds if for the basis vectors $\{e_i\}_{i=1}^n$ we substitute an *arbitrary set of linearly independent vectors* $\{p_j\}_{j=1}^n$. Second, if A is a positive diagonal matrix, then we obtain the *exact* solution by the Gauss–Seidel method after n minor steps. Similarly, if A assumes diagonal form after a coordinate transformation with, $P = (p_1, \dots, p_n)$, i.e., if $P^T AP = D$, then the exact solution will be obtained in n steps using search directions p_1, \dots, p_n . Note that this condition is equivalent to the requirement that the vectors $\{p_j\}_{j=1}^n$ should be A -orthogonal, $p_i^T Ap_j = 0$, $i \neq j$.

Let A be symmetric, positive definite and consider the error functional (11.2.11). From the expansion (11.2.12) it is clear that the negative gradient of $\phi(x)$ with respect to x equals $-\nabla \phi(x) = b - Ax$. Hence the direction in which the function ϕ decreases most rapidly at the point x_k equals the residual $r_k = b - Ax_k$. The **method of steepest descent**⁶⁹ is a one-dimensional projection method with $v_k = u_k = r_k$. This leads to the iteration

$$r_k = b - Ax_k, \quad \alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}, \quad x_{k+1} = x_k + \alpha_k r_k. \quad (11.2.14)$$

It follows from (11.2.13) that when $r_k \neq 0$, it holds that $\phi(x_{k+1}) < \phi(x_k)$.

We now derive an expression for the rate of convergence of the steepest descent method. Denoting the error in x_k by $e_k = x_k - x^*$ we have

$$\begin{aligned} \|e_{k+1}\|_A^2 &= e_{k+1}^T A e_{k+1} = -r_{k+1}^T e_{k+1} = -r_{k+1}^T (e_k + \alpha_k r_k) \\ &= -(r_k - \alpha_k A r_k)^T e_k = e_k^T A e_k - \alpha_k r_k^T r_k, \end{aligned}$$

⁶⁹This method is attributed to Cauchy 1847.

where we have used that $r_{k+1}^T r_k = 0$. Using the expression (11.2.14) for α_k we obtain

$$\|e_{k+1}\|_A^2 = \|e_k\|_A^2 \left(1 - \frac{r_k^T r_k}{r_k^T A r_k} \frac{r_k^T r_k}{r_k^T A^{-1} r_k}\right). \quad (11.2.15)$$

To estimate the right hand side we need the following result.

Lemma 11.2.5 (Kantorovich⁷⁰ inequality).

Let A be a real symmetric matrix with eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$. Then for any vector x it holds

$$\frac{(x^T A x)(x^T A^{-1} x)}{(x^T x)^2} \leq \frac{1}{4} \left(\kappa^{1/2} + \kappa^{-1/2}\right)^2, \quad (11.2.16)$$

where $\kappa = \lambda_n/\lambda_1$ is the condition number of A .

Proof. (After D. Braess) Let $\mu = (\lambda_1 \lambda_n)^{1/2}$ be the geometric mean of the eigenvalues and consider the symmetric matrix $B = \mu^{-1}A + \mu A^{-1}$. The eigenvalues of B satisfy

$$\lambda_i(B) = \mu^{-1} \lambda_i + \mu \lambda_i^{-1} \leq \kappa^{1/2} + \kappa^{-1/2}, \quad i = 1 : n.$$

Hence, by the Courant maximum principle, for any vector x it holds

$$x^T B x = \mu^{-1}(x^T A x) + \mu(x^T A^{-1} x) \leq (\kappa^{1/2} + \kappa^{-1/2})(x^T x).$$

The left hand can be bounded using the simple inequality

$$(ab)^{1/2} \leq \frac{1}{2}(\mu^{-1}a + \mu b), a, b > 0.$$

Squaring this and taking $a = x^T A x$ and $b = x^T A^{-1} x$ the lemma follows. \square

From (11.2.15) and Kantorovich's inequality it follows for the method of steepest descent that

$$\|e_{k+1}\|_A^2 \leq \|e_k\|_A^2 \left(\frac{\kappa^{1/2} - \kappa^{-1/2}}{\kappa^{1/2} + \kappa^{-1/2}}\right)^2,$$

and hence

$$\|x - x_k\|_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x\|_A. \quad (11.2.17)$$

It can also be shown that asymptotically this bound is sharp. Hence, the asymptotic rate of convergence only depends on the extreme eigenvalues of A .

⁷⁰Leonid V. Kantorovich (1912–1986) Russian mathematician. From 1934 to 1960 he was a professor of mathematics at Leningrad University. 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.

If the matrix A is ill-conditioned the level curves of ϕ are very elongated hyperellipsoids. Then the successive iterates x_k , $k = 0, 1, 2, \dots$ will zig-zag slowly towards the minimum $x = A^{-1}b$ as illustrated in Figure 11.3.1 for a two dimensional case. Note that successive search directions are orthogonal.

Now consider the more general method where $u_0 = p_0 = r_0$ (i.e., the steepest descent direction) and the search directions $u_{k+1} = p_{k+1}$ are taken to be a linear combination of the negative gradient r_{k+1} and the previous search direction p_k , i.e.,

$$p_{k+1} = r_{k+1} + \beta_k p_k, \quad k = 0, 1, 2, \dots \quad (11.2.18)$$

Here the parameter β_k remains to be determined. (Note that $\beta_k = 0$ gives the method of steepest descent.) From (11.2.13) we know that to get $\phi(x_{k+1}) < \phi(x_k)$ we must have $p_k^T r_k \neq 0$. Replacing $(k+1)$ by k in (11.2.18) and multiplying by r_k^T , we obtain

$$r_k^T p_k = r_k^T r_k + \beta_{k-1} r_k^T p_{k-1} = r_k^T r_k, \quad (11.2.19)$$

since r_k is orthogonal to p_{k-1} . It follows that $r_k^T p_k = 0$ 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, regardless of the value of the parameter β_k , and $\phi(x_{k+1}) < \phi(x_k)$. From (11.2.19) we also obtain the alternative expression

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}. \quad (11.2.20)$$

We shall discuss the choice of the parameter β_k in the next section.

11.2.3 The Conjugate Gradient Method

We now consider a projection method where the subspaces \mathcal{K} and \mathcal{L} are chosen to be a sequence of **Krylov subspaces**, named after the Russian mathematician A. N. Krylov.⁷¹ These subspaces play a fundamental role in modern iterative methods both for linear systems and eigenvalue problems.

Definition 11.2.6.

Given a matrix $A \in \mathbf{R}^{n \times n}$ and a vector $u \in \mathbf{R}^n$ the Krylov subspace is

$$\mathcal{K}_k(A, u) = \text{span} \{u, Au, \dots, A^{k-1}u\}. \quad (11.2.21)$$

The related Krylov matrix is

$$K_k(A, u) = (u, Au, \dots, A^{k-1}u), \quad (11.2.22)$$

The sequence of Krylov subspaces for $k = 1 : n$ are nested, i.e.,

$$\mathcal{K}_k(A, u) \subseteq \mathcal{K}_{k+1}(A, u).$$

⁷¹Aleksei Nikolaevich Krylov (1863–1945) joined the department of ship construction at the Maritim Academy of St. Petersburg. In 1931 he found a new method for determining the frequency of vibrations in mechanical systems using these subspaces; see Faddeeva [196, Chap. III].

The dimension of $\mathcal{K}_k(A, b)$ will usually be k unless the vector b is specially related to A or $k > n$. If b is an eigenvector of A , then the dimension of $\mathcal{K}_k(A, b)$ is one, for all $k > 0$. If $p \leq n$ is the smallest integer such that

$$\mathcal{K}_{k+1}(A, b) = \mathcal{K}_k(A, b), \quad (11.2.23)$$

then we say that the Krylov sequence **terminates** for $k = p$. and it follows that

$$A\mathcal{K}_p(A, b) \subset \mathcal{K}_p(A, b).$$

Hence, $\mathcal{K}_p(A, b)$ is an invariant eigenspace of dimension p . Further, if $b \in \mathcal{K}_p(A, b)$, then the Krylov sequence terminates for $k = p$ and the vector b lies in an invariant eigenspace of A of dimension p . We say that the grade of b with respect to A equals p .

Krylov subspaces can be characterized in terms of matrix polynomials. Any vector $v \in \mathcal{K}_k(A, b)$ can be written in the form

$$v = \sum_{i=0}^{k-1} c_i A^i b = P_{k-1}(A)b,$$

where P_{k-1} is a polynomial of degree $k - 1$. This provides a link between polynomial approximation and Krylov type methods, the importance of which will become clear in the following.

The **conjugate gradient method** is a Krylov subspace projection method for solving a symmetric (Hermitian) positive definite linear system $Ax = b$. The method was published in 1952 by Hestenes⁷² and Stiefel⁷³. Consider the Hermitian, positive definite linear system $Ax = b$, $A \in \mathbf{C}^{n \times n}$. Let a sequence of approximate solutions be characterized by

$$x_k \in \mathcal{K}_k(A, b), \quad r_k = b - Ax_k \perp \mathcal{K}_k(A, b), \quad k = 1, 2, \dots \quad (11.2.24)$$

Set $x_0 = 0$, $r_0 = p_0 = b$, and use the recurrence (11.2.18) to generate p_{k+1} . A simple induction argument then shows that the vectors p_k and r_k both lie in the Krylov subspace $\mathcal{K}_k(A, b)$. In the CG method the parameter β_k is chosen to make p_{k+1} A -orthogonal or conjugate to the previous search direction, i.e.,

$$p_{k+1}^T A p_k = 0. \quad (11.2.25)$$

(A motivation to this choice is given in the second remark to Theorem 11.2.4.) Multiplying (11.2.18) by $p_k^T A$ and using (11.2.25) it follows that

$$\beta_k = -\frac{p_k^T A r_{k+1}}{p_k^T A p_k}. \quad (11.2.26)$$

⁷²Magnus Rudolph Hestenes (1905–1991) American mathematician got his PhD from University of Chicago. He was professor at UCLA, Los Angeles 1943–1974 and a vice president of the American Mathematical Society.

⁷³Eduard 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. 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 the Institute for Numerical Analysis (INA) at UCLA, Los Angeles.

We now prove the important result that this choice will in fact make p_{k+1} A -conjugate to *all* previous search directions!

Lemma 11.2.7.

In the CG method the residual vector r_k is orthogonal to all previous search directions and residual vectors

$$r_k^T p_j = 0, \quad j = 0, \dots, k-1, \quad (11.2.27)$$

and the search directions are mutually A -conjugate

$$p_k^T A p_j = 0, \quad j = 0 : k-1. \quad (11.2.28)$$

Proof. We first prove the relations (11.2.27) and (11.2.28) jointly by induction. Clearly r_k is orthogonal to the previous search direction p_{k-1} , and (11.2.25) shows that also (11.2.28) holds for $j = k-1$. Hence these relations are certainly true for $k = 1$.

Assume now that the statements are true for some $k \geq 1$. From $p_k^T r_{k+1} = 0$, changing the index, and taking the scalar product with p_j , $0 \leq j < k$ we get

$$r_{k+1}^T p_j = r_k^T p_j - \alpha_k p_k^T A p_j.$$

From the induction hypothesis this is zero, and since $r_{k+1}^T p_k = 0$ it follows that (11.2.27) holds for $k := k + 1$. Using equation (11.2.18), the induction hypothesis and equation (11.2.9) and then (11.2.18) again we find for $0 < j < k$

$$\begin{aligned} p_{k+1}^T A p_j &= r_{k+1}^T A p_j + \beta_k p_k^T A p_j = \alpha_j^{-1} r_{k+1}^T (r_j - r_{j+1}) \\ &= \alpha_j^{-1} r_{k+1}^T (p_j - \beta_{j-1} p_{j-1} - p_{j+1} + \beta_j p_j), \end{aligned}$$

which is zero by equation (11.2.27). For $j = 0$ we use $b = p_0$ in forming the last line of the equation. For $j = k$ we use (11.2.25), which yields (11.2.28). \square

The vectors p_0, \dots, p_{k-1} span the Krylov subspace $\mathcal{K}_k(A, b)$ and from (11.2.27) it follows that $r_k \perp \mathcal{K}_k(A, b)$. This relation shows that the conjugate gradient implements the projection method obtained by taking

$$\mathcal{K} = \mathcal{L} = \mathcal{K}_k(A, b).$$

Hence from Lemma 11.2.2 we have the following *global minimization property*.

Theorem 11.2.8.

The vector x_k in the CG method solves the minimization problem

$$\min_x \phi(x) = \|x - x^*\|_A^2, \quad x \in \mathcal{K}_k(A, b) \quad (11.2.29)$$

From this property it follows directly that the “energy” norm $\|x - x^*\|_A$ in the CG method is monotonically decreasing. It can be shown that also the error

norm $\|x - x_k\|_2$ is monotonically decreased (see Hestenes and Stiefel [321, Theorem 6.3]). On the other hand the residual norm $\|b - Ax_k\|_2$ normally oscillates and may increase.

Since the vectors r_0, \dots, r_{k-1} span the Krylov subspace $\mathcal{K}_k(A, b)$ the following orthogonality relations also hold:

$$r_k^T r_j = 0, \quad j = 0 : k-1. \quad (11.2.30)$$

Equation (11.2.30) ensures that in exact arithmetic the CG method will terminate after at most n steps for a linear system of order n . For suppose the contrary is true. Then $r_k \neq 0$, $k = 0 : n$ and by (11.2.30) these $n+1$ nonzero vectors in \mathbf{R}^n are mutually orthogonal and hence linearly independent, which is impossible.

From (11.2.30) it follows that *the residuals vectors r_0, r_1, \dots, r_k are the vectors that would be obtained from the sequence $A, bb, \dots, A^k b$ by Gram–Schmidt orthogonalization.* The vectors p_0, p_1, p_2, \dots , may be constructed similarly from the conjugacy relation (11.2.28).

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}^T and using the orthogonality (11.2.30) to get

$$r_{k+1}^T r_{k+1} = -\alpha_k r_{k+1}^T A p_k.$$

Equations (11.2.20) and (11.2.26) then yield

$$\beta_k = \|r_{k+1}\|_2^2 / \|r_k\|_2^2.$$

We observe that in this expression for β_k the matrix A is not needed. This property is important when the CG method is extended to non-quadratic functionals.

When a starting approximation $x_0 \neq 0$ is known then we can set $x = x_0 + z$, and apply the algorithm to the shifted system

$$Az = r_0, \quad r_0 = b - Ax_0.$$

We have seen that there are alternative, mathematically equivalent formulas, for computing r_k , α_k and β_k . These are not equivalent with respect to accuracy, storage and computational work. A comparison (see Reid [499]) tends to favor the original version. Using the inner product $(p, q) = p^T q$ then gives the MATLAB algorithm:

Algorithm 11.2. The CG Method.

```
function x = cgsolve(A,b,x0,tol);
% Solve Ax = b by the CG method.
% Iterate while ||Ax - b|| > tol*||b||.
%
x = x0; r = b - A*x
p = r; rtr = r'*r;
while (norm(r) > tol*norm(b))
    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

```

Note that if the argument A is sparse, then the matrix-vector product $A \cdot p$ will automatically be computed in MATLAB as a sparse operation. If a subroutine call $q = \text{amult}(p)$ is substituted for $q = A \cdot p$, the matrix A need not even be explicitly available.

Four vectors x, r, p and Ap need to be stored. Each iteration step requires one matrix by vector product, two vector inner products, and three vector updates. We remark that the computation of the inner products can be relatively expensive since they cannot simply be parallelized.

By instead taking the inner product in the above algorithm to be $(p, q) = p^T A q$ we obtain a related method that in each step minimizes the Euclidian norm of the residual over the same Krylov subspace. In this algorithm the vectors Ap_i , $i = 0, 1, \dots$ are orthogonal. In addition, the residual vectors are required to be A -orthogonal, i.e., conjugate. Consequently this method is called the **conjugate residual method**. This algorithm requires one more vector of storage and one more vector update than the CG method. Therefore, when applicable the CG method is usually preferred over the conjugate residual method.

In a Krylov subspace method the approximations are of the form

$$x_k - x_0 \in \mathcal{K}_k(A, b), \quad k = 1, 2, \dots,$$

With $r_k = b - Ax_k$ it follows that $r_k - b \in A\mathcal{K}_k(A, b)$. Hence the residual vectors can be written

$$r_k = q_k(A)b,$$

where $q_k \in \tilde{\Pi}_k^1$, the set of polynomials q_k of degree k with $q_k(0) = 1$. Since

$$\phi(x) = \frac{1}{2}\|x - x^*\|_A^2 = \frac{1}{2}r^T A^{-1}r = \frac{1}{2}\|r\|_{A^{-1}}^2,$$

the optimality property in Theorem 11.2.8 can alternatively be stated as

$$\|r_k\|_{A^{-1}}^2 = \min_{q_k \in \tilde{\Pi}_k^1} \|q_k(A)b\|_{A^{-1}}^2. \quad (11.2.31)$$

Denote by $\{\lambda_i, v_i\}$, $i = 1 : n$, the eigenvalues and eigenvectors of A . Since A is symmetric we can assume that the eigenvectors are orthonormal. Expanding the right hand side as

$$b = \sum_{i=1}^n \gamma_i v_i, \quad (11.2.32)$$

we have for any $q_k \in \tilde{\Pi}_k^1$

$$\|r_k\|_{A^{-1}}^2 \leq \|q_k(A)b\|_{A^{-1}}^2 = b^T q_k(A)^T A^{-1} q_k(A) b = \sum_{i=1}^n \gamma_i^2 \lambda_i^{-1} q_k(\lambda_i)^2.$$

In particular, taking

$$q_n(\lambda) = \left(1 - \frac{\lambda}{\lambda_1}\right) \left(1 - \frac{\lambda}{\lambda_2}\right) \cdots \left(1 - \frac{\lambda}{\lambda_n}\right), \quad (11.2.33)$$

we get $\|r_n\|_{A^{-1}} = 0$. This is an alternative proof that in exact arithmetic the CG method terminates after at most n steps.

If the eigenvalues of A are distinct then q_n in (11.2.33) is the minimal polynomial of A (see Section 10.1.2). If A only has p distinct eigenvalues then the minimal polynomial is of degree p and CG converges in at most p steps for any vector b . Hence, CG is particularly effective when A has low rank!

More generally, if the grade of b with respect to A equals p then only p steps are needed to obtain the exact solution. This will be the case if in the expansion (11.2.32) $\gamma_i \neq 0$ only for m different values of i .

11.2.4 Convergence Properties of CG

In the beginning the CG method was viewed primarily as a direct method. However, The finite termination property of the CG method shown above is only valid in exact arithmetic. In finite precision it could take $3n-5n$ steps before actual convergence occurred. For a long time it was not well understood how rounding errors influenced the method. As a consequence, the method did not come into wide use until much after it was discovered. The key insight needed, was to realize that it should be used as *iterative method* to obtain good approximate solution in far less than n iterations. In this context it is appropriate to talk about the convergence properties of the CG method.

Let x_k be the approximate solution obtained from the k th step of the CG method. Then x_k has the form $x_k = x_0 + p_{k-1}(A)(b - Ax_0)$, where p_{k-1} is a polynomial of degree $k-1$. Subtracting x from both sides and using $b = Ax$, it follows that

$$x - x_k = (I + Ap_{k-1}(A))(x - x_0) = q_k(A)(x - x_0),$$

where q_k is a polynomial of degree k such that $q_k(0) = 1$, i.e., a residual polynomial. Theorem 11.2.8 implies that $\|q_k(A)(x - x_0)\|_A$ is minimized over all residual polynomials of degree k . The following theorem is basic for the convergence analysis for analyzing the rate of convergence of the CG method.

Theorem 11.2.9.

Let the set S contain all the eigenvalues of the symmetric positive definite matrix A . Assume that for some residual polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ of degree k it holds that

$$\max_{\lambda \in S} |\tilde{q}_k(\lambda)| \leq M_k. \quad (11.2.34)$$

Let x_k be the k th iterate when the conjugate gradient is applied to the linear system $Ax = b$. Then an upper bound for the energy norm of the error in x_k is given by

$$\|x - x_k\|_A \leq M_k \|x - x_0\|_A. \quad (11.2.35)$$

Proof. Using the optimality property (11.2.32)

$$\|r_k\|_{A^{-1}}^2 \leq M_k^2 \sum_{i=1}^n \gamma_i^2 \lambda_i^{-1} = M_k^2 \|b\|_{A^{-1}}^2$$

□

We now select a set S on the basis of some assumption regarding the eigenvalue distribution of A and seek a polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ such that $M_k = \max_{\lambda \in S} |\tilde{q}_k(\lambda)|$ is small. A simple choice is to take $S = [\lambda_1, \lambda_n]$ and seek the polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ which 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 Section 11.1.6. It follows that

$$\|x - x_k\|_A < 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|x - x_0\|_A. \quad (11.2.36)$$

where $\kappa = \lambda_n(A)/\lambda_1(A)$. Note that the convergence of the conjugate residual method can be analyzed using a similar technique.

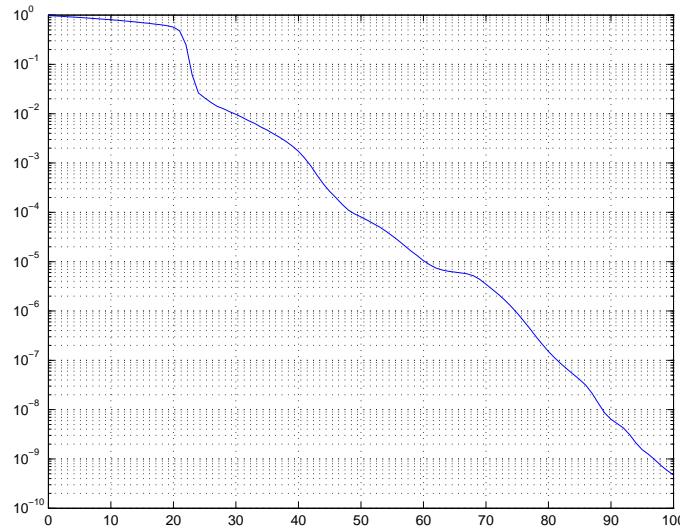


Figure 11.2.1. Convergence of the error norm for a 32 by 32 Laplace problem.

Example 11.2.3.

The model problem in Section 11.1.3 considered the Laplace equation

$$\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = 0$$

in the unit square $0 < x, y < 1$. Assume that Dirichlet boundary conditions are given so that the solution is $u(x, y) = 1$. If the Laplacian operator is approximated with the usual five-point operator with 32 mesh points in each direction (cf. Section sec10.1.1) this gives a linear system $Au = f$ of dimension $31^2 = 961$.

The extreme eigenvalues of $\frac{1}{4}A$ are $\lambda_{max} = 1 + \cos \pi h$, $\lambda_{min} = 1 - \cos \pi h$. It follows that

$$\kappa = \frac{1 + \cos \pi h}{1 - \cos \pi h} \approx \frac{1}{\sin^2 \pi h/2} \approx \frac{4}{(\pi h)^2}.$$

For $h = 1/32$ the number of iterations needed to reduce the initial error by a factor of 10^{-3} is then by (11.1.49) bounded by

$$k \approx \frac{1}{2} \sqrt{\kappa} \log(2 \cdot 10^3) \approx 77.$$

This is about the same number of iterations as needed with SOR using ω_{opt} to reduce the L_2 -norm by the same factor. However, the CG method is more general in that it does not require the matrix A to have “property A”.

Figure 11.2.2 shows the actual convergence of the error norm $\|x_k - x\|_2$ for the CG method when the initial approximation is taken to be identically zero. After an initial phase of slow convergence the error has been reduced to 10^{-6} after about 74 iteration.

The error estimate (11.2.36) tends to describe the convergence well for matrices with uniformly distributed eigenvalues. For more uneven distributions of eigenvalues it is pessimistic. As the iterations proceeds, the effect 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 (11.2.36) tends to be sharp asymptotically.

As has been shown in Axelsson [19] 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 error in 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 a number of iterations much less than n ,

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 in Theorem 11.2.4 will no longer be satisfied exactly. Indeed, orthogonality between residuals r_i and r_j , for $|i - j|$ is large, will usually be completely lost. Because of this, the finite termination property does not hold in practice.

The behavior of the conjugate gradient algorithm in finite precision is much more complex than in exact arithmetic. It has been observed that the bound (11.2.36) 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 the conjugate gradient algorithm in finite precision behaves like the exact algorithm applied to a larger linear system $\hat{A}\hat{x} = \hat{b}$, where the matrix \hat{A} has many eigenvalues distributed in tiny intervals about the eigenvalues of A . This means that $\kappa(\hat{A}) \approx \kappa(A)$, which explains why the bound (11.2.36) 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 true residual $\|b - Ax_k\|_2$ cannot be expected to approach zero.) This means that a termination criterion $\|r_k\|_2 \leq \epsilon$ will eventually always be satisfied even if $\epsilon \approx u$, where u is the machine precision.

11.2.5 The Lanczos Process

The **Lanczos⁷⁴ algorithm** (see [391]) is closely related to the CG method and was developed almost simultaneously. It can be viewed as an alternative algorithm for reducing a Hermitian matrix $A \in \mathbf{C}^{n \times n}$ by a unitary similarity to real symmetric tridiagonal form

$$U^H A U = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}.$$

In Section 10.5.1 we showed that, provided that all off-diagonal elements are nonzero, this decomposition is uniquely determined once $u_1 = Ue_1$ has been chosen. Equating the first $n - 1$ columns in

$$A(u_1, u_2, \dots, u_n) = (u_1, u_2, \dots, u_n)T_n$$

⁷⁴Cornelius Lanczos (1893–1974) Hungarian mathematician wrote his PhD thesis on relativity theory at the University of Budapest, where he also studied mathematics for Fèjer. 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 took up a permanent position at Boeing Aircraft Company. In 1949 he moved to the Institute of Numerical Analysis (INA) at UCLA, Los Angeles. In 1952 he took up the position as head of the Theoretical Physics Department at the Dublin Institute for Advanced Study in Ireland.

gives

$$Au_j = \beta_j u_{j-1} + \alpha_j u_j + \beta_{j+1} u_{j+1}, \quad j = 1 : n-1,$$

where we have put $\beta_1 u_0 \equiv 0$. The requirement that $u_{j+1} \perp u_j$ gives

$$\alpha_j = u_j^H (Au_j - \beta_j u_{j-1}),$$

(Note that since $u_j \perp u_{j-1}$ the last term could in theory be dropped. However, since in practice there will be a loss of orthogonality, it should be kept. This corresponds to using the modified rather than the classical Gram–Schmidt orthogonalization process.) Solving for u_{j+1} gives

$$\beta_{j+1} u_{j+1} = w_j, \quad w_j = Au_j - \alpha_j u_j - \beta_j u_{j-1}.$$

If $\beta_{j+1} = \|w_j\|_2 \neq 0$, then the next vector is given by

$$u_{j+1} = w_j / \beta_{j+1}.$$

Given the initial unit vector u_1 these equations can be used recursively to compute the elements in the tridiagonal matrix T and the orthogonal matrix U . The process terminates if $\beta_{j+1} = 0$. Then u_{j+1} can be chosen as an arbitrary unit vector orthogonal to u_k , $k \leq j$.

Algorithm 11.3. The Lanczos Process.

Given a Hermitian matrix $A \in \mathbb{C}^{n \times n}$ and a unit vector $u_1 \neq 0$. The Lanczos process generates after k steps a real symmetric tridiagonal matrix $T_k = \text{trid}(\beta_j, \alpha_j, \beta_{j+1})$ and a matrix $U_k = (u_1, \dots, u_k)$ with orthogonal columns spanning the Krylov subspace $\mathcal{K}_k(A, u_1)$:

```

function [U,alpha,beta] = lanczos(A,u1);
% LANCZOS computes a unitary matrix U
% and a symmetric tridiagonal matrix
% T_n with diagonals alpha_i, beta_(i+1)
n = length(u1);
U = zeros(n); U(:,1) = u1;
alpha = zeros(n,1); beta = zeros(n,1);
w = A*u1;
for j = 1:n-1
    alpha(j) = u1'*w;
    w = w - alpha(j)*u1;
    beta(j+1) = norm(w);
    if beta(j+1) == 0 return end
    u2 = w/beta(j+1);
    w = A*u2 - beta(j+1)*u1;
    U(:,j+1) = u2;
    u1 = u2;
end

```

Note that A only occurs in the matrix-vector operation Au_j . Hence, the matrix A need not be explicitly available, and can be represented by a subroutine which computes the matrix–vector product Aq for a given vector q . Only storage for T_k and three n -vectors u_{j-1} , u_j , and Au_j are needed.

Starting with a vector u_1 , the Lanczos process generates for $k = 1, 2, \dots$ a symmetric tridiagonal matrix

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & \ddots & \\ & & \ddots & \alpha_{k-1} & \beta_k \\ & & & \beta_k & \alpha_k \end{pmatrix}. \quad (11.2.37)$$

and a matrix $U_k = (u_1, \dots, u_k)$ with orthogonal columns spanning the Krylov subspace $\mathcal{K}_k(A, u_1)$. The generated quantities satisfy

$$AU_k = U_k T_k + \beta_{k+1} u_{k+1} e_k^T, \quad (11.2.38)$$

which is the **Lanczos decomposition**.

In the context of solving the linear system $Ax = b$, the appropriate choice of starting vector is given by

$$\beta_1 u_1 = b, \quad \beta_1 = \|b\|_2. \quad (11.2.39)$$

In the k th step of the CG method we determine an approximation $x_k \in \mathcal{K}_k(A, b)$ such that $r_k = b - Ax_k$ is orthogonal to $\mathcal{K}_k(A, b)$. Since U_k is an orthogonal basis in $\mathcal{K}_k(A, b)$, we set $x_k = U_k y_k$ and require that $U_k^T(b - Ax_k) = 0$. Using (11.2.38) we get

$$\begin{aligned} r_k &= b - Ax_k = \beta_1 u_1 - AU_k y_k \\ &= \beta_1 u_1 - U_k T_k y_k - \beta_{k+1} (e_k^T y_k) u_{k+1}. \end{aligned} \quad (11.2.40)$$

Multiplying this by U_k^T and using $U_k^T u_{k+1} = 0$ and $U_k^T u_1 = e_1$, gives

$$0 = U_k^T r_k = \beta_1 e_1 - T_k y_k.$$

It follows that $x_k = U_k y_k$, where y_k satisfies the tridiagonal linear system

$$T_k y_k = \beta_1 e_1. \quad (11.2.41)$$

If $\beta_{k+1} = \|r_k\|_2 = 0$, the Lanczos process terminates and u_{k+1} is not defined. In this case $AU_k = U_k T_k$, and using (11.2.40)

$$0 = r_k = \beta_1 u_1 - U_k T_k y_k = b - AU_k y_k = b - Ax_k.$$

Hence, $Ax_k = b$, that is, x_k is an exact solution.

The three-term recurrence

$$\beta_{j+1} u_{j+1} = Au_j - \alpha_j u_j - \beta_j u_{j-1},$$

is the core of the Lanczos process. This closely resembles the three-term recurrence satisfied by real orthogonal polynomials (see [132, Theorem 4.5.19]). Assume that the grade of u_1 is n . Let \mathcal{P}_{n-1} be the space of polynomials of degree at most $n-1$ equipped with the inner product

$$\langle p, q \rangle_{u_1} = (p(A)u_1)^T q(A)u_1. \quad (11.2.42)$$

The Lanczos process is just the Stieltjes algorithm for computing the corresponding sequence of orthogonal polynomials. 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 (11.2.42). The Lanczos recurrence computes the sequence of vectors $q_k(A)u_1$, where q_k is the characteristic polynomial of T_k .

There is a close connection between the CG method and the Lanczos process. In exact arithmetic $x_k = U_k y_k$, $k = 1, 2, \dots$, is the same sequence of approximations as generated by the CG method. Further, the columns of U_k equal the first k residual vectors in the CG method, normalized to unit length. One drawback with using the Lanczos process as outlined above. To compute $x_k = U_k y_k$ seems to require that all the vectors y_1, \dots, y_k are saved, whereas in the CG method a two-term recurrence relation is used to update x_k .

The recursion in the CG method is obtained from (11.2.41) by computing the Cholesky factorization of $T_k = R_k^T R_k$. Suppose the Lanczos process stops for $k = p \leq n$. Then, since A is a positive definite matrix the matrix $T_p = U_p^T A U_p$ is also positive definite. Thus T_k , $k \leq p$, which is a principal submatrix of T_p , is also positive definite and its Cholesky factorization must exist.

So far we have discussed the Lanczos process in exact arithmetic. In practice, roundoff will cause the generated vectors to lose orthogonality. A possible remedy is to reorthogonalize each generated vector u_{k+1} to all previous vectors u_k, \dots, u_1 . However, this is very costly both in terms of storage and operations. The effect of finite precision on the Lanczos method is the same as for the CG method; it slows down convergence, but does not prevent accurate approximations to be 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.

Assuming that $r_j \neq 0$, $j = 1 : k$ we introduce for the conjugate gradient method the following matrices:

$$R_k = (r_1 \ r_2 \ \dots \ r_k) S_k^{-1}, \quad P_k = (p_1 \ p_2 \ \dots \ p_k) S_k^{-1}, \quad (11.2.43)$$

$$L_k = \begin{pmatrix} 1 & & & \\ -\sqrt{\beta_1} & 1 & & \\ & -\sqrt{\beta_2} & 1 & \\ & & \ddots & \ddots \\ & & & -\sqrt{\beta_{k-1}} & 1 \end{pmatrix} \quad (11.2.44)$$

and the diagonal matrices

$$S_k = \text{diag} (\|r_1\|_2 \ \|r_2\|_2 \ \dots \ \|r_k\|_2), \quad (11.2.45)$$

$$D_k = \text{diag} (\alpha_1 \ \alpha_2 \ \dots \ \alpha_k). \quad (11.2.46)$$

Since the residual vectors r_j , $j = 1 : k$, are mutually orthogonal, R_n is an orthogonal matrix. Further, we have the relations

$$AP_nD_n = R_nL_n, \quad P_nL_n = R_n.$$

Eliminating P_n from the first relation we obtain

$$AR_n = R_n(L_nD_n^{-1}L_n) = R_nT_n,$$

Hence this provides an orthogonal similarity transformation of A to symmetric tridiagonal form T_n .

11.2.6 Symmetric Indefinite Systems

For symmetric positive definite matrices A the CG method computes iterates x_k that satisfy the minimization property

$$\min \|x - x_k\|_A, \quad x_k \in \mathcal{K}_k(A, b).$$

In case A is symmetric but indefinite $\|\cdot\|_A$ is no longer a norm. Hence the standard CG method may break down. This is also true for the conjugate residual method. On the other hand, the Lanczos process does not require A to be positive definite, so it can be used to develop Krylov subspace methods for symmetric indefinite systems.

Using the Lanczos basis U_k we seek approximations $x_k = U_ky_k \in \mathcal{K}_k(A, b)$, which are stationary values of $\|\hat{x} - x_k\|_A^2$. These are given by the Galerkin condition

$$U_k^T(b - AU_ky_k) = 0.$$

This leads again to the tridiagonal system (11.2.41). However, when A is indefinite, although the Lanczos process is still well defined, the Cholesky factorization of T_k may not exist. Moreover, it may happen that T_k is singular at certain steps, and then y_k is not defined.

If the Lanczos process stops for some $k \leq n$ then $AU_k = U_kT_k$. It follows that the eigenvalues of T_k are a subset of the eigenvalues of A , and thus if A is nonsingular so is T_k . Hence the problem with a singular T_k can only occur at intermediate steps. To solve the tridiagonal system (11.2.41) we compute the LQ factorization

$$T_k = \bar{L}_k Q_k, \quad Q_k^T Q_k = I, \tag{11.2.47}$$

where \bar{L}_k is lower triangular and Q_k orthogonal. Such a factorization always exists and can be computed by multiplying T_k with a sequence of plane rotations from the right

$$T_k G_{12} \cdots G_{k-1,k} = \bar{L}_k \begin{pmatrix} \gamma_1 & & & \\ \delta_2 & \gamma_2 & & \\ \epsilon_3 & \delta_3 & \gamma_3 & \\ \ddots & \ddots & \ddots & \\ & \epsilon_k & \delta_k & \bar{\gamma}_k \end{pmatrix}. \tag{11.2.48}$$

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 elements in $G_{k,k+1}$ are given by

$$\gamma_k = (\bar{\gamma}_k^2 + \beta_{k+1}^2)^{1/2}, \quad c_k = \bar{\gamma}_k / \gamma_k, \quad s_k = \beta_{k+1} / \gamma_k.$$

Since the solution y_k of $T_k y_k = \beta_1 e_1$ will change fully with each increase in k we write

$$x_k = U_k y_k = (U_k Q_k^T) \bar{z}_k = \bar{W}_k \bar{z}_k,$$

and let

$$\begin{aligned} \bar{W}_k &= (w_1, \dots, w_{k-1}, \bar{w}_k), \\ \bar{z}_k &= (\zeta_1, \dots, \zeta_{k-1}, \bar{\zeta}_k) = Q_k y_k. \end{aligned}$$

Here quantities without bars will be unchanged when k increases, and \bar{W}_k can be updated with \bar{T}_k . The system (11.2.41) 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 implementing the algorithm we should note that x_k^c need not be updated at each step, and that if $\bar{\gamma}_k = 0$, then \bar{z}_k is not defined. Instead we update

$$x_k^L = W_k z_k = x_{k-1}^L + \zeta_k w_k,$$

where L_k is used rather than \bar{L}_k . We can then always obtain x_{k+1}^c when needed from

$$x_{k+1}^c = x_k^L + \bar{\zeta}_{k+1} \bar{w}_{k+1}.$$

This defines the SYMMLQ algorithm. 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.

The **minimal residual algorithm** (MINRES) computes a vector in the Krylov subspace $\mathcal{K}_k(A, b)$ that minimizes the Euclidian norm of the residual, i.e., $k = 1, 2, \dots$ solves the least squares problem

$$\min \|b - Ax_k\|_2, \quad x_k \in \mathcal{K}_k(A, b) \quad (11.2.49)$$

The solution of (11.2.49) can be found using the Lanczos decomposition (11.2.37), which we rewrite as

$$AU_k = U_{k+1} \hat{T}_k, \quad \hat{T}_k = \begin{pmatrix} T_k \\ \beta_{k+1} e_k \end{pmatrix} \in \mathbf{R}^{(k+1) \times k}. \quad (11.2.50)$$

Setting $x_k = U_k y_k$ and using (11.2.50) we have

$$b - AU_k y_k = \beta_1 u_1 - U_{k+1} \hat{T}_k y_k = U_{k+1} (\beta_1 e_1 - \hat{T}_k y_k)$$

Using the orthogonality of U_{k+1} , the least squares problem (11.2.49) is seen to be equivalent to

$$\min_{y_k} \|\beta_1 e_1 - \hat{T}_k y_k\|_2.$$

This can be solved by computing the QR factorization $\hat{T}_k = \hat{Q}_k R_k$. As in (11.2.48) \hat{Q}_k can be chosen as a product of plane rotations. The rotations are also applied to e_1 giving $c_k = \hat{Q}^T e_1$. Finally y_k is obtained from the upper triangular banded system $R_k y_k = c_k$.

The minimal residual algorithm can be used also if A is singular and the system $Ax = b$ is not consistent. This method suffers more from poorly conditioned systems than SYMMLQ. However, it is of theoretical interest because of its relation to the harmonic Ritz values for eigenvalue approximations; see Section 11.6.3.

Review Questions

- 2.1** Let $\phi(x) = \frac{1}{2}x^T Ax - x^T b$, where A is symmetric positive definite, and consider the function $\varphi(\alpha) = \phi(x_k + \alpha p_k)$, where p_k is a search direction and x_k the current approximation to $x = A^{-1}b$. For what value $\alpha = \alpha_k$ is $\varphi(\alpha)$ minimized? Show that for $x_{k+1} = x_k + \alpha_k p_k$ it holds that $b - Ax_{k+1} \perp p_k$.
- 2.2** Show that minimizing the quadratic form $\frac{1}{2}x^T Ax - x^T b$ along the search directions $p_i = e_i$, $i = 1 : n$ is equivalent to one step of the Gauss–Seidel method.
- 2.3** How are the search directions chosen in the method of steepest descent? What is the asymptotic rate of convergence of this method?
- 2.4** Define the Krylov space $\mathcal{K}_j(A, b)$. Show that it is invariant under (i) scaling τA . (ii) translation $A - sI$. How is it affected by an orthogonal similarity transformation $\Lambda = V^T AV$, $c = V^T b$?
- 2.5** What minimization problems are solved by the conjugate gradient method? How can this property be used to derive an upper bound for the rate of convergence of the CG method.
- 2.6** Let the symmetric matrix A have eigenvalues λ_i and orthonormal eigenvectors v_i , $i = 1 : n$. If only $d < n$ eigenvalues are distinct, what is the maximum dimension of the Krylov space $\mathcal{K}_j(A, b)$?

Problems and Computer Exercises

- 2.1** Let λ_i, v_i be an eigenvalue and eigenvector of the symmetric matrix A .
 - (a) Show that if $v_i \perp b$, then also $v_i \perp \mathcal{K}_j(A, b)$, for all $j > 1$.
 - (b) Show that if b is orthogonal against p eigenvectors, then the maximum dimension of $\mathcal{K}_j(A, b)$ is at most $n - p$. Deduce that the the CG method converges in at most $n - p$ iterations.

- 2.2** Let $A = I + BB^T \in \mathbf{R}^{n \times n}$, where B is of rank p . In exact arithmetic, how many iterations are at most needed to solve a system $Ax = b$ with the CG method?
- 2.3** Write down explicitly the conjugate residual method. Show that in this algorithm one needs to store the vectors x, r, Ar, p and Ap .
- 2.4** SYMMLQ is based on solving the tridiagonal system (11.2.38) using an LQ factorization of T_k . Derive an alternative algorithm, which solves this system with Gaussian elimination with partial pivoting.
- 2.5** Suppose that we have obtained two approximative solutions x_1 and x_2 to the linear system $Ax = b$.
- (a) Show that the linear combination $y = \alpha x_1 + (1 - \alpha)x_2$ which minimizes $\|b - Ay\|_2$ is obtained by choosing

$$\alpha = \frac{r_2^T(r_1 - r_2)}{\|r_1 - r_2\|_2^2}.$$

- (b) Show that if $0 < \alpha < 1$, then the residual norm of y is strictly smaller than $\min\{\|r_1\|_2, \|r_2\|_2\}$.

11.3 Least Squares Problems

11.3.1 Introduction

Many applications lead to least squares problems

$$\min_x \|Ax - b\|_2, \quad A \in \mathbf{R}^{m \times n}, \quad (11.3.1)$$

where A is large and sparse. Such problems are often solved by iterative methods. Unless otherwise stated, we assume in the following that A has full column rank, so that the problem has a unique solution.

One way to proceed would be to form the positive definite system of normal equations

$$A^T Ax = A^T b.$$

Then any iterative method for symmetric positive definite linear systems can be used. The computation of $A^T A$ is costly and is equivalent to n iterations for the original system. However, it is easy to see that the explicit formation of the matrix $A^T A$ can usually be avoided by using the **factored form** of the normal equations

$$A^T(Ax - b) = 0 \quad (11.3.2)$$

of the normal equations. Note that this approach can also be used for an unsymmetric linear system.

Working only with A and A^T separately has two important advantages. First, as has been much emphasized for direct methods, a small perturbation in $A^T A$, e.g., by roundoff, may change the solution much more than perturbations of similar size

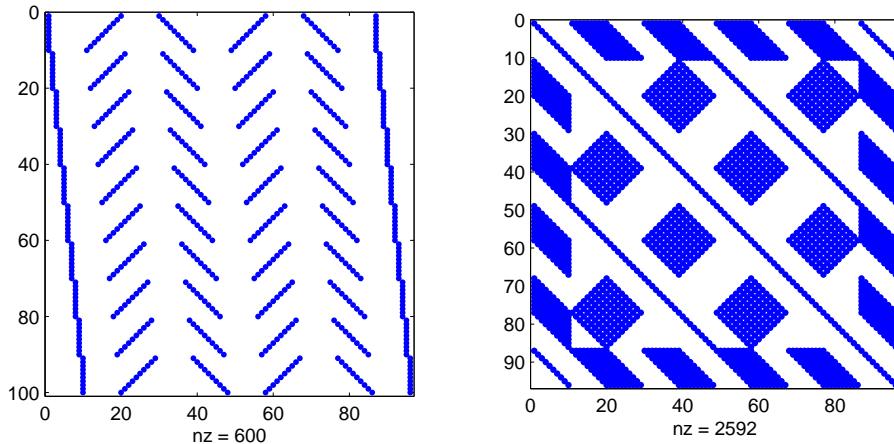


Figure 11.3.1. Structure of A (left) and $A^T A$ (right) for a model problem.

in A itself. Second, we avoid the fill which can occur in the formation of $A^T A$. The matrix $A^T A$ often has many more nonzero elements than A and can be quite dense even when A is sparse.

Iterative methods can be used also for computing a minimum norm solution of a (consistent) underdetermined system,

$$\min \|y\|_2, \quad A^T y = c.$$

If A^T has full row rank the unique solution satisfies the normal equations of the second kind

$$y = Az, \quad A^T(Az) = c. \quad (11.3.3)$$

Again the explicit formation of the cross-product matrix $A^T A$ should be avoided. Another possible approach is to use an iterative method applied to the augmented system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}; \quad (11.3.4)$$

see Theorem 1.1.5. This also avoids forming the normal equations, but has the drawback that, since the augmented system is symmetric indefinite, many standard iterative methods cannot be applied.

If A is square and nonsingular, then the normal equations are symmetric and positive definite with solution $x = A^{-1}b$. Hence a natural extension of iterative methods for symmetric positive definite systems to general nonsingular, non-symmetric linear systems $Ax = b$ is to apply them to the normal equations of first or second kind. A severe drawback can be the *squaring of the condition number*

$$\kappa(A^T A) = \kappa(AA^T) = \kappa^2(A),$$

of system $Ax = b$. From the estimate (11.2.36) we note that this can lead to a substantial decrease in the rate of convergence.

For some classes of sparse least squares problems fill-in will make sparse direct methods prohibitively costly in terms of storage and operations. There are problems where A is large and sparse but $A^T A$ is almost dense; compare Figure 11.3.1. Then the Cholesky factor R will in general also be dense. This rules out the use of sparse direct methods based on QR decomposition. 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 equals 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.

11.3.2 Some Classical Methods

The non-stationary Richardson iteration applied to the normal equations $A^T A x = A^T b$ can be written in the factored form

$$x_{k+1} = x_k + \omega_k A^T (b - Ax_k), \quad k = 1, 2, \dots, \quad (11.3.5)$$

where $\omega_k > 0$ are acceleration parameters. This method is often referred to as **Landweber's method** [393]. An important thing to notice in the implementation is that the matrix $A^T A$ should be kept in factored form and *not explicitly computed*. Only matrix-vector products with A and A^T are required in (11.3.5). As remarked, this avoids numerical instability and fill-in. Landweber's method is often used for solving system originating from discretized ill-posed problems; see Section 11.3.5.

In the stationary case $\omega_k = \omega$, and the iteration matrix of (11.3.5) equals $G = I - \omega A^T A$. If the singular values of A are σ_i , $i = 1 : n$, then the eigenvalues of G are

$$\lambda_k(G) = 1 - \omega \sigma_i^2, \quad i = 1 : n.$$

Using Theorem 11.1.4 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 that $\omega < 1/\sigma_1^2$. Conditions for convergence in the nonstationary case are given in the following theorem, which follows from a more general convergence result by Albaroni and Censor [5].

Theorem 11.3.1. *The iterates of (11.3.5) converge to a solution \hat{x} to $\min_x \|Ax - b\|_2$ for all vectors b if and only if for some $\epsilon > 0$ it holds*

$$0 < \epsilon < \omega_k < (2 - \epsilon)/\sigma_1^2, \quad \forall k,$$

where σ_1 is the largest singular value of A . If in addition $x^{(0)} \in \mathcal{R}(A^T)$ then \hat{x} is the unique minimum norm solution.

If we take $x_0 = 0$, then it follows from (11.3.5) that the sequence of approximations satisfy $x_k \in \mathcal{R}(A^T)$. Hence, when A is rank deficient, Landweber's method will converge to the pseudoinverse solution x^\dagger . The assumption that $\text{rank}(A) = n$ is unnecessary; zero singular values do not matter. Of course, this is only strictly true in exact computation, but rounding errors will just result in a small error component in $\mathcal{N}(A)$ that growth linearly with k . This is true also for most other iterative methods for solving the normal equations.

A method related to Landweber's method was introduced by Cimmino [111, 1932].⁷⁵ Let $Ax = b$, be a linear system where $A \in \mathbf{R}^{m \times n}$ and $a_i^T = e_i^T A$, $i = 1 : m$, be the rows of A . A solution must lie on the m hyperplanes

$$a_i^T x = b_i, \quad i = 1 : m. \quad (11.3.6)$$

In Cimmino's method one computes the m points

$$x_i^{(0)} = x^{(0)} + 2 \frac{(b_i - a_i^T x^{(0)})}{\|a_i\|_2^2} a_i, \quad i = 1 : m, \quad (11.3.7)$$

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.,

$$x^{(1)} = \frac{1}{\mu} \sum_{i=1}^m \mu_i x_i^{(0)}, \quad \mu = \sum_{i=1}^m \mu_i. \quad (11.3.8)$$

This has a nice geometrical interpretation. The points $x_i^{(0)}$ are the orthogonal reflections of $x^{(0)}$ with respect to the hyperplanes (11.3.6). Cimmino noted that the initial 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 hypersphere is the unique solution of the linear system $Ax = b$. Subtracting x from both sides of (11.3.7) and using $b_i = a_i^T x$, we find that

$$x_i^{(0)} - x = P_i(x^{(0)} - x), \quad P_i = I - 2 \frac{a_i a_i^T}{\|a_i\|_2^2} \quad (11.3.9)$$

⁷⁵Gianfranco 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 in the literature for his early paper on solving linear systems; see Benzi [47].

where P_i , $i = 1 : n$, 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 this hypersphere,

$$\|x^{(1)} - x\|_2 < \|x^{(0)} - x\|_2,$$

that is, the error is reduced in each step. It follows that Cimmino's method is convergent.

In matrix form Cimmino's method can be written as

$$x^{(k+1)} = x^{(k)} + \frac{2}{\mu} A^T D^T D(b - Ax^{(k)}), \quad (11.3.10)$$

where

$$D = \text{diag}(d_1, \dots, d_m), \quad d_i = \frac{\sqrt{\mu_i}}{\|a_i\|_2}. \quad (11.3.11)$$

If $A \in \mathbf{R}^{m \times n}$ 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 (11.3.10) is just Landweber's method (11.3.5) with $\omega = 2/\mu$. If A is rank deficient and the initial approximation is taken to be $x^{(0)} = 0$, then the method converges to the minimum norm solution.

We now look at applying The methods of Jacobi and Gauss-Seidel to normal equations of the first or second kind. First assume that all columns in $A = (a_1, \dots, a_n) \in \mathbf{R}^{m \times n}$ are nonzero. In the j th minor step of Jacobi's method for $A^T(Ax - b) = 0$ the approximation $x_j^{(k+1)}$ is determined so that the j th equation in the normal system $a_j^T(Ax - b) = 0$ is satisfied, i.e.,

$$x_j^{(k+1)} = x_j^{(k)} + a_j^T(b - Ax^{(k)})/\|a_j\|_2^2, \quad j = 1 : n. \quad (11.3.12)$$

If (11.3.12) is used with the starting approximation $x^{(0)} = 0$, then all iterates will lie in $\mathcal{R}(A^T)$. Hence, if $\lim_{k \rightarrow \infty} x^{(k)} = x$, then x is a minimum norm solution of the least squares problem.

Jacobi's method can be written in matrix form as

$$x^{(k+1)} = x^{(k)} + D_A^{-1} A^T(b - Ax^{(k)}), \quad (11.3.13)$$

where

$$D_A = \text{diag}(d_1, \dots, d_n) = \text{diag}(A^T A).$$

Note that Jacobi's method is symmetrizable, since

$$D_A^{1/2}(I - D_A^{-1} A^T A)D_A^{-1/2} = I - D_A^{-1/2} A^T A D_A^{-1/2}.$$

The Gauss-Seidel method for $A^T A x = A^T b$ is a special case of the following class of **residual reducing** projection methods studied by Householder and

Bauer [342, 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

$$x^{(j+1)} = x^{(j)} + \alpha_j p_j, \quad \alpha_j = p_j^T A^T (b - Ax^{(j)}) / \|Ap_j\|_2^2. \quad (11.3.14)$$

It is easily verified that $r^{(j+1)} \perp Ap_j = 0$, where $r_j = b - Ax^{(j)}$, and by Pythagoras' theorem

$$\|r^{(j+1)}\|_2^2 = \|r^{(j)}\|_2^2 - |\alpha_j|^2 \|Ap_j\|_2^2 \leq \|r^{(j)}\|_2^2.$$

This shows that this class of methods (11.3.14) is residual reducing. For a square matrix A method (11.3.15) was developed by de la Garza [142, 1951].

If A has linearly independent columns we obtain the Gauss–Seidel method for the normal equations by taking p_j in (11.3.14) equal to the unit vectors e_j in cyclic order e_1, e_2, \dots, e_n . Then if $A = (a_1, a_2, \dots, a_n)$, we have $Ap_j = Ae_j = a_j$. An iteration step in the Gauss–Seidel method consists of n minor steps where $z^{(1)} = x^{(k)}$,

$$z^{(j+1)} = z^{(j)} + e_j a_j^T (b - Az^{(j)}) / \|a_j\|_2^2, \quad j = 1 : n, \quad (11.3.15)$$

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. With $r^{(1)} = b - Ax^{(k)}$, we obtain the simple recurrence

$$z^{(j+1)} = z^{(j)} + \delta_j e_j, \quad r^{(j+1)} = r^{(j)} - \delta_j a_j, \quad j = 1 : n. \quad (11.3.16)$$

$$\delta_j = a_j^T r^{(j)} / \|a_j\|_2^2, \quad (11.3.17)$$

$$(11.3.18)$$

Note that in the j th minor step only the j th column of A is accessed. and that it can be implemented without forming the matrix $A^T A$ explicitly. In contrast to the Jacobi method the Gauss–Seidel method is not symmetrizable and the ordering of the columns of A will influence the convergence.

The SOR method for the normal equations $A^T(Ax - b) = 0$ is obtained by introducing a relaxation parameter ω in the Gauss–Seidel method (11.3.16). The method is given by (11.3.16) with

$$\delta_j = \omega a_j^T r^{(j)} / d_j, \quad (11.3.19)$$

The SOR method always converges when $A^T A$ is positive definite and ω satisfies $0 < \omega < 2$. The SOR method shares with the Gauss–Seidel method the advantage of simplicity and small storage requirements. However, the rate of convergence may be slow for any choice of ω . Then the SSOR method is to be preferred. This is easily implemented by following each forward sweep of SOR with a backward sweep $j = n : -1 : 1$. A disadvantage with the Gauss–Seidel and SOR methods is that they are less easily adapted for parallel computation than Jacobi's method, which just requires a matrix–vector multiplication.

For a consistent system $A^T y = c$ the minimum norm solution satisfies the normal equations of second type

$$A^T(Az) = c, \quad y = Az. \quad (11.3.20)$$

In the j th minor step of Jacobi's method applied to (11.3.20) the j th component of the approximation is modified so that the j th equation $a_j^T A z = c_j$ is satisfied, i.e., we take

$$z_j^{(k+1)} = z_j^{(k)} + (c_j - a_j^T (A z^{(k)})) / \|a_j\|_2^2, \quad j = 1 : n. \quad (11.3.21)$$

Multiplying by A and setting $y^{(k)} = A z^{(k)}$, the iteration can be written in matrix form

$$y^{(k+1)} = y^{(k)} + A D_A^{-1} (c - A^T y^{(k)}), \quad D_A = \text{diag}(A^T A). \quad (11.3.22)$$

The Gauss–Seidel method for solving the normal equations of second kind can also be implemented without forming $A^T A$. It is a special case of a family of **error reducing** methods defined as follows: Let $p_i \notin \mathcal{N}(A)$, $i = 1, 2, \dots$, be a sequence of nonzero n -vectors and compute approximations of the form

$$y^{(j+1)} = y^{(j)} + \omega_j A p_j, \quad \omega_j = p_j^T (c - A^T y^{(j)}) / \|A p_j\|_2^2. \quad (11.3.23)$$

If the system $A^T y = c$ is consistent there is a unique solution y of minimum norm. If we denote the error by $e^{(j)} = y - y^{(j)}$, then by construction $e^{(j+1)} \perp A p_i$. By Pythagoras' theorem it follows that

$$\|e^{(j+1)}\|_2^2 = \|e^{(j)}\|_2^2 - |\omega_j|^2 \|A p_j\|_2^2 \leq \|e^{(j)}\|_2^2,$$

i.e. this class of methods is error reducing.

We obtain the Gauss–Seidel method by taking p_j to be the unit vectors e_j in cyclic order e_1, e_2, \dots, e_n . Then $A p_j = a_j$, where a_j is the j th column of A . The iterative method (11.3.23) takes the form

$$y^{(j+1)} = y^{(j)} + a_j (c_j - a_j^T y^{(j)}) / \|a_j\|_2^2, \quad j = 1 : n. \quad (11.3.24)$$

This shows that, if we take $x^{(0)} = A y^{(0)}$, then for an arbitrary $y^{(0)}$ (11.3.24) is equivalent to the Gauss–Seidel method for (11.3.20). For the case of a square matrix A this method was originally devised by Kaczmarz [357, 1937].⁷⁶

The SOR method applied to the normal equations of the second kind can be obtained by introducing an acceleration parameter ω in (11.3.24), giving

$$y^{(j+1)} = y^{(j)} + \omega a_j (c_j - a_j^T y^{(j)}) / \|a_j\|_2^2, \quad j = 1 : n. \quad (11.3.25)$$

The SSOR method is obtained by performing a backward sweep $j = n : -1 : 1$ after each forward sweep in (11.3.25).

Block versions of the Jacobi and Gauss–Seidel methods can be devised. Assume that A has linearly independent columns and is partitioned in blocks as

$$A = (A_1, A_2, \dots, A_s), \quad A_j \in \mathbf{R}^{m \times n_j}.$$

⁷⁶Stefan Kaczmarz, Polish mathematician and professor at the Technical University of Lwów. Kaczmarz was a close collaborator of Stefan Banach and Hugo Steinhaus. He was killed in action in September 1939, when German troops invaded Poland.

The corresponding block version of Jacobi's method for the normal equations $A^T(Ax - b) = 0$ is

$$x_j^{(k+1)} = x_j^{(k)} + A_j^\dagger(b - Ax^{(k)}), \quad j = 1 : s, \quad (11.3.26)$$

where $A_j^\dagger = (A_j^T A_j)^{-1} A_j^T$. If the QR factorization of the blocks

$$A_j = Q_j R_j, \quad Q_j \in \mathbf{R}^{m \times n_j}$$

are available then $A_j^\dagger = R_j^{-1} Q_j^T$ can be used in (11.3.26).

For the minimum norm solution of consistent system $A^T y = b$ we solve $A^T A z = c$ and take $y = A^T z$. In the j th minor step of the block Jacobi method the j th block component of the approximation to z is modified so that the j th block equation $A_j^T A z = c_j$ is satisfied. After multiplication by A and eliminating $z^{(k)}$ this gives the iteration

$$y^{(k+1)} = y^{(k)} + (A_j^\dagger)^T(c_j - A_j^T y^{(k)}), \quad j = 1 : s, \quad (11.3.27)$$

where $(A_j^\dagger)^T = A_j (A_j^T A_j)^{-1} = Q_j R_j^{-T}$.

Block Gauss-Seidel and block SOR and SSOR methods can be derived for both kinds of normal equations. Convergence properties of these and other block-iterative methods have been studied by Elfving [190].

11.3.3 Krylov Subspace Methods

We recall that the least squares problem

$$\min_x \|Ax - b\|_2, \quad A \in \mathbf{C}^{m \times n}, \quad (11.3.28)$$

$\text{rank}(A) = n$, has a unique solution x , which satisfies the Hermitian positive definite system of normal equations

$$A^H A x = A^H b.$$

A Krylov subspace method for solving (11.3.28) can be derived by applying the CG method from Section 11.2.3 to this system. However, the algorithm obtained by a straightforward substitution of $A^H A$ and $A^H b$ for A and b in Algorithm 11.2.3 needs to be slightly modified. First, as remarked in Section 11.3 the matrix $A^H A$ should be kept in product form and not be formed explicitly. Further, better stability is obtained if the residuals $r_k = b - Ax_k$ are recurred instead of the residuals $A^H r_k$ of the normal equations. The resulting Krylov subspace algorithm for solving the normal equations is called CGLS.⁷⁷

⁷⁷This algorithm has been given many different names in the literature. Saad [518] calls it CGNR, Axelsson [19] GCG-LS and Hanke [305] CGNE.

Algorithm 11.4. CGLS.

```

function [x,r] = cglsl(A,b,x0,eps);
% Solve the normal equation A^H Ax = A^H b by the CG
% method. Iterate until ||b - Ax|| < eps.
x = x0; r = b - A*x;
s = A'*r; p = s;
sts = s'*s;
while (norm(r) > eps)
    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 iterates in CGLS lie in the shifted Krylov subspace,

$$x_k \in x_0 + \mathcal{K}_k(A^H A, A^H r_0). \quad (11.3.29)$$

By the minimization property of the CG method it follows that x_k minimizes the norm of the residual error

$$\|A(x - x_k)\|_2^2 = \|r - r_k\|_2^2 = \|r_k\|_2^2 - \|r\|_2^2.$$

in this Krylov subspace. The last equality follows from the relation $r \perp r - r_k$ and Pythagoras' theorem.

The convergence properties of CGLS can be deduced from those of the CG method; see Section 11.2.4. It follows that

$$\|r - r_k\|_2^2 < 2 \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|r_0\|_2, \quad \kappa = \kappa(A).$$

This variational property implies that $\|r - r_k\|_2$ and also $\|x - x_k\|_2$ will decrease monotonically. This is not only true in exact arithmetic but also in practice. However, $\|A^T r_k\|_2$ will often exhibit large oscillations when $\kappa(A)$ is large. We stress that *this behavior is not a result of rounding errors*.

The convergence of the CGLS also depends on the distribution of the singular values of A and on the right hand side b . In particular, if A has only $t \leq n$ distinct singular values, then in exact arithmetic the solution is obtained in at most t steps. The three phases of convergence described for the CG method occur also for CGLS.

If we take $x_0 = 0$, then it follows from (11.3.29) that the sequence of approximations satisfy $x_k \in \mathcal{R}(A^H)$. Hence, the assumption that $\text{rank}(A) = n$ is

unnecessary, and when A is rank deficient, CGLS will converge to the pseudoinverse solution x^\dagger .

Assume now that A has full row rank. Then the matrix $AA^H y$ is positive definite and the minimum norm solution of $Ax = b$ satisfies the normal equations of the second kind

$$AA^H y = b, \quad x = A^H y.$$

A CG method for this system is obtained by substituting AA^H for A in Algorithm 11.2.3. If y_k is eliminated by setting $A^H y_k = x_k$, the CGME algorithm is obtained. This is also known as **Craig's algorithm** [123].⁷⁸

Algorithm 11.5. CGME.

```

function [x,r] = cgme(A,b,x0,eps);
% Solve x = A^Ty, AA^Ty = b by the CG method
% Iterate until ||b - Ax|| < eps||b||
del = eps*norm(b);
x = x0; r = b - A*x;
p = r; rtr = r'*r;
while (norm(r) > del)
    q = A'*p;
    alpha = rtr/(q'*q);
    x = x + alpha*q;
    r = r - alpha*(A*q);
    rtrold = rtr;
    rtr = r'*r;
    beta = rtr/rtrold;
    p = r + beta*p;
end

```

As for CGLS, the iterates x_k in CGME lie in the shifted Krylov subspace,

$$x_k \in x_0 + \mathcal{K}_k(A^H A, A^H r_0).$$

and minimize the error norm

$$\|A^H(y - y_k)\|_2^2 = \|x - x_k\|_2^2.$$

Hence, CGME is an error reducing method, but $\|r - r_k\|_2$ will in general not decrease monotonically. An upper bound for the rate of convergence is given by

$$\|x - x_k\|_2^2 < 2 \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|x - x_0\|_2^2.$$

For consistent problems both CGLS and CGME can be applied. Craig's method is attractive in that it minimizes at each step the error norm $\|x - x_k\|_2$.

⁷⁸Also this algorithm has been given many different names. Saad [518] calls it CGNE, We follow Hanke [305] and call it CGME for minimal error.

However, in contrast to CGLS, the residual norm $\|r_k\|_2$ will not decrease monotonically and can fluctuate wildly for ill-conditioned problems. Since many stopping criteria rely on the residual norm this behavior is a disadvantage.

For both methods the squaring of the condition number may lead to very slow convergence. However, there are special cases when both CGLS and CGME converge much faster than alternative methods. For example, when A is orthogonal, $A^H A = AA^H = I$, both methods converge in one step.

11.3.4 Lanczos Bidiagonalization and LSQR

We have seen that the CG method can be formulated in terms of the Lanczos process. This was derived from the transformation to tridiagonal form using a sequence of orthogonal similarity transformations. For the least squares problem this role is played by the bidiagonalization of a general rectangular matrix using Householder transformations given in Section 8.4.5. There it was shown that for a matrix $A \in \mathbf{R}^{m \times n}$ the decomposition

$$U^T A V = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad B = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \ddots & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix} \in \mathbf{R}^{(n+1) \times n}, \quad (11.3.30)$$

to lower bidiagonal form is essentially unique. This result can be used to derive a Lanczos bidiagonalization algorithm, due to Golub and Kahan [265].

$$U = (u_1, u_2, \dots, u_m), \quad V = (v_1, v_2, \dots, v_n) \quad (m > n).$$

Using the orthogonality of U and V in (11.3.30) we get

$$AV = U \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad A^T U = V (B^T \ 0).$$

Equating the j th column in these two matrix equations gives ($v_0 = 0$)

$$Av_j = \alpha_j u_j + \beta_{j+1} u_{j+1}, \quad (11.3.31)$$

$$A^T u_j = \beta_j v_{j-1} + \alpha_j v_j, \quad (11.3.32)$$

where $u_1 \in \mathbf{R}^m$, $\|u_1\|_2 = 1$ is a unit starting vector. These relations can be used to generate the vectors $v_1, u_2, v_2, u_3, \dots, v_n, u_{n+1}$ together with the elements in the columns of B . We obtain the recursion:

Set $v_0 = 0$ and for $j = 1 : n$

$$\alpha_j v_j = A^T u_j - \beta_j v_{j-1}, \quad (11.3.33)$$

$$\beta_{j+1} u_{j+1} = Av_j - \alpha_j u_j. \quad (11.3.34)$$

The elements α_j and β_{j+1} in B are determined by the condition that $\|u_j\|_2 = \|u_{j+1}\|_2 = 1$. Since the bidiagonal decomposition is unique Lanczos bidiagonalization generates in exact computation the same decomposition as the Householder algorithm.

The recurrence relations (11.3.32) can be rewritten in matrix form as

$$AV_k = U_{k+1}B_k, \quad (11.3.35)$$

$$A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \quad (11.3.36)$$

which are the two Lanczos decompositions associated with the bidiagonalization process. These equations will continue to hold to within machine precision even when there is a loss of orthogonality in U_{k+1} and V_k .

In (11.3.33)–(11.3.34) only matrix–vector multiplications with A and A^T are used and the matrix A is not transformed. This is an important advantage when the matrix A is structured or sparse. A drawback compared to the Householder algorithm in Section 8.4.5 is that there will be a gradual loss of orthogonality in the vectors u_k and v_k . It is easily shown by induction that the vectors generated by these recurrence relations satisfy $u_k \in \mathcal{K}_k(AA^T, u_1)$, $v_k \in \mathcal{K}_k(A^TA, A^T u_1)$ and hence,

$$\mathcal{R}(U_k) = \mathcal{K}_k(AA^T, u_1), \quad \mathcal{R}(V_k) = \mathcal{K}_k(A^TA, A^T u_1), \quad k = 1 : n. \quad (11.3.37)$$

Note the relation

$$K_{k+1}(AA^T, u_1) = AK_k(A^TA, u_1).$$

The bidiagonalization algorithm can be used for developing methods related to CGLS and CGME. To compute a sequence of approximate solutions to the least squares problem we start the recursion (11.3.33)–(11.3.34) with

$$b = \beta_1 u_1 = \beta_1 U_{k+1} e_1, \quad \beta_1 = \|b\|_2, \quad (11.3.38)$$

where e_1 denotes the first unit vector. We seek an approximate solution $x_k \in \mathcal{K}_k = \mathcal{K}_k(A^TA, A^T b)$. From the recursions (11.3.33)–(11.3.34) it follows that $\mathcal{K}_k = \text{span}(V_k)$, so we write $x_k = V_k y_k$. Multiplying (11.3.35) by y_k gives $Ax_k = AV_k y_k = U_{k+1} B_k y_k$. From (11.3.38) it follows that

$$b - Ax_k = U_{k+1} t_{k+1}, \quad t_{k+1} = \beta_1 e_1 - B_k y_k, \quad (11.3.39)$$

and these relations hold to working accuracy. Using 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 y_k to be the solution to the least squares problem

$$\min_{y_k} \|B_k y_k - \beta_1 e_1\|_2. \quad (11.3.40)$$

This forms the basis for the algorithm LSQR. Note the special form of the right-hand side, which holds because the starting vector is taken to be b . Now $x_k = V_k y_k$ solves

$$\min_{x_k \in \mathcal{K}_k} \|Ax - b\|_2,$$

where $\mathcal{K}_k = \mathcal{K}_k(A^T A, A^T b)$. Thus *mathematically* LSQR generates the same sequence of approximations as Algorithm 11.3.3 CGLS.

To solve (11.3.40) stably we need the QR factorization of $(B_k \mid \beta_1 e_1)$. This can be constructed by premultiplying with a sequence of Givens rotations, where $G_{k,k+1}$, $k = 1, 2, \dots$ is used to zero the element β_{k+1} . This step has already been considered in connection with the PLS method; see (8.4.45)–(8.4.48). It leads to the recursion

$$\begin{pmatrix} c_k & s_k \\ s_k & -c_k \end{pmatrix} \begin{pmatrix} \bar{\rho}_k & 0 & | & \bar{\phi}_k \\ \beta_{k+1} & \alpha_{k+1} & | & 0 \end{pmatrix} = \begin{pmatrix} \rho_k & \theta_{k+1} & | & \phi_k \\ 0 & \bar{\rho}_{k+1} & | & \bar{\phi}_{k+1} \end{pmatrix}, \quad (11.3.41)$$

where $\bar{\rho}_1 = \alpha_1$ and $\bar{\phi}_1 = \beta_1$. It is easily verified that R_k is an upper bidiagonal matrix. The least squares solution y_k and the norm of the corresponding residual are then obtained from

$$R_k y_k = \beta_1 e_1, \quad \|b - Ax_k\|_2 = |\rho_k|.$$

Note that the whole vector y_k differs from y_{k-1} . An updating formula for x_k can be derived using an idea due to Paige and Saunders. With $W_k = V_k R_k^{-1}$ we can write

$$\begin{aligned} x_k &= V_k y_k = \beta_1 V_k R_k^{-1} d_k = \beta_1 W_k d_k \\ &= \beta_1(W_{k-1}, w_k) \begin{pmatrix} d_{k-1} \\ \tau_k \end{pmatrix} = x_{k-1} + \beta_1 \tau_k w_k. \end{aligned} \quad (11.3.42)$$

The following MATLAB function performs k steps of the LSQR algorithm, assuming that all elements generated in B_k are nonzero. Note also that the Givens rotations should preferably be generated using a call to the function `givens(a,b)` in Algorithm 8.3.1.

Algorithm 11.6. LSQR (Paige and Saunders [470]).

```
function [x,nr] = lsqr(A,b,k)
% Performs k steps of the LSQR algorithm for
% the least squares problem min ||A x - b||_2.
% Returns approximate solution x and residual norm
% Initialization.
[m,n] = size(A);
x = zeros(n,1);
beta = norm(b); u = b/beta;
r = A'*u; nr = beta;
alpha = norm(r); v = r/alpha;
w = v; phi_bar = beta; rho_bar = alpha;
% Continue bidiagonalization
for i=1:k
    p = A*v - alpha*u;
    beta = norm(p); u = p/beta;
```

```

r = A'*u - beta*v;
alpha = norm(r); v = r/alpha;
% Construct and apply orthogonal transformation.
rho = norm([rho_bar,beta]);
c = rho_bar/rho; s = beta/rho;
theta = s*alpha; rho_bar = -c*alpha;
phi = c*phi_bar; phi_bar = s*phi_bar;
% Update the solution and residual norm
x = x + (phi/rho)*w;
w = v - (theta/rho)*w;
nr = nr*s;
end

```

Several stopping criteria are used in LSQR. Given an iterate x_k with a residual $r_k = b - Ax_k$, the algorithm stops if one of the following three conditions are satisfied:

1. $\|r_k\|_2 \leq \epsilon(\|A\|_F \|x_k\|_2 + \|b\|_2)$;
2. $\|A^T r_k\|_2 / \|r_k\|_2 \leq \alpha \|A\|_F$;
3. $1/\kappa(A) \leq \beta$;

Condition 1 is a test for consistent systems. The parameter ϵ are set according to the accuracy of the data or as a small multiple of the unit roundoff. Condition 2 with a small value of α is sufficient to ensure that x_k is an acceptable least squares solution. The quantity $\|A^T r_k\|_2 / \|r_k\|_2$ can vary dramatically, but tends to stabilize for larger values of k . Condition 3, where $\kappa(A)$ is estimated from the matrix B_k , is a regularization condition for ill-conditioned or singular problems.

For consistent systems $Ax = b$ Craig's method (CGME) can be used. This can also be expressed in terms of the Lanczos bidiagonalization. Let L_k be the lower bidiagonal matrix formed by the first k rows of B_k

$$L_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \alpha_3 & \\ & & \ddots & \ddots \\ & & & \beta_k & \alpha_k \end{pmatrix} \in \mathbf{R}^{k \times k}. \quad (11.3.43)$$

The relations (11.3.35) can now be rewritten as

$$AV_k = U_k L_k + \beta_{k+1} u_{k+1} e_k^T, \quad A^T U_k = V_k L_k^T. \quad (11.3.44)$$

The iterates in Craig's method can be computed as

$$L_k y_k = \beta_1 e_1, \quad x_k = V_k z_k. \quad (11.3.45)$$

Using (11.3.44) and (11.3.38) it follows that the residual vector satisfies

$$r_k = b - AV_k z_k = -\beta_{k+1} u_{k+1} (e_k^T z_k) = -\beta_{k+1} \eta_k u_{k+1},$$

and hence $U_k^T r_k = 0$. It can be shown that if $r_{k-1} \neq 0$ then $\alpha_k \neq 0$. Hence the vectors y_k and x_k can recursively be formed using

$$\eta_k = -\frac{\beta_k}{\alpha_k} \eta_{k-1}, \quad x_k = x_{k-1} + \eta_k v_k.$$

Craig's method is attractive in that in each step the error norm $\|x - x_k\|_2$ is minimized. As remarked before the residual norm $\|r_k\|_2$ usually fluctuates wildly for ill-conditioned problems.

11.3.5 Regularization by Iterative Methods

The discretization of an ill-posed problem gives rise to a linear system $Ax = b$, or, more generally, a linear least squares problem

$$\min_x \|Ax - b\|_2, \quad A \in \mathbf{R}^{m \times n}, \quad (11.3.46)$$

We recall from Section 8.6.4 some typical properties of such systems. The singular values σ_i of A will 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

$$x_i = \sum_{i=1}^n \frac{c_i}{\sigma_i} v_i, \quad c_i = u_i^T b, \quad (11.3.47)$$

the coefficients c_i decrease faster than σ_i . Therefore, in spite the large condition number of A , the problem with the exact right-hand side is in some sense quite well-conditioned. However, in practice the right hand side is contaminated by noise in a way that affects *all coefficients* c_i in (11.3.47) more or less equally. Unless some sort of regularization is introduced, the computed solution may be useless.

One method to regularize a problem studied in In Section 8.6.3 is to truncate the SVD expansion (11.3.47) after a small number of terms. For large scale applications, derived from two- or three-dimensional inverse problems, such direct methods may be impractical. Then an iterative regularization algorithm is an alternative. Such methods are advantageous to use also for linear systems coming from the discretization of convolution-type integral equations. In these the matrix A typically is a (dense) structured matrix, e.g., a Toeplitz matrix. Then matrix-vector products Ax and $A^T y$ can be computed in $O(n \log_2 n)$ multiplications using FFT; see [61, Sec. 8.4.2].

Like other methods applied to discrete ill-posed problems iterative methods will exhibit **semiconvergence**. By this we mean that the iterates first seem to converge to the true solution. Later, the effect of the perturbations in the right-hand side sets in and the iterative process starts to diverge from the true solution. Regularization is achieved by terminating the iterations before the unwanted irregular part of the solution has caused misconvergence. *The regularization is controlled by the number of iterations carried out.* Iterative regularization methods are much cheaper than TSVD to apply, in particular for large scale problems. Finding proper rules for terminating the iterations is not an easy problem.

One of the earliest methods used for iterative regularization is Landweber's method

$$x_{k+1} = x_k + \omega_k A^T(b - Ax_k). \quad (11.3.48)$$

The solution satisfies $A^T(b - Ax) = 0$. Hence, for a fixed parameter $\omega_k = \omega$ the error satisfies

$$x - x_k = (I - \omega A^T A)(x - x_{k-1}) = (I - \omega A^T A)^k(x - x_0).$$

Setting $x_0 = 0$ and using the SVD expansion (11.3.47), we get

$$x - x_k = \sum_{i=1}^n (1 - \omega \sigma_i^2)^k \frac{c_i}{\sigma_i} v_i, \quad (11.3.49)$$

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 is called the filter factor for Landweber's method.

Taking $\omega = \sigma_1^{-2}$ in (11.3.49), 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 for large values of σ_i will converge more quickly. On the other hand, it will take many iterations before the components of the solution corresponding to small singular values have converged to any extent. In the beginning the iterates x_k will 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 what was called semiconvergence. It can be deduced that terminating the iterations after k iterations roughly gives similar result as truncating the SVD expansion for

$$\sigma_i \leq \mu \sim k^{-1/2}.$$

Thus Landweber's method produces a *sequence* of less and less regularized solutions.

A related method is the **iterated Tikhonov method**

$$x_{k+1} = x_k + (A^T A + \mu^2 I)^{-1} A^T(b - Ax_k), \quad (11.3.50)$$

which corresponds to taking $p(\lambda) = (\lambda + \mu^2)^{-1}$. With $x_0 = 0$ the next iterate is

$$x_1 = (A^T A + \mu^2 I)^{-1} A^T b.$$

Each further iteration step involves the computation of a Tikhonov regularized solution. The filter functions have the form

$$\varphi_k = 1 - \left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \mu^2}\right)^k.$$

For $k > 1$ this gives slightly sharper cutoff than standard Tikhonov regularization.

In general, projections methods have a regularizing effect since 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 method 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 .

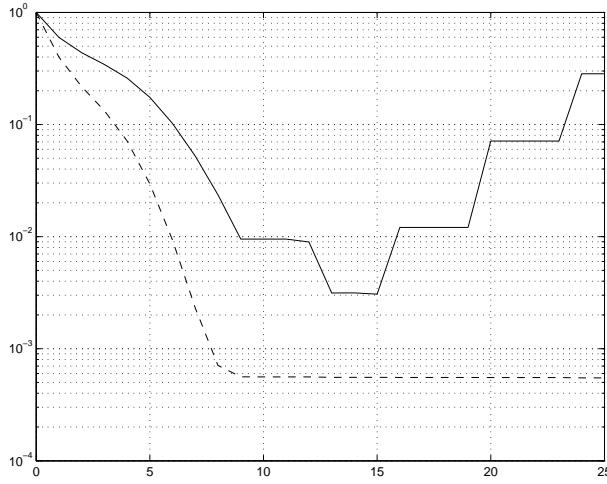


Figure 11.3.2. Relative error $\|x_k - x\|_2 / \|x\|_2$ and residual $\|Ax_k - b\|_2 / \|b\|_2$ for LSQR solutions after k steps.

Example 11.3.1.

Consider the ill-posed linear system $Ax = b$, $A \in \mathbf{R}^{n \times n}$ in Example 8.4.2. Figure 11.3.2 shows the relative error $\|x_k - x\|_2 / \|x\|_2$ and residual $\|Ax_k - b\|_2 / \|b\|_2$ after k steps of LSQR. In exact arithmetic these should be identical to PLS. 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.

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. As remarked, this is a non-trivial problem, since an a priori knowledge of the number of iterations usually is lacking.

A effective way to avoid misconvergence in Krylov subspace methods is to

combine them with an inner regularizing algorithm. For example, the LSQR method can be applied to the regularized problem

$$\min_x \left\| \begin{pmatrix} A \\ \mu I_n \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2, \quad (11.3.51)$$

where μ is a regularization parameter. Such iterative methods are called **hybrid methods** and are easy to implement. The original implementation of the LSQR algorithm by Paige and Saunders [470] includes a fixed parameter μ can be chosen as an option (see Section 8.4.5).

We consider a hybrid LSQR in more detail. Only the sequence of small bidiagonal subproblems in LSQR needs to be regularized. If Tikhonov regularization is used, then in the k th step the approximation is taken to be $x_k(\mu) = V_k y_k(\mu)$, where $y_k(\mu)$ solves

$$\min_{y_k} \left\| \begin{pmatrix} B_k \\ \mu I_k \end{pmatrix} y_k - \beta_1 \begin{pmatrix} e_1 \\ 0 \end{pmatrix} \right\|_2. \quad (11.3.52)$$

Here μ is a suitably chosen regularization parameter. The solution y_k can be computed at a small cost matrix if we take advantage of the lower bidiagonal form of $B_k \in \mathbf{R}^{k+1 \times k}$. Let Q be an orthogonal matrix such that

$$Q^T \left(\begin{array}{c|c} B_k & \beta_1 e_1 \\ \mu I_k & 0 \end{array} \right) = \left(\begin{array}{c|c} \tilde{B}_k & g_1 \\ 0 & g_2 \end{array} \right).$$

Then the solution is obtained by solving the small upper bidiagonal system $\tilde{B}_k y_k = g_1$. The matrix Q can be constructed as a product of plane rotations. It suffices to consider the case $k = 3$.

$$\begin{aligned} &\rightarrow \begin{pmatrix} \alpha_1 & & \\ \beta_2 & \alpha_2 & \\ & \beta_3 & \alpha_3 \\ & & \beta_4 \end{pmatrix} && \rightarrow \begin{pmatrix} \tilde{\alpha}_1 & & \\ \beta_2 & \alpha_2 & \\ \beta_3 & \alpha_3 & \\ & & \beta_4 \end{pmatrix} && \rightarrow \begin{pmatrix} \tilde{\alpha}_1 & \gamma_2 & \\ 0 & \tilde{\alpha}_2 & \\ & \beta_3 & \alpha_3 \\ & & \beta_4 \end{pmatrix} - \\ &\rightarrow \begin{pmatrix} \mu & & \\ & \mu & \\ & & \mu \end{pmatrix} && \rightarrow \begin{pmatrix} 0 & & \\ & \mu & \\ & & \mu \end{pmatrix} && \rightarrow \begin{pmatrix} 0 & & \\ & \mu & \\ & & \mu \end{pmatrix} \end{aligned}$$

First a rotation in the $(1, n+1)$ -plane annihilates the element $(n+1, 1)$ and modifies the $(1, 1)$ element $\tilde{\alpha}_1 = \sqrt{\alpha_1^2 + \mu^2}$. A rotation in the $(1, 2)$ -plane then annihilates the element β_2 and creates a nonzero element in position $(1, 2)$. The extra cost just equals k square roots.

In Section 8.6.4 we showed that a regularized problem with $L \neq I$ but nonsingular could be transformed into a problem of standard form

$$\min_y \left\| \begin{pmatrix} AL^{-1} \\ \mu I \end{pmatrix} y - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2 \quad (11.3.53)$$

by setting $x = L^{-1}y$. To solve (11.3.53) by an iterative method the explicit formation of the product AL^{-1} should be avoided. Instead the transformation should be

applied implicitly. For example, in CGLS we can implicitly work with the matrix $\tilde{A} = AL^{-1}$ by splitting the matrix-vector products as

$$\tilde{A}p = A(L^{-1}p), \quad \tilde{A}^T r = (L^{-T}r)A^T. \quad (11.3.54)$$

This requires in each step the solution of the two linear systems

$$L^T y_j = A^T u_j, \quad Lw_j = v_j.$$

Also in the case that L has a nontrivial nullspace the more complicated transformation given in Section 8.6.4 can be applied implicitly. Similar modification can be used when LSQR is used instead of CGLS.

Formally the matrix L^{-1} acts here as a “preconditioner” for the iterative method; see Section 11.5. By this we mean that Because the singular values of AL^{-1} differs from those of A the convergence behavior will be different than for the unmodified iteration. Usually a preconditioner is used to improve the condition number of A . For ill-posed problems this makes no sense. Rather, we would like to improve the part of the singular value spectrum that is relevant for the regularized solution and leave the other part unchanged or possibly suppressed. How to achieve this is a subject for current research.

Review Questions

- 3.1** Give two advantages with writing the normal equations in the factored form $A^T(Ax - b) = 0$, when applying an iterative method.
- 3.2** What is the drawback with using an iterative method for the augmented system to solve for r and x simultaneously?
- 3.3** (a) Landweber’s iteration for solving the normal equations is related to a classic method for solving a nonsingular linear systems. Which?
(b) For what values of the parameter ω does the stationary Landweber’s method converge?
- 3.4** What special iteration results when Jacobi’s method is applied to the normal equations?
- 3.5** What property of Landweber’s iteration method is useful when it is applied to an ill-posed linear system? Derive the filter factors for Landweber’s method.
- 3.6** Derive Algorithms CGLS and CGME by applying the conjugate gradient algorithm to the normal equations $A^T Ax = A^T b$ and $AA^T y = b$, $x = A^T y$, respectively.

Problems and Computer Exercises

- 3.1** 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 which minimizes $\|x - x^{(0)}\|_2$.

11.4 Unsymmetric Problems

A simple approach to solving a square nonsymmetric linear system $Ax = b$ is to symmetrize the problem by forming the normal system $A^H A x = A^H b$ or $AA^H y = b$, $y = A^H x$. Then one of the iterative methods described in the previous section can be applied. A serious drawback with this approach is that these methods often converge very slowly. This is related to the fact that the singular values of $A^H A$ (and AA^H) are the square of the singular values of A . There also are applications where it is not possible to compute matrix-vector products $A^H x$ —note that A may only exist as subroutine for computing Ax .

An ideal conjugate gradient-like method for unsymmetric systems would be characterized by one of the properties (11.2.29) or (11.2.40). We would also like to be able to base the implementation on a short vector recursion. As shown independently by Faber and Manteuffel [194] and V. V. Voevodin [596], such an ideal method essentially can only exist for matrices of very special form. In particular, a two term recursion like in the CG method is only possible in case A either has a minimal polynomial of degree ≤ 1 , or is Hermitian, or is of the form

$$A = e^{i\theta}(B + \rho I), \quad B = -B^H,$$

where θ and ρ are real. Hence the class essentially consists of shifted and rotated Hermitian matrices.

11.4.1 Arnoldi's Method and GMRES

In Section 10.4.2 we gave an algorithm to reduce a matrix $A \in \mathbf{C}^{n \times n}$ to upper Hessenberg form by a sequence of orthogonal similarity transformation. The **Arnoldi's process** performs the same reduction using only matrix-vector operations. It generalizes Lanczos method to non-Hermitian matrices. The main difference is that in the Arnoldi process each new basis vector has to be orthogonalized with respect to *all previous basis vectors*.

Starting with a vector v_1 , the Arnoldi process builds an orthogonal basis for the Krylov subspaces $\mathcal{K}_k(A, v_1)$, $k = 1, 2, \dots$. After k steps the Arnoldi process has generated a matrix

$$V = (v_1, v_2, \dots, v_{k+1})$$

with orthonormal columns and a Hessenberg matrix

$$H_k = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ \ddots & \ddots & \ddots & \vdots \\ & h_{k,k-1} & h_{kk} \end{pmatrix} \in \mathbf{C}^{k \times k}. \quad (11.4.1)$$

such that the fundamental relation

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T, \quad k = 1, 2, \dots, \quad (11.4.2)$$

holds. To develop the recursion it is convenient to introduce the rectangular Hessenberg matrix

$$\hat{H}_k = \begin{pmatrix} H_k \\ h_{k+1,k} e_k^T \end{pmatrix} \in \mathbf{R}^{(k+1) \times k}. \quad (11.4.3)$$

so that

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1} \hat{H}_k, \quad (11.4.4)$$

Suppose we have computed V_j and $\hat{H}_{j-1} \in \mathbf{R}^{j \times (j-1)}$ in (11.4.4). In the next step we form the vector $w = Av_j$ and orthogonalize this against V_j . Using classical Gram–Schmidt this gives

$$h_j = V_j^H w, \quad r_j = w - V_j h_j.$$

If $\beta_j = \|r_j\|_2 = 0$ the process breaks down. Otherwise set $h_{j+1,j} = \beta_j$, $v_{j+1} = r_j/h_{j+1,j}$ and update

$$V_{j+1} = (V_j \quad v_{j+1}), \quad H_j = (\hat{H}_{j-1} \quad h_j).$$

The Arnoldi process breaks down at step j if and only if $A^j v_1 \in \mathcal{K}_j(A, v_1)$. An implementation of this process using classical Gram–Schmidt is given below.

Algorithm 11.7. *The Arnoldi Process.*

```

function [V,H,f] = arnoldi(A,v1,k)
% ARNOLDI computes, given A, v1, and k, a Hessenberg
% matrix H and matrix V with orthogonal columns spanning
% the Krylov subspace K_k(A,v1) such that A*V = V*H + r*e_k'
[n,n] = size(A);
V = [] ; H = [] ;
beta = norm(v1);
for j=1:k
    if (beta ~= 0),
        v = v1/beta; V = [V,v];
        w = A*v;
        h = V'*w; H = [H,h];
        f = w - V*h;
    end
end

```

```

    end
    beta = norm(f);
    be = zeros(1,j);  be(j) = beta;
    if (j < k), H = [H; be]; end
end

```

The construction ensures that (11.4.2) is satisfied with $h_{k+1,k} = \beta_k$. Note that the statements $V = []$; initialize the matrix as an empty matrix. This convenient convention in MATLAB often simplifies the description of an algorithm and is as natural as allowing empty sums and products. For a rigorous treatment of the algebra of empty matrices, see de Boor [71].

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

$$x_k = V_k y_k \in \mathcal{K}_k(b, A), \quad (11.4.5)$$

In exact arithmetic the result after k steps is a matrix $V_k = (v_1, \dots, v_k)$, that (in exact arithmetic) gives an orthogonal basis for the Krylov subspace $\mathcal{K}_k(A, v_1)$ and a related square Hessenberg matrix $H_k = (h_{ij}) \in \mathbf{R}^{k \times k}$. In the above version of the Arnoldi method the orthogonalization is performed by CGS. As shown in Section 8.3.4 the loss of orthogonality using CGS is much worse than with MGS. Also with MGS there will be some loss of orthogonality, but as shown in [465], this is not crucial for the accuracy of the computed approximate solution x_k . *The Arnoldi vectors will lose orthogonality completely only after the residual $r_k = b - Ax_k$ is reduced close to its final level.*

There are two different ways to choose the approximation x_k in (11.4.5). In the **full orthogonalization method** (FOM) x_k is determined by the Galerkin condition

$$r_k \perp \mathcal{K}_k(b, A), \quad r_k = b - Ax_k.$$

Using (11.4.2) the residual r_k can be expressed as

$$r_k = b - AV_k y_k = \beta_1 v_1 - V_k H_k y_k - \beta_k v_{k+1} e_k^T. \quad (11.4.6)$$

The Galerkin condition gives $V_k^T r_k = \beta_1 e_1 - H_k y_k = 0$. Hence, y_k is obtained as the solution of

$$y_k = \beta_1 H_k^{-1} e_1.$$

This is a small Hessenberg linear system, which is easy to solve. We remark that H_k is nonsingular if $\mathcal{K}_k(b, A)$ has full rank.

In the **generalized minimum residual method** (GMRES) y_k is chosen so that $\|b - Ax_k\|_2$ is minimized. 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 (11.4.3). Since (in exact arithmetic) V_{k+1} has orthogonal columns, it follows that $\|r_k\|_2$ is minimized by taking y_k to be the solution of the least squares problem

$$\min_{y_k} \|\beta_1 e_1 - \hat{H}_k y_k\|_2. \quad (11.4.7)$$

Notice that this ensures that in GMRES $\|r_k\|_2$ is monotonically decreasing as the iteration proceeds.

The Arnoldi process breaks down at step k if and only if $A^k b \in \mathcal{K}_k(b, A)$. Then z_k vanishes, $\beta_k = 0$ and by (11.4.7) $AV_k = V_k H_k$. 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 the important property (in exact arithmetic) that *GMRES does not break down before the exact solution is found*. It follows that GMRES terminates in at most n steps.

We now discuss the solution of the least squares subsystem (11.4.7) in GMRES. To solve this the QR factorization

$$Q_k^T (\hat{H}_k e_1) = \begin{pmatrix} R_k & d_k \\ 0 & \rho_k \end{pmatrix}, \quad Q_k^T = G_{k,k+1} G_{k-1,k} \cdots G_{12}, \quad (11.4.8)$$

is computed using a sequence of plane rotations. Here $G_{j+1,j}$ is chosen to zero the subdiagonal element $h_{j+1,j}$. The solution to (11.4.7) and its residual is then obtained from

$$R_k y_k = \beta_1 d_k, \quad \|r_k\|_2 = \beta_1 |\rho_k|. \quad (11.4.9)$$

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* (11.4.8) at each step of the Arnoldi process. To derive the updating formulas for step $j = k$ 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

$$Q_{k-1}^T h_k = G_{k-1,k} \cdots G_{12} h_k = \begin{pmatrix} c_{k-1} \\ \delta_k \end{pmatrix}, \quad (11.4.10)$$

The rotation $G_{k,k+1}$ is then determined by

$$G_{k,k+1} \begin{pmatrix} \delta_k \\ h_{k+1,k} \end{pmatrix} = \begin{pmatrix} \gamma_k \\ 0 \end{pmatrix}. \quad (11.4.11)$$

and gives the last element in the k th column in R_k .

Proceeding similarly with the right hand side, we have

$$Q_k^T e_1 = G_{k,k+1} \begin{pmatrix} Q_{k-1}^T e_1 \\ 0 \end{pmatrix} = G_{k,k+1} \begin{pmatrix} d_{k-1} \\ \rho_{k-1} \\ 0 \end{pmatrix} \begin{pmatrix} d_{k-1} \\ \tau_k \\ \rho_k \end{pmatrix} \equiv \begin{pmatrix} d_k \\ \rho_k \end{pmatrix}. \quad (11.4.12)$$

(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 $x_k = V_k y_k$. Note that the whole vector y_k differs from y_{k-1} so the expensive operation of forming can be delayed until GMRES has converged. Alternatively, an updating formula for x_k can be derived as follows: Set $W_k R_k = V_k$, which can be written

$$(W_{k-1}, w_k) \begin{pmatrix} R_{k-1} & c_{k-1} \\ 0 & \gamma_k \end{pmatrix} = (V_{k-1}, v_k).$$

Equating the first block columns gives $W_{k-1} R_{k-1} = V_{k-1}$, which shows that the first $k-1$ columns of W_k equal W_{k-1} . Equating the last columns and solving for w_k we get

$$w_k = (v_k - W_{k-1} r_{k-1}) / \gamma_k \quad (11.4.13)$$

Then from (11.3.42) $x_k = x_{k-1} + \beta_1 \tau_k w_k$. Note that if this formula is used we only need the last column of the matrix R_k . (We now need to save W_k but not R_k .)

The steps in the resulting GMRES algorithm can now be summarized as follows:

1. Obtain last column of \hat{H}_k from the Arnoldi process and apply old rotations $g_k = G_{k-1,k} \cdots G_{12} h_k$.
2. Determine rotation $G_{k,k+1}$ and new column in R_k , i.e., c_{k-1} and γ_k according to (11.4.11). This also determines τ_k and $|\rho_k| = \|r_k\|_2$.
3. If x_{k-1} is recursively updated, then compute w_k using (11.4.12) and x_k from (11.3.42).

Suppose that the matrix A is diagonalizable,

$$A = X \Lambda X^{-1}, \quad \Lambda = \text{diag}(\lambda_i).$$

Then, using the property that the GMRES approximations minimize the Euclidian norm of the residual $r_k = b - Ax_k$ in the Krylov subspace $\mathcal{K}_k(A, b)$, it can be shown that

$$\frac{\|r_k\|_2}{\|b\|_2} \leq \kappa_2(X) \min_{q_k} \max_{i=1,2,\dots,n} |q_k(\lambda_i)|, \quad (11.4.14)$$

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 Section 11.2.3. This results shows that if A 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 we can also expect GMRES to provide accurate approximations after about p iterations.

Because of the factor $\kappa_2(X)$ in (11.4.14) an upper bound for the rate of convergence can no longer be deduced from the spectrum $\{\lambda_i\}$ of A alone. In the special case that A is normal we have $\kappa_2(X) = 1$, and the convergence is related to the complex approximation problem

$$\min_{q_k} \max_{i=1,2,\dots,n} |q_k(\lambda_i)|, \quad q_k(0) = 1.$$

Because complex approximation problems are harder than real ones, no simple results are available even for this special case.

In practice it is often observed that GMRES (like the CG method) has a so-called superlinear convergence. By this we mean that the rate of convergence improves as the iteration proceeds. It has been proved that this is related to the convergence of Ritz values 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.

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 . In practice the number of steps taken by GMRES must therefore often be limited by **restarting** GMRES after each m iterations. In practice, m typically is chosen between 10 and 30. We denote the corresponding algorithm GMRES(m). GMRES(m) cannot break down (in exact arithmetic) before the true solution has been produced, but for $m < n$ GMRES may never converge.

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 then used to precondition the restarted iteration.

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 . This leads to the method MINRES by Paige and Saunders mentioned in Section 11.4.1.

11.4.2 The Bi-Lanczos and Bi-CG Methods

The GMRES method is related to the reduction of a unsymmetric matrix $A \in \mathbf{R}^{n \times n}$ to Hessenberg form by an orthogonal similarity $H = Q^T AQ$. It gives up the short recurrences of the CG method. Another possible generalization proposed by Lanczos [391] is related to the reduction of A to tridiagonal form by a general similarity transformation.

$$W^T AV = T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \gamma_2 & \alpha_2 & \beta_3 & & \\ & \gamma_3 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \gamma_n & \alpha_n \end{pmatrix}, \quad (11.4.15)$$

where $W^T V = I$, i.e., $W^T = V^{-1}$. Setting $V = (v_1, \dots, v_n)$ and $W = (w_1, \dots, w)$, it follows that the two vector sequences $\{v_1, \dots, v_n\}$ and $\{w_1, \dots, w\}$, are **bi-orthogonal**

$$w_i^T v_j = \begin{cases} 1, & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (11.4.16)$$

Comparing columns in $AV = VT$ and $A^T W = WT^T$ we find (with $v_0 = w_0 = 0$)

$$Av_k = \beta_k v_{k-1} + \alpha_k v_k + \gamma_{k+1} v_{k+1}, \quad (11.4.17)$$

$$A^T w_k = \gamma_k w_{k-1} + \alpha_k w_k + \beta_{k+1} w_{k+1}. \quad (11.4.18)$$

$k = 1 : n$. Rearranging these equation we get the recurrence relations

$$\tilde{v}_{k+1} \equiv \gamma_{k+1} v_{k+1} = (A - \alpha_k I) v_k - \beta_k v_{k-1}, \quad (11.4.19)$$

$$\tilde{w}_{k+1} \equiv \beta_{k+1} w_{k+1} = (A^T - \alpha_k I) w_k + \gamma_k w_{k-1}, \quad (11.4.20)$$

Multiplying equation (11.4.19) by w_k^T , and using the bi-orthogonality we have

$$\alpha_k = w_k^T A v_k.$$

To satisfy the bi-orthogonality relation (11.4.20) for $i = j = k + 1$ it suffices to choose γ_{k+1} and β_{k+1} so that.

$$\gamma_{k+1} \beta_{k+1} = \tilde{w}_{k+1}^T \tilde{v}_{k+1}.$$

Hence there is some freedom in choosing these scale factors.

If we denote

$$V_k = (v_1, \dots, v_k), \quad W_k = (w_1, \dots, w_k),$$

then we have $W_k^T A V_k = T_k$, and the recurrences in this process can be written in matrix form as

$$A V_k = V_k T_k + \gamma_{k+1} v_{k+1} e_k^T, \quad (11.4.21)$$

$$A^T W_k = W_k T_k^T + \beta_{k+1} w_{k+1} e_k^T. \quad (11.4.22)$$

By construction these vector sequences form basis vectors for the two Krylov spaces

$$\mathcal{R}(V_k) = \mathcal{K}_k(A, v_1), \quad \mathcal{R}(W_k) = \mathcal{K}_k(A^T, w_1). \quad (11.4.23)$$

We summarize the algorithm for generating the two sequences of vectors v_1, v_2, \dots and w_1, w_2, \dots :

Algorithm 11.8. *The Lanczos Bi-Orthogonalization Process.*

Let v_1 and w_1 be two vectors such that $w_1^T v_1 = 1$. The algorithm computes after k steps a symmetric tridiagonal matrix $T_k = \text{trid}(\gamma_j, \alpha_j, \beta_{j+1})$ and two matrices W_k and V_k with (in exact arithmetic) bi-orthogonal columns spanning the Krylov subspaces $\mathcal{K}_k(A, v_1)$ and $\mathcal{K}_k(A^T, w_1)$:

```

 $w_0 = v_0 = 0;$ 
 $\beta_1 = \gamma_1 = 0;$ 
for  $j = 1, 2, \dots$ 
     $\alpha_j = w_j^T A v_j;$ 
     $v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1};$ 
     $w_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1};$ 

```

```

 $\delta_{j+1} = |w_{j+1}^T v_{j+1}|^{1/2};$ 
if  $\delta_{j+1} = 0$  then exit;
 $\beta_{j+1} = (w_{j+1}^T v_{j+1})/\delta_{j+1};$ 
 $v_{j+1} = v_{j+1}/\delta_{j+1};$ 
 $w_{j+1} = w_{j+1}/\beta_{j+1};$ 
end

```

Note that if $A = A^T$, $w_1 = v_1$, and we take $\beta_k = \gamma_k$, then the two sequences generated will be identical. The process then is equivalent to the symmetric Lanczos process.

There are two cases when the above algorithm breaks down. The first occurs when either \tilde{v}_{k+1} or \tilde{w}_{k+1} (or both) is null. In this case it follows from (11.4.21)–(11.4.22) that an invariant subspace has been found. If $v_{k+1} = 0$, then $AV_k = V_k T_k$ and $\mathcal{R}(V_k)$ is an A -invariant subspace. If $w_{k+1} = 0$, then $A^T W_k = W_k T_k^T$ and $\mathcal{R}(W_k)$ is an A^T -invariant subspace. This is called regular termination. The second case, called *serious breakdown*, occurs when $\tilde{w}_k^T \tilde{v}_k = 0$, with neither \tilde{v}_{k+1} nor \tilde{w}_{k+1} null.

The Lanczos bi-orthogonalization algorithm is the basis for several iterative methods for unsymmetric linear system $Ax = b$. Let $r_0 = b - Ax_0$ and take

$$\beta_1 v_1 = r_0, \quad \beta_1 = \|r_0\|_2,$$

and set $w_1 = v_1$. We seek an approximate solution x_k in the Krylov subspace $\mathcal{K}_k(A, v_1)$

$$x_k = V_k y_k \in \mathcal{K}_k(A, v_1).$$

The corresponding residual is

$$r_k = b - Ax_k = \beta_1 v_1 - AV_k y_k,$$

Here y_k is determined so that the Galerkin condition $r_k \perp \mathcal{K}_k(w_1, A^T)$ is satisfied, or equivalently $W_k^T r_k = 0$. Using (11.4.21) and the bi-orthogonality conditions $W_k^T V_k = I$ this gives

$$W_k^T (\beta_1 v_1 - AV_k y_k) = \beta_1 e_1 - T_k y_k = 0. \quad (11.4.24)$$

Hence, if the matrix T_k is nonsingular y_k satisfies the tridiagonal system $T_k y_k = \beta_1 e_1$ and $x_k = V_k y_k$. The resulting method is known as the **Bi-Lanczos method**.

If A is symmetric, this method becomes the SYMMLQ method, see Section 11.4.1. We remark again that in the unsymmetric case this method can break down without producing a good approximate solution to $Ax = b$. In case of a serious breakdown, the method can be restarted from the beginning with the new starting vector r_k . A drawback with this is that the Krylov subspaces that have been built are discarded. This may destroy the possibility of faster convergence. The occurrence of breakdown can be shown to equivalent to that of identical entries

appearing in the Padé table of approximants to a function. A way around this problem is to use the so-called *look-ahead strategy*. In this several successive basis vectors for the Krylov subspaces are taken together and made blockwise bi-orthogonal.

The Bi-Lanczos method can be written in a form more like the conjugate gradient algorithm. In this the LU factorization without pivoting

$$T_k = L_k U_k, \quad (11.4.25)$$

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 or **Bi-CG method**. The LU factorization may fail to exist for some k . This is an additional cause for breakdown of the Bi-CG algorithm. Such a breakdown of the *second kind* can be avoided in the Bi-Lanczos formulation by making an LU decomposition with 2 by 2 block diagonal elements.

To derive the Bi-CG algorithm from the Lanczos bi-orthogonalization we use the LU factorization (11.4.25) and write

$$x_k = x_0 + V_k T_k^{-1}(\beta e_1) = x_0 + P_k L_k^{-1}(\beta e_1),$$

where $P_k = V_k U_k^{-1}$. Similarly, define the matrix $\tilde{P}_k = W_k L_k^{-T}$. Then the columns of P_k and \tilde{P}_k are A -conjugate, since

$$\tilde{P}_k^T A P_k = L_k^{-1} W_k^T A V_k U_k^{-1} = L_k^{-1} T_k U_k^{-1} = I.$$

Notice that the matrices U_k and L_{-k} are upper bidiagonal. Therefore successive column in P_k and \tilde{P}_k^T can be obtained by two-term recursions. Similarly, x_k can be obtained by updating x_{k-1} as in the CG method. An outline of the Bi-CG algorithm without provision for breakdown is given below.

Algorithm 11.9. *Bi-Conjugate Gradient Algorithm.*

Set $r_0 = b - Ax_0$ and choose \tilde{r}_0 so that $(r_0, \tilde{r}_0) \neq 0$.

```

 $p_0 = r_0; \quad \tilde{p}_0 = \tilde{r}_0;$ 
 $\rho_0 = (\tilde{r}_0, r_0);$ 
for  $j = 0, 1, 2, \dots$ 
     $v_j = Ap_j; \quad \alpha_j = \rho_j / (\tilde{p}_j, v_j);$ 
     $x_{j+1} = x_j + \alpha_j p_j;$ 
     $r_{j+1} = r_j - \alpha_j v_j;$ 
     $\tilde{r}_{j+1} = \tilde{r}_j - \alpha_j (A^T \tilde{p}_j);$ 
     $\rho_{j+1} = (\tilde{r}_{j+1}, r_{j+1});$ 
     $\beta_j = \rho_{j+1} / \rho_j;$ 
     $p_{j+1} = r_{j+1} + \beta_j p_j;$ 
     $\tilde{p}_{j+1} = \tilde{r}_{j+1} + \beta_j \tilde{p}_j;$ 
end

```

The vectors r_j and \tilde{r}_j are in the same direction as v_{j+1} and w_{j+1} , respectively. Hence they form a biorthogonal sequence. Note that in Bi-CG the most time-consuming operations Ap_j and $A^T\tilde{p}_j$ can be carried out in parallel.

One can encounter convergence problems with Bi-CG, since for general matrices the bilinear form

$$[x, y] = (\psi(A^T)x, \psi(A)y)$$

used to define bi-orthogonality, does not define an inner product. Therefore if \tilde{r}_0 is chosen unfavorably, it may occur that ρ_j or (\tilde{p}_j, v_j) is zero (or very small), without convergence having taken place. Nothing is minimized in the Bi-CG 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.

A method called the **Quasi-Minimal Residual** (QMR) method is related to Bi-CG similarly to the way MINRES is related to CG. It can be developed as follows. After k steps of the unsymmetric Lanczos process we have from the relation (11.4.21) that

$$AV_k = V_{k+1}\hat{T}_k, \quad \hat{T}_k = \begin{pmatrix} T_k \\ \gamma_{k+1}e_k^T \end{pmatrix},$$

where \hat{T}_k is an $(k+1) \times k$ tridiagonal matrix. We can now proceed as was done in developing GMRES. If we take $v_1 = \beta r_0$, the the residual associated with with an approximate solution of the form $x_k = x_0 + V_k y$ is given by

$$\begin{aligned} b - Ax_k &= b - A(x_0 + V_k y) = r_0 - AV_k y \\ &= \beta v_1 - V_{k+1}\hat{T}_k y = V_{k+1}(\beta e_1 - \hat{T}_k y). \end{aligned} \quad (11.4.26)$$

Hence the norm of the residual vector is

$$\|b - Ax_k\|_2 = \|V_{k+1}(\beta e_1 - \hat{T}_k y)\|_2.$$

If the matrix V_{k+1} had orthonormal columns then the residual norm would become $\|(\beta e_1 - \hat{T}_k y)\|_2$, as in GMRES, and a least squares solution in the Krylov subspace could be obtained by solving

$$\min_y \|\beta e_1 - \hat{T}_k y\|_2.$$

for y_k and taking $x_k = x_0 + V_k y_k$.

Recent surveys on progress in iterative methods for non-symmetric systems are given by Freund, Golub and Nachtigal [219] and Golub and van der Vorst [274]. There is a huge variety of methods to choose from. Unfortunately in many practical situations it is not clear what method to select. In general there is no best method. In [446] examples are given which show that, depending on the linear system to be solved, each method can be clear winner or clear loser! Hence insight into the characteristics of the linear system is needed in order to discriminate between methods. This is different from the symmetric case, where the rate of convergence can be deduced from the spectral properties of the matrix alone.

11.4.3 Transpose-Free Methods

A disadvantage of the methods previously described for solving non-symmetric linear systems is that they require subroutines for the calculation of both Ax and $A^T y$ for arbitrary vectors x and y . If the data structure favors the calculation of Ax then it is often less favorable for the calculation of $A^T y$. Moreover, for some problems deriving from ordinary differential equations the rows of A arise naturally from a finite difference approximation and the matrix product Ax may be much more easily computed than $A^T y$. These considerations have led to the development of “transpose-free” methods.

The first of the transpose-free iterative methods **Bi-CGS**, is a modification of the Bi-CG algorithm. Here CGS stands for “conjugate gradient squared”. The key observation behind this algorithm is the following property of the vectors generated in the Bi-CG algorithm. Taking into account that $p_0 = r_0$, it is easily shown that there are polynomials $\phi_j(x)$ and $\psi_j(x)$ of degree such that for $j = 1, 2, \dots$,

$$\begin{aligned} r_j &= \phi_j(A) r_0, & \tilde{r}_j &= \phi_j(A^T) \tilde{r}_0, \\ p_j &= \psi_j(A) r_0, & \tilde{p}_j &= \psi_j(A^T) \tilde{r}_0. \end{aligned}$$

That is r_j and \tilde{r}_j are obtained by premultiplication by *the same polynomial* $\phi(t)$ in A and A^T , respectively. The same is true for p_j and \tilde{p}_j for the polynomial $\psi(t)$. Using the fact that the polynomial of a transpose matrix is the transpose of the polynomial, it follows that the quantities needed in the Bi-CG algorithm can be expressed as

$$(\tilde{r}_j, r_j) = (\tilde{r}_0, \phi_j^2(A) r_0), \quad (\tilde{p}_j, Ap_j) = (\tilde{p}_0, \psi_j^2(A) r_0).$$

Therefore, if we somehow could generate the vectors $\phi_j(A)^2 r_0$ and $\psi_j(A)^2 p_0$ directly, then no products with A^T would be required. To achieve this we note that from the Bi-CG algorithm we have the relations, $\phi_0(A) = \psi_0(A) = I$,

$$\phi_{j+1}(A) = \phi_j(A) - \alpha_j A \psi_j(A), \tag{11.4.27}$$

$$\psi_{j+1}(A) = \phi_{j+1}(A) + \beta_j \psi_j(A), \tag{11.4.28}$$

Squaring these relations we obtain

$$\begin{aligned} \phi_{j+1}^2 &= \phi_j^2 - 2\alpha_j A \phi_j \psi_j + \alpha_j^2 A^2 \psi_j^2, \\ \psi_{j+1}^2 &= \phi_{j+1}^2 + 2\beta_j \phi_{j+1} \psi_j + \beta_j^2 \psi_j^2. \end{aligned}$$

where we have omitted the argument A . For the first cross product term we have using (11.4.28)

$$\phi_j \psi_j = \phi_j(\phi_j + \beta_{j-1} \psi_{j-1}) = \phi_j^2 + \beta_{j-1} \phi_j \psi_{j-1}.$$

From this and (11.4.27) we get for the other cross product term

$$\phi_{j+1} \psi_j = (\phi_j - \alpha_j A \psi_j) \psi_j = \phi_j \psi_j - \alpha_j A \psi_j^2 = \phi_j^2 + \beta_{j-1} \phi_j \psi_{j-1} - \alpha_j A \psi_j^2.$$

Summarizing, we now have the three recurrence relations, which are the basis of the Bi-CGS algorithm:

$$\begin{aligned}\phi_{j+1}^2 &= \phi_j^2 - \alpha_j A (2\phi_j^2 + 2\beta_{j-1}\phi_j\psi_{j-1} - \alpha_j A \psi_j^2), \\ \phi_{j+1}\psi_j &= \phi_j^2 + \beta_{j-1}\phi_j\psi_{j-1} - \alpha_j A \psi_j^2 \\ \psi_{j+1}^2 &= \phi_{j+1}^2 + 2\beta_j\phi_{j+1}\psi_j + \beta_j^2\psi_j^2.\end{aligned}$$

If we now define

$$r_j = \phi_j^2(A)r_0, \quad q_j = \phi_{j+1}(A)\psi_j(A)r_0, \quad p_j = \psi_j^2(A)r_0. \quad (11.4.29)$$

we get

$$r_{j+1} = r_j - \alpha_j A (2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j A p_j), \quad (11.4.30)$$

$$q_j = r_j + \beta_{j-1}q_{j-1} - \alpha_j A p_j, \quad (11.4.31)$$

$$p_{j+1} = r_{j+1} + 2\beta_j q_j + \beta_j^2 p_j. \quad (11.4.32)$$

These recurrences can be simplified by introducing the auxiliary vectors

$$u_j = r_j + \beta_{j-1}q_{j-1}, \quad d_j = u_j + q_j. \quad (11.4.33)$$

The resulting algorithm is given below.

Algorithm 11.10. *Bi-CGS Algorithm.*

Set $r_0 = b - Ax_0$ and choose \tilde{r}_0 so that $(r_0, \tilde{r}_0) \neq 0$.

```

 $p_0 = u_0 = r_0; \rho_0 = (\tilde{r}_0, r_0);$ 
for  $j = 0, 1, 2, \dots$ 
     $v_j = Ap_j; \alpha_j = \rho_j / (\tilde{r}_0, v_j);$ 
     $q_j = u_j - \alpha_j v_j;$ 
     $d_j = u_j + q_j;$ 
     $x_{j+1} = x_j + \alpha_j d_j;$ 
     $r_{j+1} = r_j - \alpha_j A d_j;$ 
     $\rho_{j+1} = (\tilde{r}_0, r_{j+1});$ 
     $\beta_j = \rho_{j+1} / \rho_j;$ 
     $u_{j+1} = r_{j+1} + \beta_j q_j;$ 
     $p_{j+1} = u_{j+1} + \beta_j (q_j + \beta_j p_j);$ 
end
```

There are now two matrix-vector multiplications with A in each step. When Bi-CG converges well we can expect Bi-CGS to converge about twice as fast.

Although the Bi-CGS algorithm often is competitive with other methods such as GMRES, a weak point of Bi-CGS is that the residual norms may behave very

erratically, in particular when the iteration is started close to the solution. For example, although the norm of the vector $\psi_j(A)r_0$ is small it may happen that $\|\psi_j^2(A)r_0\|$ is much bigger than $\|r_0\|$. This may even lead to such severe cancellation that the accuracy of the computed solution is destroyed.

This problem motivated the development a stabilized version called Bi-CGSTAB by van der Vorst [157]), which is more smoothly converging. Instead of computing the residuals $\psi_j^2(A)r_0$, this algorithm uses

$$r_j = \chi_j(A)\psi_j(A)r_0, \quad \chi_j(t) = (1 - \omega_1 t)(1 - \omega_2 t) \cdots (1 - \omega_j t), \quad (11.4.34)$$

where the constants ω_j are determined so that $\|r_j\|_2$ is minimized as a function of ω_j . From the orthogonality property $(\psi_i(A)r_0, \chi_j(A)r_0) = 0$, for $j < i$, it follows that Bi-CGSTAB is a finite method, i.e. in exact arithmetic it will converge in at most n steps. As for Bi-CGS it requires two matrix-vector products with A .

Algorithm 11.11. Bi-CGSTAB Algorithm.

Let x_0 be an initial guess, $r_0 = b - Ax_0$ and choose \tilde{r}_0 so that $(\tilde{r}_0, r_0) \neq 0$.

```

 $p_0 = u_0 = r_0; \quad \rho_0 = (\tilde{r}_0, r_0);$ 
for  $j = 0, 1, 2, \dots$ 
     $v_j = Ap_j; \quad \alpha_j = \rho_j / (\tilde{r}_0, v_j);$ 
     $s_j = r_j - \alpha_j v_j;$ 
     $t_j = Ap_j;$ 
     $\omega_j = (t_j, s_j) / (t_j, t_j);$ 
     $q_j = u_j - \alpha_j v_j;$ 
     $d_j = u_j + q_j;$ 
     $x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j;$ 
     $r_{j+1} = s_j - \omega_j t_j;$ 
     $\rho_{j+1} = (\tilde{r}_0, r_{j+1});$ 
     $\beta_j = (\rho_{j+1} / \rho_j)(\alpha_j / \omega_j);$ 
     $p_{j+1} = r_{j+1} + \beta_j(p_j - \omega_j v_j);$ 
end

```

Review Questions

- 4.1 What optimality property does the residual vectors $r_k = b - Ax_k$ in the GMRES method satisfy. In what subspace does the vector $r_k - r_0$ lie?
- 4.2 Name two iterative methods that are suitable for solving general unsymmetric linear systems.

- 4.3** In Lanczos bi-orthogonalization bases for two different Krylov subspaces are computed. Which are the Krylov subspaces? What property does these bases have?
- 4.4** What are the main advantages and drawbacks of the Bi-CG method compared to GMRES.
- 4.5** How are the approximations x_k defined in QMR?

Problems and Computer Exercises

- 4.1** Consider using GMRES to solve the system $Ax = b$, where

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

using $x_0 = 0$. Show that $x_1 = 0$, and that therefore GMRES(1) will never produce a solution.

11.5 Preconditioned Iterative Methods

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 more efficiently be solved. Preconditioners have been widely used together with the krylov subspace methods since the early 1970s. General purpose techniques for constructing preconditioners have made iterative methods successful in many industrial applications.

The rate of convergence of iterative methods depends, often in a complicated way on the eigenvalue distribution and of the matrix and the projection of the initial residual vectors onto the eigenvectors. Often convergence is slow and can even stagnate. In order to be effective the iterative method must often be applied to a preconditioned system

$$M^{-1}Ax = M^{-1}b. \quad (11.5.1)$$

The idea is to choose M so that the rate of convergence of the iterative method is improved. In some sense M should be chosen as an approximation to A . The rational behind this is that if $M = A$ the transformed system simply becomes $x = A^{-1}b$. The system in (11.5.1) is said to be *left-preconditioned*. We can also consider the *right-preconditioned* system

$$AM^{-1}u = b, \quad u = Mx. \quad (11.5.2)$$

Note that the spectrum of $M^{-1}A$ and AM^{-1} are the same. One difference between these two approaches is that in the right-preconditioned case the actual residual norm is available

Note that in iterative methods the product $M^{-1}A$ (or AM^{-1}) need never be formed. The preconditioned iteration is instead implemented by forming matrix vector products with A and M^{-1} separately. Since forming $u = M^{-1}v$ for an arbitrary vector v is equivalent to solving a linear system $Mu = v$, the inverse M^{-1} is not needed either. If the system $Mu = v$ is solved by a direct method, the preconditioned iterative method can be viewed as a compromise between a direct and iterative solution method. There are also preconditioners which are formed by constructing a sparse approximations $C = A^{-1}$. Then the iterative method is applied to the system $CAx = Cb$ and Cv is formed by matrix-vector multiplication.

To be useful a preconditioner should satisfy the following conditions:

- (i) The residual norm $\|M^{-1}A - I\|$ (or $\|AM^{-1} - I\|$) is small.
- (ii) Linear systems of the form $Mu = v$ should be easy to solve.

Condition (i) implies fast convergence, (ii) that the arithmetic cost of preconditioning is reasonable. Obviously these conditions are contradictory and a compromise must be sought. For example, taking $M = A$ 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 $M^{-1}A$ cluster around another number than 1.

If A is symmetric, positive definite, the preconditioned system should also have this property. To ensure this we take M symmetric, positive definite and consider a **split preconditioner**. Let $M = LL^T$ where L is the Cholesky factor of M , and set $\tilde{A}\tilde{x} = \tilde{b}$, where

$$\tilde{A} = L^{-1}AL^{-T}, \quad \tilde{x} = L^T x, \quad \tilde{b} = L^{-1}b. \quad (11.5.3)$$

Then \tilde{A} is symmetric positive definite. Note that the spectrum of $A = L^{-1}AL^{-T}$ is the same as for $L^{-T}L^{-1}A = M^{-1}A$.

The choice of preconditioner is strongly problem dependent and possibly the most crucial component in the success of an iterative method. There is no general theory that can be used. Often the selection of a preconditioner for a given class of problems is an educated guess based on trial and error. A preconditioner, which is expensive to compute, may become viable if it is to be used many times. This may be the case, e.g., when dealing with time-dependent or nonlinear problems. The choice is also dependent on the architecture of the computing system. Preconditioners that are efficient in a scalar computing environment may show poor performance on vector and parallel machines. It should be remembered that the goal always is to reduce the CPU time (or storage requirement) for the computation.

In choosing a preconditioner one should try to take advantage of the structure of the problem. Suppose that a large linear system comes from the discretization of a partial differential equation on a fine grid. Then a very efficient preconditioner may be obtained by taking M to be the product of three operations: restriction of the fine grid approximation to a coarse grid, solving the problem on the coarse grid, and the transfer from the coarse grid to the fine grid by interpolation. When repeated on several grids, this technique is equivalent to **multigrid iteration**.

Another alternative, when a high order discrete approximation has been used, is to let M correspond to a lower order approximation.

More generally, if the matrix can be split as $A = A_1 + A_2$, where systems with A_1 are easy to solve, then one can try taking $M = A_1$. Note that this idea is related to **domain decomposition**, where neglecting the operator A_2 decomposes a problem to a set of smaller problems on subdomains.

In some problems the elements in A represent a coupling between elements of the solution. The elements can represent particles or other physical objects. In this case a preconditioner M may be derived from A by omitting small or long range interactions. In simple cases this means letting M consist of a few diagonals of A near the main diagonal.

11.5.1 Preconditioned Krylov Subspace Methods

The CG algorithm can be applied to linear systems $Ax = b$, where A is symmetric positive definite. In this case the preconditioner M should also chosen to be symmetric positive definite. Since $M^{-1}A$ and AM^{-1} are not symmetric in general, we cannot apply the CG method directly to the left- or right-preconditioned system. One solution is to use a split preconditioner and apply CG to the system (11.5.3). In implementing the preconditioned CG method we need to perform in each step the operations

$$L^T q = p, \quad s = Aq, \quad Lt = s.$$

The extra work per step in using the preconditioner is to solve two triangular linear systems. The split preconditioned CG algorithm will have recursions for the transformed variables and residuals vectors $\tilde{x} = L^T x$ and $\tilde{r} = L^{-1}(b - Ax)$. It can be simplified by reformulating it in terms of the original variables x and residual $r = b - Ax$. It is left as an exercise to show that if we let $p_k = L^{-T}\tilde{p}_k$, $z_k = L^{-T}\tilde{r}_k$, and

$$M = LL^T, \tag{11.5.4}$$

the following implementation of the preconditioned CG (PCG) method is obtained:

Algorithm 11.12. PCG Method.

```

function x = cgsolve(A,M,b,x0,tol);
% Solve Ax = b by the CG method.
% Iterate while ||Ax - b|| > tol*||b||.
%
x = x0; r = b - A*x;
z = M\r; p = z;
ztr = z'*r;
while (norm(r) > tol*norm(b))
    w = A*p;
    alpha = ztr/(p'*w);
    x = x + alpha*p;
    r = r - alpha*w;
end

```

```

z = M\r;
ztrold = rtr;
ztr = z'*r;
beta = ztr/ztrold;
p = z + beta*p;
end

```

A surprising and important feature of this version is that it depends only on the symmetric positive definite matrix $M = LL^T$. Hence, the factored form of M is not needed and the back transformation of the solution.

Since

$$\|\tilde{x} - \tilde{x}_k\|_{\tilde{A}}^2 = (\tilde{x} - \tilde{x}_k)^T L^{-1} A L^{-T} (\tilde{x} - \tilde{x}_k) = \|x - x_k\|_A^2,$$

we are still minimizing the A -norm of the error in x , although now over a Krylov subspace $\mathcal{K}_k(M^{-1}A, M^{-1}r_0)$. The rate of convergence depends on $\kappa(\tilde{A})$. The MATLAB program above works also for positive definite Hermitian systems. From the convergence analysis in Section 11.2.4 we know that the preconditioned CG method will have rapid convergence if one or both of the following conditions are satisfied:

- i. The condition number of $M^{-1}A$ is small , or
- ii. $M^{-1}A$ has only a few distinct clusters of eigenvalues.

For symmetric indefinite systems SYMMLQ can be combined with a positive definite preconditioner M . To solve the symmetric indefinite system $Ax = b$ the preconditioner is regarded to have the form $M = LL^T$ and SYMMLQ *implicitly* applied to the system

$$L^{-1}AL^{-T}w = L^{-1}b.$$

The algorithm accumulates approximations to the solution $x = L^{-T}w$, without approximating w . A MATLAB implementation of this algorithm, which only requires solves with M , is given by Gill et al. [253].

Preconditioned CGLS and CGME.

We now develop preconditioned versions of the least squares CG algorithms given in Section 11.3.3. To precondition CGLS Algorithm (11.3.3) it is natural to use a right preconditioner $M \in \mathbf{R}^{n \times n}$, i.e., perform the transformation of variables $Mx = y$

$$\min_y \|(AM^{-1})y - b\|_2. \quad (11.5.5)$$

(Note that for a inconsistent system $Ax = b$ a left preconditioner would change the problem.) It is straightforward to apply Algorithm CGLS to (11.5.5). Note that the algorithm can be formulated in terms of the original variables x instead of $y = Mx$.

Algorithm 11.13. PCGLS Method.

```

function [x,r] = pcccls(A,M,b,x0,eps);
% Solve the normal equation A^H Ax = A^H b with
% the CGLS method and right preconditioner M.
x = x0; r = b - A*x;
s = M'\(A'*r); p = s;
sts = s'*s;
while (norm(r) > eps)
    t = M\p; q = A*t;
    alpha = sts/(q'*q);
    x = x + alpha*t;
    r = r - alpha*q;
    s = M'(A'*r);
    stsold = sts; sts = s'*s;
    beta = sts/stsold;
    p = s + beta*p;
end

```

Preconditioned CGLS minimizes the same error functional $\|\hat{r} - r_k\|_2^2$, but over the shifted Krylov subspace

$$x_k = x_0 + \mathcal{K}_k, \quad \mathcal{K}_k = ((M^H M)^{-1} A^H A, (M^H M)^{-1} A^H r_0). \quad (11.5.6)$$

An upper bound on the rate of convergence is given by

$$\|r - r_k\|_2 < 2 \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|r - r_0\|_2, \quad \kappa = \kappa(AM^{-1}).$$

For computing the minimum norm solution of a consistent systems a preconditioned version of CGME. Here it is natural to use a left preconditioner M , and apply CGME to the problem

$$\min \|x\|_2 \quad \text{such that} \quad M^{-1}Ax = M^{-1}b.$$

In this algorithm the residual vectors are transformed. If it is formulated in terms of the original residuals, the following algorithm results:

Algorithm 11.14. PCGME Method.

```

function [x,r] = pccgme(A,M,b,x0,eps);
% Solve x = A^H y, AA^H y = b by the CGME method.
% with left preconditioner M.
x = x0; r = b - A*x;
s = M\r; p = s; sts = s'*s;
while (norm(r) > eps)
    q = A'*(M'\p);
    alpha = sts/(q'*q);

```

```

x = x + alpha*q;
r = r - alpha*(A*q);
s = M\r;
stsold = sts; sts = s'*s;
beta = sts/stsold;
p = s + beta*p;

```

Preconditioned CGME minimizes the error functional $\|x - x_k\|_2^2$, over the Krylov subspace

$$x_k = x_0 + \mathcal{K}_k, \quad \mathcal{K}_k = (A^H(MM^H)^{-1}A, A^H(MM^H)^{-1}r_0). \quad (11.5.7)$$

An upper bound on the rate of convergence is given by

$$\|x - x^{(k)}\|_2 < 2 \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|x - x_0\|_2, \quad \kappa = \kappa(M^{-1}A).$$

Preconditioned GMRES

For unsymmetric linear systems there are two options for applying the preconditioner. We can either use the left preconditioned system (11.5.1) or the right preconditioned system (11.5.2). (If A is almost symmetric positive definite, then a split preconditioner might also be considered.) The resulting changes needed in the GMRES algorithm are quite small.

In the case of using a left preconditioner M only the following changes in the Arnoldi algorithm are needed. We start the recursion with

$$r_0 = M^{-1}(b - Ax_0), \quad \beta_1 = \|r_0\|_2; \quad v_1 = r_0/\beta_1,$$

and define

$$z_j = M^{-1}Av_j, \quad j = 1 : k.$$

All computed residual vectors will be preconditioned residuals $M^{-1}(b - Ax_k)$. This is a disadvantage since most stopping criteria use the actual residuals $r_m = b - Ax_k$. In this left preconditioned version the transformed residual norm $\|M^{-1}(b - Ax)\|_2$ will be minimized among all vectors of the form

$$x_0 + \mathcal{K}_m(M^{-1}A, M^{-1}r_0). \quad (11.5.8)$$

In the right preconditioned version of GMRES the actual residual vectors are used, but the variables are transformed according to $u = Mx$ ($x = M^{-1}u$). The right preconditioned algorithm can easily be modified to give the untransformed solution. We have

$$z_j = AM^{-1}v_j, \quad j = 1 : k.$$

The k th approximation is $x_k = x_0 + M^{-1}V_ky_k$, where y_k solves

$$\min_{y_k} \|\beta_1 e_1 - \hat{H}_k y_k\|_2.$$

As before this can be written as

$$x_k = x_{k-1} + \beta_1 \tau_k M_k^{-1} w_k, \quad w_k = R_k y_k,$$

see (11.3.42). In this version the residual norm $\|b - AM^{-1}u\|_2$ will be minimized among all vectors of the form $u_0 + \mathcal{K}_m(AM^{-1}, r_0)$. However, this is equivalent to minimizing $\|b - Ax\|_2$ among all vectors of the form

$$x_0 + M^{-1}\mathcal{K}_m(AM^{-1}, r_0). \quad (11.5.9)$$

Somewhat surprisingly the two affine subspaces (11.5.8) and (11.5.9) are the same! The j th vector in the two Krylov subspaces are $w_j = (M^{-1}A)^j M^{-1}r_0$ and $\tilde{w}_j = M^{-1}(AM^{-1})^j r_0$. By a simple induction proof it can be shown that $M^{-1}(AM^{-1})^j = (M^{-1}A)^j M^{-1}$. This is clearly true for $j = 1$. Further, the induction step follows from

$$\begin{aligned} (M^{-1}A)^{j+1}M^{-1} &= M^{-1}A(M^{-1}A)^j M^{-1} = M^{-1}AM^{-1}(AM^{-1})^j \\ &= M^{-1}(AM^{-1})^{j+1}. \end{aligned}$$

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.

For the case when A is diagonalizable, $A = X\Lambda X^{-1}$, where $\Lambda = \text{diag}(\lambda_i)$ we proved the error estimate

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa_2(X) \min_{q_k} \max_{i=1,2,\dots,n} |q_k(\lambda_i)|, \quad (11.5.10)$$

where q_k is a polynomial of degree $\leq k$ and $q_k(0) = 1$. Because of the factor $\kappa_2(X)$ in (11.5.10) the rate of convergence can no longer be deduced from the spectrum $\{\lambda_i\}$ of the matrix A alone. Since the spectrum of $M^{-1}A$ and AM^{-1} are the same we can expect the convergence behavior to be similar if A is close to normal.

11.5.2 Preconditioners from Matrix Splittings

The stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \quad k = 0, 1, \dots, \quad (11.5.11)$$

corresponds to a matrix splitting $A = M - N$, and the iteration matrix

$$B = M^{-1}N = I - M^{-1}A.$$

The iteration (11.5.11) can be considered as a fixed point iteration applied to the preconditioned system $M^{-1}Ax = M^{-1}b$. Hence, the basic iterative methods considered in Section 11.1.4 give simple examples of preconditioners.

The Jacobi and Gauss–Seidel methods are both special cases of one-step stationary iterative methods. Using the standard splitting $A = D_A - L_A - U_A$, where

D_A is diagonal, L_A and U_A are strictly lower and upper triangular, these methods correspond to the matrix splittings

$$M_J = D_A, \quad \text{and} \quad M_{GS} = D_A - L_A.$$

If A is symmetric positive definite then $M_J = D_A > 0$ and symmetric. However, M_{GS} is lower triangular and unsymmetric.

The simplest choice related to this splitting is to take $M = D_A$. This corresponds to a diagonal scaling of the rows of A , such that the scaled matrix $M^{-1}A = D_A^{-1}A$ has a unit diagonal. For s.p.d. matrices symmetry can be preserved by using a split preconditioner with $L = L^T = D_A^{1/2}$. In this case it can be shown that this is close to the optimal diagonal preconditioning.

Lemma 11.5.1. Van der Sluis [582]

Let $A \in \mathbf{R}^{n \times n}$ be a symmetric positive definite matrix with at most $q \leq n$ nonzero elements in any row. Then if $A = D_A - L_A - L_A^T$, it holds that

$$\kappa(D_A^{-1/2} A D_A^{-1/2}) = q \min_{D > 0} \kappa(DAD).$$

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 Section 11.1.5 it was shown that for a symmetric matrix A the SSOR iteration method corresponds to a splitting with the matrix

$$M_{SSOR} = \frac{1}{\omega(2-\omega)} (D_A - \omega L_A) D_A^{-1} (D_A - \omega U_A).$$

Since M_{SSOR} is given in the form of an LDL^T factorization it is easy to solve linear systems when using this as a preconditioner. It also has the same sparsity as the original matrix A and for $0 < \omega < 2$, if A is s.p.d., so is M_{SSOR} . The performance of the SSOR splitting turns out to be fairly insensitive to the choice of ω . For systems arising from second order boundary values problems, e.g., the model problem studied previously, the original condition number $\kappa(A) = \mathcal{O}(h^{-2})$ can be reduced to $\kappa(M^{-1}A) = \mathcal{O}(h^{-1})$. Taking $\omega = 1$ is often close to optimal. This corresponds to the symmetric Gauss–Seidel (SGS) preconditioner

$$M_{SGS} = LU, \quad L = (I - L_A D_A^{-1}), \quad U = (D_A - L_A^T).$$

The SGS preconditioner has the form $M_{SGS} = LU$ where L is lower triangular and U is upper triangular. To find out how well M_{SGS} approximates A we form the defect matrix

$$A - LU = D_A - L_A - U_A - (I - L_A D_A^{-1})(D_A - U_A) = -L_A D_A^{-1} U_A.$$

The application of all the above preconditioners involves only triangular solves and multiplication with a diagonal matrix. They are all defined in terms of elements of the original matrix A , and hence do not require extra storage.

11.5.3 Incomplete Factorization Methods

An interesting question is whether we can find matrices L and U with the same nonzero structure as for M_{SGS} , but with a smaller defect matrix $R = LU - A$. This question leads to the development of an important class of preconditioners 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 $R = A - LU$ is small. For symmetric positive definite matrices the same concept applies to the Cholesky factorization and gives **incomplete Cholesky IC** factorizations.

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 matrices*. Assume that these places (i, j) are 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 always are included in \mathcal{P} . For example, we could take $\mathcal{P} = \mathcal{P}_A$, the set of nonzero elements in A .

Algorithm 11.15. *Incomplete LU Factorization.*

```

for  $k = 1 : n - 1$ 
  for  $i = k + 1, \dots, n$ 
    if  $(i, k) \in \mathcal{P}$   $l_{ik} = a_{ik}/a_{kk};$ 
    for  $j = k + 1, \dots, n$ 
      if  $(k, j) \in \mathcal{P}$   $a_{ij} = a_{ij} - l_{ik}a_{kj};$ 
    end
  end
end

```

The elimination consists of $n - 1$ steps. In the k th step we first subtract from the current matrix elements with indices (i, k) and $(k, i) \notin \mathcal{P}$ and place 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 notice that since the first $m - 1$ rows of R_m are zero it follows that $L_k R_m = R_m$, if $k < m$. Then by combining the above equations we find $LU = A + R$, where

$$U = A_{n-1}, \quad L = (L_{n-1} L_{n-2} \cdots L_1)^{-1}, \quad R = R_1 + R_2 + \cdots + R_{n-1}.$$

Algorithm 10.8.1 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 and can also be included in the defect matrix R_k . This is achieved simply by changing line five in the algorithm to

$$\text{if } (k, j) \in \mathcal{P} \text{ and } (i, j) \in \mathcal{P} \text{ } a_{ij} = a_{ij} - l_{ik} a_{kj};$$

In practice the matrix A is sparse and the algorithm should be specialized to take this into account. In particular, a version where A is processed a row at a time is more convenient for general sparse matrices. Such an algorithm can be derived by interchanging the k and i loops in Algorithm 10.8.1.

Example 11.5.1.

For the model problem using 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 $A = LU + R$, where

$$L = L_{-n} + L_{-1} + L_0, \quad U = U_0 + U_1 + U_n,$$

where L_{-k} (and U_k) denote matrices with nonzero elements only in the k -th lower (upper) diagonal. By the rule for multiplication by diagonals (see Lemma 7.4.3)

$$A_k B_l = C_{k+l}, \text{ if } k + l \leq n - 1,$$

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 are 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 [296]. 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 $R = LL^T - A$. Higher level incomplete factorizations are defined in a similar way, and so on. ILU(0) preconditioners are simple to implement and quite effective for problems such as low-order discretizations of elliptic partial differential equations. 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 construction and application of these.

An incomplete LU factorization may not exist even if A is nonsingular and has an LU factorization. However, for some more restricted classes of matrices the existence of incomplete factorizations can be guaranteed. The following important result was proved by Meijerink and van der Vorst [431, 1977].

Theorem 11.5.2.

If A is an M -matrix, there exists for every set \mathcal{P} such that $(i, j) \in \mathcal{P}$ for $i = j$, 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.

In case A is s.p.d., an **incomplete Cholesky factorization** can be used as preconditioner. In this the nonzero set \mathcal{P} is assumed to be symmetric, i.e., if $(i, j) \in \mathcal{P}$ then also $(j, i) \in \mathcal{P}$. Positive definiteness of A alone is not sufficient to guarantee the existence of an incomplete Cholesky factorization, because zero elements may occur on the diagonal during the factorization.

For the case when A is a symmetric M -matrix, a variant of the above theorem guarantees the existence for each symmetric set \mathcal{P} such that $(i, j) \in \mathcal{P}$ for $i = j$, a uniquely defined lower triangular matrix L , with $l_{ij} = 0$ if $(i, j) \notin \mathcal{P}$ such that the splitting $A = LL^T - R$ is regular.

A description of the incomplete Cholesky factorization in the general case is given below.

Algorithm 11.16. IC 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
            
$$l_{ij} = \left( a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right)$$

        end
    end

```

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 since 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 proposed by Saad [517]. This consists of using a threshold τ but also limiting the number of nonzeros allowed in each row of the triangular factors to p . First all fill-in which are smaller than τ times the 2-norm of the current row. Of the remaining ones keep only the p largest. The resulting preconditioner is called ILUT(τ, p). This strategy applies also to the IC factorization.

Jennings and Ajiz [352] were the first to propose an preconditioner based on an incomplete QR factorization. Later work on this topic include papers by Wang, Gallivan and Bramley [599] and Bai, Papadoupolos, Duff and Whaten [27, 474]. MIQR is a multilevel incomplete QR preconditioner for large sparse least squares problems developed by Na and Saad nasa:06

11.5.4 Sparse Approximate Inverses

Although successful in many applications incomplete have disadvantages. The difficulty in parallelization the back substitutions needed in applying the preconditioner limits their use. This has motivated the development of preconditioners based on a sparse approximation of the inverse A^{-1} . The application of the preconditioner is then a matrix-vector operation that is more amenable to parallelization.

It is not at all obvious that it is possible to find a sparse matrix M which 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 Section 7.4.5. However, if A is a banded symmetric positive definite matrix, then a classical result (see [143]) 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 diagonal 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 very efficient preconditioner.

We now discuss a method to compute a sparse matrix M which approximately minimizes the Frobenius norm of the error matrix $I - AM$. We first note that

$$\|I - AM\|_F^2 = \sum_{k=1}^n \|e_k - Am_k\|_2^2 \quad (11.5.12)$$

where m_k is the k th column of M . Therefore, the problem decouples into n inde-

pendent least squares problems

$$\|e_k - Am_k\|_2^2, \quad k = 1 : n, \quad (11.5.13)$$

which can be solved in parallel. If m_k is sparse, then (11.5.13) only involves a few columns of A . Further, we can compress A by deleting all rows which are identically zero in the selected columns. The result is a small least squares problems that can be solved by computing the 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 [286] 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 the SPAI approach is that even in A is symmetric M will be unsymmetric in general. A natural remedy is to use the symmetrized preconditioner $\frac{1}{2}(A + A^T)$ instead of M . A more direct way is to seek an approximate sparse inverse in factored form. We first note that if A admits a factorization $A = LDU$, with L and U nonsingular triangular matrices, then $L^{-1}AU^{-1} = D$. We seek upper triangular matrices $Z \approx U^{-1}$ and $W \approx L^{-T}$ to minimize

$$\|I - W^T AZ\|_F^2, \quad (11.5.14)$$

In the case that A is symmetric case $W = Z$. This corresponds to a symmetric positive definite preconditioner $M = Z^T Z$ when Z is nonsingular. Since Z is triangular this is the case when all diagonal elements of Z are nonzero. This approach was introduced by Kolotilina and Yeremin [383] and is known as the **FSAI** algorithm. The motivation behind its development was to obtain a symmetric positive definite preconditioner, which can be used with the CG method.

In general, if the matrices $W = (w_1, \dots, w_n)$ and $Z = (z_1, \dots, z_n)$ satisfy the relation $W^T AZ = D$, the column vectors are A-biconjugate, $w_i^T Az_i = 0$, $i \neq j$, and

$$A^{-1} = ZD^{-1}W^T = \sum_{i=1}^n \frac{z_i w_i^T}{d_i}. \quad (11.5.15)$$

A direct projection method given by Benzi and Meyer [48] can be used to compute A-biconjugate matrices Z and W satisfying (11.5.15). The procedure to compute Z can be written as follows, where a_i^T denotes the i th row of A :

```

 $z_1^{(0)} = e_1; \quad p_1^{(0)} = a_{11};$ 
for  $i = 2 : n$ 
   $z_i^{(0)} = e_i;$ 
  for  $j = 1 : i - 1$ 
     $p_i^{(j-1)} = a_j^T z_i^{(j-1)};$ 
     $z_i^{(j)} = z_i^{(j-1)} - (p_i^{(j-1)} / p_j^{(j-1)}) z_j^{(j-1)};$ 
  end
   $p_i^{(i-1)} = a_i^T z_i^{(i-1)};$ 
end

```

As shown in [48] the matrix Z computed by this algorithm is unit upper triangular and such that $L = AZ$ is lower triangular. By uniqueness it follows that $A = LZ^{-1}$ is the Crout factorization of A . It follows that 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 the unsymmetric case the matrix $W = L^{-1}$ is computed identically, except that a_i^T is replaced by c_i^T , where c_i is the i th column of A . The algorithm can also be interpreted as a generalized Gram–Schmidt orthogonalization process with respect to the bilinear form associated with A .

By dropping elements in Z and W according to a drop tolerance in the above process, Benzi and Tøuma [49] developed the **AINV** algorithm for constructing an approximate inverse. Incompleteness can also be imposed by enforcing a prescribed nonzero structure on Z and W .

11.5.5 Block Tridiagonal Matrices

Many matrices arising from the discretization of multidimensional problems have a block structure. For such matrices one can develop **block incomplete factorizations**. In particular, we consider here symmetric positive definite block tridiagonal matrices of the form

$$A = \begin{pmatrix} D_1 & A_2^T \\ A_2 & D_2 & A_3^T \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots & A_N^T \\ & & & A_N & D_N \end{pmatrix} = D_A - L_A - L_A^T, \quad (11.5.16)$$

with square diagonal blocks D_i . For the model problem with the natural ordering of mesh points we obtain this form with $A_i = -I$, $D_i = \text{tridiag}(-1 \ 4 \ -1)$. If systems with D_i can be solved efficiently a simple choice of preconditioner is the block diagonal preconditioner

$$M = \text{diag}(D_1, D_2, \dots, D_N).$$

The case $N = 2$ is of special interest. For the system

$$\begin{pmatrix} D_1 & A_2^T \\ A_2 & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad (11.5.17)$$

the **block diagonal preconditioner** gives a preconditioned matrix of the form

$$M^{-1}A = \begin{pmatrix} I & D_1^{-1}A_2^T \\ D_2^{-1}A_2 & I \end{pmatrix}.$$

Note that this matrix is of the form (11.1.30) and therefore has property A. Suppose the CG method is used with this preconditioner and a starting approximation $x_1^{(0)}$. If we set

$$x_2^{(0)} = D_2^{-1}(b_2 - A_2x_1^{(0)}),$$

then the corresponding 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 will 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

If we eliminate x_1 in the system (11.5.17) then we obtain

$$Sx_2 = b_2 - A_2 D_1^{-1} b_1, \quad S = D_2 - A_2 D_1^{-1} A_2^T, \quad (11.5.18)$$

where S is the Schur complement of D_1 in A . If A is s.p.d., then S is also s.p.d., and hence the conjugate gradient can be used to solve the system (11.5.18). This process is called **Schur complement preconditioning**. Here it is not necessary to form the Schur complement S , since we only need the effect of S on vectors. We can save some computations by writing the residual of the system (11.5.18) as

$$r_2 = (b_2 - D_2 x_2) - A_2 D_1^{-1} (b_1 - A_2^T x_2).$$

Note here 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 in each step by an iterative method in an **inner iteration**.

We now describe a **block incomplete Cholesky factorization** due to Concus, Golub and Meurant [115], which has proved to be very useful. We assume in the following that in (11.5.16) D_i is tridiagonal and A_i is diagonal, as in the model problem. First recall from Section 7.6.2 that the exact block Cholesky factorization of a symmetric positive definite block-tridiagonal matrix can be written as

$$A = (\Sigma + L_A) \Sigma^{-1} (\Sigma + L_A^T),$$

where L_A 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^{-1} and hence Σ_i , $i \geq 2$, are dense. Because of this the exact block Cholesky factorization is not useful.

Instead we consider computing an incomplete block factorization from

$$\Delta_1 = D_1, \quad \Delta_i = D_i - A_i \Lambda_{i-1}^{-1} A_i^T, \quad i = 2 : N. \quad (11.5.19)$$

Here, for each i , Λ_{i-1} is a sparse approximation to Δ_{i-1} . The incomplete block Cholesky factorization is then

$$M = (\Delta + L_A) \Delta^{-1} (\Delta + L_A^T), \quad \Delta = \text{diag}(\Delta_1, \dots, \Delta_n).$$

The corresponding defect matrix is $R = M - A = \text{diag}(R_1, \dots, R_n)$, 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

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix}$$

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

$$L = \begin{pmatrix} \gamma_1 & & & & \\ \delta_1 & \gamma_2 & & & \\ & \delta_2 & \ddots & & \\ & & \ddots & \gamma_{n-1} & \\ & & & \delta_{n-1} & \gamma_n \end{pmatrix}.$$

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 matrix L^{-1} , which is lower triangular and dense, is approximated by a banded lower triangular matrix $L^{-1}(p)$, taking only the first $p + 1$ lower diagonals of the exact L^{-1} . Note that elements of the matrix L^{-1}

$$L^{-1} = \begin{pmatrix} 1/\gamma_1 & & & & \\ \zeta_1 & 1/\gamma_2 & & & \\ \eta_1 & \zeta_2 & \ddots & & \\ \vdots & \ddots & \ddots & 1/\gamma_{n-1} & \\ \cdots & \eta_{n-2} & \zeta_{n-1} & 1/\gamma_n & \end{pmatrix},$$

can be computed diagonal by diagonal. For example, we have

$$\zeta_i = \frac{\delta_i}{\gamma_i \gamma_{i+1}}, \quad i = 2 : n - 1.$$

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 (11.5.19) will in this case be tridiagonal.

11.5.6 Constant Coefficient Approximation

For the solution of discretizations of some elliptic problems on a rectangular domain fast direct methods can be developed, provided that some strong assumptions about the regularity of the system are satisfied. It applies only to discretizations of problems with constant coefficients on a rectangular domain. However, if we

have variable coefficients the fast solver may be used as a preconditioner in the CG method. Similarly, problems on an irregular domain may be embedded in a rectangular domain, and again a preconditioner based on a fast solver may be used.

Consider a linear system $Ax = b$ where A has the tridiagonal block-Toeplitz form

$$A = \begin{pmatrix} B & T & & \\ T & B & T & \\ & T & B & \ddots \\ & & \ddots & \ddots & T \\ & & & T & B \end{pmatrix} \in \mathbf{R}^{n \times n}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{pmatrix}, \quad (11.5.20)$$

where $n = pq$ and $B, T \in \mathbf{R}^{p \times p}$ are symmetric matrices. We assume that B and T commute, $BT = TB$. Then B and T share a common system of eigenvectors, and we let

$$Q^T B Q = \Lambda = \text{diag}(\lambda_i), \quad Q^T T Q = \Omega = \text{diag}(\omega_i).$$

If we make the change of variables

$$x_j = Qz_j, \quad y_j = Q^T b_j, \quad j = 1 : q,$$

the system $Ax = b$ can be written $Cz = y$, where

$$C = \begin{pmatrix} \Lambda & \Omega & & \\ \Omega & \Lambda & \Omega & \\ & \Omega & \Lambda & \ddots \\ & & \ddots & \ddots & \Omega \\ & & & \Omega & \Lambda \end{pmatrix},$$

with Λ and Ω diagonal matrices.

For the model problem with Dirichlet boundary conditions the eigenvalues and eigenvectors are known (see Section 11.1.2). Furthermore, the multiplication of a vector by Q and Q^T is efficiently obtained by the FFT algorithm. This allows a fast method for the model problem to be developed that uses a total of only $\mathcal{O}(n^2 \log n)$ operations to compute the n^2 unknown components of x .

Algorithm 11.17.

Fast method using FFT for the model problem (11.5.20).

1. Compute

$$y_j = Q^T b_j, \quad j = 1 : q.$$

2. Rearrange taking one element from each vectors y_j ,

$$\hat{y}_i = (y_{i1}, y_{i2}, \dots, y_{iq})^T, \quad i = 1 : p,$$

and solve by elimination the p systems

$$\Gamma_i \hat{z}_i = \hat{y}_i, \quad i = 1 : p,$$

where

$$\Gamma_i = \begin{pmatrix} \lambda_i & \omega_i & & & \\ \omega_i & \lambda_i & \omega_i & & \\ & \omega_i & \lambda_i & \ddots & \\ & & \ddots & \ddots & \omega_i \\ & & & \omega_i & \lambda_i \end{pmatrix}, \quad i = 1 : p.$$

3. Rearrange (inversely to step 2) taking one element from each \hat{z}_i ,

$$z_j = (\hat{z}_{j1}, \hat{z}_{j2}, \dots, \hat{z}_{jq})^T, \quad j = 1 : q,$$

and compute

$$x_j = Qz_j, \quad j = 1 : q.$$

The fast Fourier transforms in steps 1 and 3 take $\mathcal{O}(n^2 \log n)$ operations. Solving the tridiagonal systems in step 2 only takes $\mathcal{O}(n^2)$ operations, and hence for this step Gaussian elimination is superior to FFT.

11.5.7 Preconditioners for Toeplitz Systems

Boundary conditions are 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 when this approach works well.

A **Toeplitz matrix** $T = (t_{i-j})_{1 \leq i,j \leq n}$ is a matrix whose entries are constant along each diagonal;

$$T_n = \begin{pmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_{-1} & t_0 & \dots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{-n+1} & t_{-n+2} & \dots & t_0 \end{pmatrix} \in \mathbf{R}^{n \times n}. \quad (11.5.21)$$

T_n is defined by the $2n - 1$ values of $t_{-n+1}, \dots, t_0, \dots, t_{n-1}$. The Toeplitz structure implies that the matrix-vector product Tx for a given vector x reduces to a convolution problem, and can be computed via the fast Fourier transform in $\mathcal{O}(n \log n)$ operations. This is true also when T is a rectangular Toeplitz matrix; see O'Leary and Simmons [455, 1981].

The special Toeplitz matrix

$$C_n = \begin{pmatrix} c_0 & c_1 & c_2 & \dots & c_{n-1} \\ c_{n-1} & c_0 & c_1 & \dots & c_{n-2} \\ c_{n-2} & c_{n-1} & c_0 & \dots & c_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & c_3 & \dots & c_0 \end{pmatrix} \in \mathbf{R}^{n \times n} \quad (11.5.22)$$

corresponds to problems where the boundary conditions are periodic. A matrix of this form (11.5.22) is called a **circulant matrix**. Each column in C_n is a cyclic up-shifted version of the previous row.

If P_n is the (circulant) permutation matrix,

$$P_n = \begin{pmatrix} 0 & I_{n-1} \\ e_1^T & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ & & & \ddots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{pmatrix} \quad (11.5.23)$$

then

$$P_n e_1 = e_2, P_n e_2 = e_3, \dots, P_n e_{n-1} = e_n, P_n e_n = e_1,$$

where e_i be the i th unit vector. It follows that $P_n^n e_i = e_i$, $i = 1 : n$, and thus $P_n^n - I = 0$. Since no polynomial of degree $n - 1$ in P_n vanishes, the characteristic polynomial of P_n is $\varphi(\lambda) = \lambda^n - 1$. Hence, the eigenvalues of P_n are the n roots of unity

$$\omega_j = e^{-2\pi j/n}, \quad j = 0 : n - 1.$$

It is readily verified that the eigenvectors are the Fourier vectors, i.e., the columns of the matrix $F = (f_{jk}$, where

$$f_{jk} = \frac{1}{\sqrt{n}} e^{2\pi i j k / n}, \quad 0 \leq j, k \leq n, \quad (i = \sqrt{-1}).$$

The circulant matrix (11.5.22) can be written as a polynomial in P_n , $C_n = \sum_{k=0}^{n-1} c_k P_n^k$. Therefore it has the same eigenvectors as P_n . Its eigenvalues are given by the components of the Fourier transform of its first column

$$F(c_0, c_{n-1}, \dots, c_{-1})^T = (\lambda_1 \ \dots \ \lambda_n)^T. \quad (11.5.24)$$

The matrix C can thus be factorized as

$$C = F \Lambda F^H, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n). \quad (11.5.25)$$

It follows that linear systems with a circulant matrix can be solved quickly using FFT; see Vol. I, Section 4.7.2.

Any rectangular Toeplitz matrix $T \in \mathbf{R}^{m \times n}$ can be embedded in a square circulant matrix

$$C_T = \left(\begin{array}{cccc|ccc} t_0 & t_1 & \dots & t_{n-1} & t_{-m+1} & \dots & t_{-1} \\ t_{-1} & t_0 & \dots & t_{n-2} & t_n & \dots & t_{-m+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hline t_{-m+1} & t_{-m+2} & \dots & t_0 & t_1 & \dots & t_{n-1} \\ t_{n-1} & t_{-m+1} & \dots & t_{-1} & t_0 & \dots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ t_1 & t_2 & \dots & t_{-m+1} & t_{-m+2} & \dots & t_0 \end{array} \right) \in \mathbf{R}^{p \times p},$$

where $p = m + n - 1$ and T corresponds to the $(1, 1)$ -block. To form the product of $y = Tx$, where $x \in \mathbf{R}^{n+1}$ is an arbitrary vector, we pad x with zeros and calculate

$$z = C_T \begin{pmatrix} x \\ 0 \end{pmatrix} = F \Lambda F^H \begin{pmatrix} x \\ 0 \end{pmatrix} \quad y = (I_m \quad 0) z.$$

This can be done with two FFTs, and one multiplication with a diagonal matrix, and hence the cost of this is $O(n \log_2 n)$ operations.

Since the transpose of T is also a Toeplitz matrix, a similar scheme can be used for the fast computation of $T^T y$. It follows that the fast multiplication algorithm can be used also to obtain efficient implementations of iterative methods for least squares problems such as LSQR or CGLS. These only require matrix-vector multiplications with T and T^T .

Efficient iterative methods for solving symmetric positive definite Toeplitz systems have been developed that use the conjugate gradient methods preconditioned with a suitable circulant matrix. One possible choice is given in the following theorem.

Theorem 11.5.3 (T. Chan [100]).

The circulant matrix C that minimizes $\|C - T\|_F$ for a (not necessarily symmetric positive definite) Toeplitz matrix has elements given by

$$c_k = \frac{kt_{-(n-k)} + (n-k)t_k}{n}, \quad k = 0 : (n-1).$$

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 equal to zero proves the result. \square

Example 11.5.2.

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 $n = 4$ we obtain

$$T = \begin{pmatrix} t_0 & t_1 & t_2 & t_3 \\ t_1 & t_0 & t_1 & t_2 \\ t_2 & t_1 & t_0 & t_1 \\ t_3 & t_2 & t_1 & t_0 \end{pmatrix}, \quad C = \begin{pmatrix} t_0 & \alpha & t_2 & \alpha \\ \alpha & t_0 & \alpha & t_2 \\ t_2 & \alpha & t_0 & \alpha \\ \alpha & t_2 & \alpha & t_0 \end{pmatrix}, \quad \alpha = (3t_1 + t_3)/3.$$

Note that $T = C$ if and only if $t_1 = t_3$.

Circulant preconditioners can also be used for the least squares problem

$$\min \|Tx - b\|_2, \quad T = \begin{pmatrix} T_1 \\ \vdots \\ T_q \end{pmatrix} \in \mathbf{R}^{m \times n}, \quad (11.5.26)$$

where each block T_j , $j = 1, \dots, q$, is a square Toeplitz matrix. (Note that if T itself is a rectangular Toeplitz matrix, then each block T_j is necessarily Toeplitz.)

First a circulant approximation C_j is constructed for each block T_j . Each circulant matrix C_j , $j = 1, \dots, q$, is diagonalized by the Fourier matrix F , $C_j = F\Lambda_j F^H$, where Λ_j is diagonal and F^H denotes the conjugate transpose of the complex Fourier matrix F . The eigenvalues Λ_j can be found from the first column of C_j ; cf. (11.5.24). Hence the spectrum of C_j , $j = 1, \dots, q$, can be computed in $O(m \log n)$ operations by using 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 defined by

$$C \equiv F^H \left(\sum_{j=1}^q \Lambda_j^H \Lambda_j \right)^{1/2} F. \quad (11.5.27)$$

The preconditioned (PCGLS) method is then applied with $S = C$ and $A = T$. Notice that to use the preconditioner C we need only know its eigenvalues, since the right-hand side of (11.5.27) can be used to solve linear systems involving C and C^T .

Similar methods can be applied also to two-dimensional or multidimensional problems.

Review Questions

- 5.1** What is meant by preconditioning of a linear system $Ax = b$. What are the properties that a good preconditioner should have?
- 5.2** What is meant by an incomplete factorization of a matrix? How can incomplete factorizations be used as preconditioners.
- 5.3** Describe two different transformations of a general unsymmetric linear system $Ax = b$, that allows the transformed system to be solved by the standard CG method.
- 5.4** Any circulant matrix. C can be written as a polynomial in a certain permutation matrix. Which?

- 5.5** Let $T \in \mathbf{R}^{m \times n}$ be a rectangular Toeplitz matrix. Describe how T can be embedded in a square circulant matrix. How can this embedding be used to perform fast matrix-vector operations Tx , for an arbitrary vector $x \in \mathbf{R}^n$?
-

Problems and Computer Exercises

- 5.1** Consider square matrices of order n , with nonzero elements only in the k -th upper diagonal, i.e., of the form

$$T_k = \begin{pmatrix} t_1 & & & \\ & \ddots & & \\ & & t_{n-k} & \end{pmatrix}, \quad k \geq 0.$$

Show the following rule for *multiplication by diagonals*:

$$A_k B_l = \begin{cases} C_{k+l}, & \text{if } k+l \leq n-1; \\ 0, & \text{otherwise,} \end{cases}$$

where the elements in C_{k+l} are $a_1 b_{k+1}, \dots, a_{n-k-l} b_{n-l}$.

- 5.2** 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.

- 5.3** Implement in MATLAB the right preconditioned CG method.

- 5.4** The penta-diagonal matrix of the model problem has nonzero elements in positions $\mathcal{P}_A = \{(i, j) \mid |i - j| = 0, 1, n\}$, which defines a level 0 incomplete factorization. Show that the level 1 incomplete factorization has two extra diagonals corresponding to $|i - j| = n - 1$.

- 5.5** The triangular solves needed when using an incomplete Cholesky factorizations as a preconditioner are inherently sequential and difficult to vectorize. 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.

- 5.6** Show that the optimal circulant preconditioner given in Theorem 11.5.3 is symmetric if and only if T is symmetric.

5.7 Let A be a symmetric positive definite matrix. An incomplete Cholesky preconditioner for A is obtained by neglecting elements in places (i, j) prescribed by a symmetric set

$$\mathcal{P} \subset \mathcal{P}_n \equiv \{(i, j) \mid 1 \leq i, j \leq n\},$$

where $(i, j) \in \mathcal{P}$, if $i = j$.

The simplest choice is to take \mathcal{P} equal to the sparsity pattern of A , which for the model problem is $\mathcal{P}_A = \{(i, j) \mid |i - j| = 0, 1, n\}$. This is called a level 0 incomplete factorization. A level 1 incomplete factorization is obtained by using the union of \mathcal{P}_0 and the pattern of the defect matrix $R = LL^T - A$. Higher level incomplete factorizations are defined in a similar way.

(a) Consider the model problem, where A is block tridiagonal

$$A = \text{tridiag}(-I, T + 2I, -I) \in \mathbf{R}^{n^2 \times n^2}, \quad T = \text{tridiag}(-1, 2, -1) \in \mathbf{R}^{n \times n}.$$

Show that A is an M -matrix and hence that an incomplete Cholesky factorizations of A exists?

(b) Write a MATLAB function, which computes the level 0 incomplete Cholesky factor L_0 of A . (You should *NOT* write a general routine like that in the textbook, but an efficient routine using the special five diagonal structure of A !) Implement also the preconditioned CG method in MATLAB, and a function which 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!

(c) 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 level 1 preconditioner. You may use, e.g., the built-in MATLAB functions to compute the eigenvalues, and efficiency is not a premium here. (To handle the level 1 preconditioner you need to generalize your incomplete Cholesky routine.)

11.6 Large-Scale Eigenvalue Problems

In many applications eigenvalue problems arise involving matrices so large that they cannot be conveniently treated by the methods described so far. For such problems, it is not reasonable to ask for a complete set of eigenvalues and eigenvectors, and usually only some extreme eigenvalues (often at one end of the spectrum) are required. In the 1980's typical values could be to compute 10 eigenpairs of a matrix of order 10,000. In 2007 some problems in material science were reported to require the computation of 20,000 eigenpairs of a Hermitian matrix of size several millions.

11.6.1 The Rayleigh–Ritz Procedure

For obtaining approximate solutions of a linear system from a subspace, the projection methods described in Section 11.2.1 played a central role. Similar projection techniques are useful also for obtaining approximate solutions to eigenvalue problems.

Let the matrix $A \in \mathbf{C}^{n \times n}$ have eigenvalues λ_i and eigenvectors x_i , $i = 1 : n$. Let \mathcal{K} be a given (low-dimensional) subspace of dimension $k \ll n$. Let $\hat{x} \in \mathcal{K}$ be an approximate eigenvector and $\hat{\lambda}$ the corresponding approximate eigenvalue. To obtain k independent conditions that determine this approximate eigenpair, we impose the Petrov–Galerkin conditions, i.e., require that the residual $(A - \hat{\lambda}I)\hat{x}$ is orthogonal to all vectors in a subspace \mathcal{L} of dimension k ,

$$(A - \hat{\lambda}I)\hat{x} \perp \mathcal{L}. \quad (11.6.1)$$

We first consider the choice $\mathcal{K} = \mathcal{L}$. A matrix form of the condition (11.2.1) is obtained by introducing orthogonal basis vectors u_1, \dots, u_k in \mathcal{K} . Setting $U_k = (u_1, \dots, u_k)$ and $\hat{x} = U_k y$, $y \in \mathbf{C}^k$ (11.6.1) becomes

$$U_k^H(A - \hat{\lambda}I)U_k y = 0. \quad (11.6.2)$$

Since $U_k^H U_k = I$, this is equivalent to the eigenvalue problem

$$By = \hat{\lambda}y, \quad B = U_k^T A U_k.$$

Here B is the orthogonal projection of A on \mathcal{K} . This small eigenvalue problems can be solved using the methods developed in Chapter 10. The solution yields k approximate eigenvalues of A . The matrix B is precisely the matrix Rayleigh quotient of A , which minimizes the residual norm

$$\|AU_k - U_k B\|_2.$$

Note that since U_k is orthogonal the conditioning of the projected eigenvalue problem is not changed. This orthogonal projection method known as the Rayleigh–Ritz procedure⁷⁹ and can be summarized as follows:

Algorithm 11.18. *The Rayleigh–Ritz procedure.*

1. Form the matrix $AU_k = (Au_1, \dots, Au_k)$ and the matrix Rayleigh quotient matrix

$$B = U_k^H(AU_k) \in \mathbf{R}^{k \times k}. \quad (11.6.3)$$

2. Compute the k eigenvalues and eigenvectors of B

$$Bz_i = \theta_1 z_i, \quad i = 1 : k. \quad (11.6.4)$$

The eigenvalues θ_i are the **Ritz values**, and the vectors $y_i = U_k z_i$ the **Ritz vectors**.

⁷⁹Named after Walther Ritz (1878–1909) Swiss theoretical physicist. Ritz studied in Zürich at the same department as Albert Einstein. He is most famous for his work in electrodynamics. He died prematurely from tuberculosis.

3. Compute the residual matrix $R = (r_1, \dots, r_k)$, where

$$r_i = Ay_i - y_i\theta_i = (AU_k)z_i - y_i\theta_i. \quad (11.6.5)$$

We now consider projection methods obtained when in (11.6.1) the subspace \mathcal{L} is chosen different from \mathcal{K} . We assume that v_1, \dots, v_m is a basis for the subspace \mathcal{L} such that $V_k^H U_k = I$. Such a pair of biorthogonal basis always exists if no vector in \mathcal{K} is orthogonal to \mathcal{L} . The projected eigenvalue problem now becomes

$$V_k^H (A - \hat{\lambda}I)U_k y = (B - \hat{\lambda}I)y = 0. \quad (11.6.6)$$

where $B = V_k^H A U_k$. This is an **oblique projection method**. Note that the eigenvalue problem for B potentially can be much worse conditioned than the original.

The choice $\mathcal{L} = A\mathcal{K}$ is of a particular interest. The projected problem becomes $(AU_k)^H(A - \hat{\lambda}I)U_k y = 0$, or

$$(AU_k)^H(AU_k)y = \hat{\lambda}U_k^H A^H U_k y. \quad (11.6.7)$$

This is a generalized eigenvalue problem for which the left hand side matrix is Hermitian positive definite. If we choose the basis matrix U_k such that $W_k = AU_k$ is an orthonormal system, then the projected eigenproblem becomes

$$\hat{\lambda}^{-1}I - W_k^H A^{-1}W_k)y = 0,$$

which is a projected eigenproblem for the eigenvalues of A^{-1} . The corresponding eigenvalue approximations are called the **harmonic Ritz values**⁸⁰ and are appropriate for determining *interior eigenvalues* of A . The smallest positive harmonic Ritz values approach the smallest positive eigenvalues of A monotonically from above and the largest positive harmonic Ritz values approach the largest eigenvalues monotonically from below.

The pairs (θ_i, y_i) , $i = 1 : k$ are the best approximate eigenpairs of A which can be derived from the subspace span (U_k) . From the residuals computed in step 3 of the Rayleigh–Ritz procedure we get computable backward error bounds for the approximate eigenvalues θ_i , $i = 1 : k$. By Theorem 10.2.6 the Ritz value θ_i is an exact eigenvalue of a matrix $A + E_i$, where

$$\|E_i\|_2 \leq \|r_i\|_2.$$

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 Hermitian case we have $\kappa(X) = 1$. Further, the Ritz vectors can be chosen so that $Z = (z_1, \dots, z_m)$ is unitary and the projected matrix B is Hermitian. Then, to each Ritz value θ_i there is an eigenvalue λ_i of A such that

$$|\theta_i - \lambda_i| \leq \|r_i\|_2, \quad j = 1 : k. \quad (11.6.8)$$

⁸⁰They are called harmonic because they are harmonic averages of Ritz values of A^{-1} .

No residual bound for the error in a Ritz vector y_i can be given without further information. This is to be expected, since 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.

Recall that the gap in the spectrum with respect to an approximate eigenvalue θ_i was defined 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 10.2.16 we have the improved bound

$$|\theta_i - \lambda_i| \leq \|r_i\|_2^2 / \text{gap}(\theta_i). \quad (11.6.9)$$

Further, if x_i is an eigenvector of A associated with λ_i we have

$$\sin \theta(y_i, x_i) \leq \|r_i\|_2 / \text{gap}(\theta_i). \quad (11.6.10)$$

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 11.6.1 (Kahan [361]).

Let $A \in \mathbf{C}^{n \times n}$ be Hermitian and $U_k \in \mathbf{C}^{n \times k}$ have orthonormal columns and set

$$B = U_k^H A U_k, \quad R = A U_k - U_k B.$$

Then to the eigenvalues $\theta_1, \dots, \theta_k$ of B there correspond eigenvalues $\lambda_1, \dots, \lambda_k$ of A such that

$$|\lambda_i - \theta_i| \leq \|R\|_2, \quad i = 1 : k.$$

Further, there are eigenvalues $\lambda_1, \dots, \lambda_k$ of A such that

$$\sum_{i=1}^k (\lambda_i - \theta_i)^2 \leq \|R\|_F^2.$$

Proof. See Parlett [476, Sec. 11-5]. \square

11.6.2 The Arnoldi Algorithm

An obvious choice of subspaces for the Rayleigh–Ritz procedure is the Krylov subspaces $\mathcal{K}_m(A, v)$ (see Definition 11.2.6). These satisfy the following simply verified invariance properties:

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

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

These invariance can be used to deduce some important properties of methods using Krylov subspaces. Since A and $-A$ generate the same subspaces the left and right part of the spectrum of A are equally approximated.

We recall that any vector $x \in \mathcal{K}_m(A, v)$ can be written in the form

$$x = \sum_{i=0}^{m-1} c_i A^i v = P_{m-1}(A)v,$$

where P_{m-1} is a polynomial of degree less than m . This provides a link between polynomial approximation and Krylov subspace methods.

The Rayleigh–Ritz procedure can be applied efficiently to the *sequence* of Krylov subspaces $\mathcal{K}_k(A, v)$, $k = 1, 2, \dots$. At each step the dimension of the subspace increases by one. Let $A \in \mathbf{C}^{n \times n}$ and $v_1, \|v_1\|_2 = 1$, a given starting vector. The Arnoldi process (Algorithm 11.4.1) computes a Hessenberg matrix $H_m = (h_{ij})$ and a matrix $V_m = (v_1, \dots, v_m)$ with orthogonal columns spanning the Krylov subspace $\mathcal{K}_m(A, v_1)$, such that the Arnoldi decomposition

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T \quad (11.6.11)$$

holds. Here 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—that is, if its subdiagonal elements are nonzero—then by Theorem 10.4.4 this decomposition is uniquely determined by the starting vector $v_1 = V_m e_1$. The Ritz values are the eigenvalues of H_m ,

$$H_m z_i = \theta_i z_i, \quad i = 1 : m, \quad (11.6.12)$$

and the corresponding Ritz vectors are $y_i = V_m z_i$. Using (11.6.11) and (11.6.12) it follows that the residual norm satisfies

$$\begin{aligned} \|Ay_i - y_i \theta_i\|_2 &= \|(AV_m z_i - V_m z_i \theta_i)\|_2 = \|(AV_m - V_m H_m)z_i\|_2 \\ &= h_{m+1,m} |e_m^H z_i|. \end{aligned} \quad (11.6.13)$$

The process will break down at step j if and only if the vector r_{j+1} vanishes. When this happens we have $h_{m+1,m} = 0$ and hence $AV_m = V_m H_m$. Then $\mathcal{R}(V_m)$ is an invariant subspace of A .

When used in GMRES, modified Gram–Schmidt orthogonalization in the Arnoldi process gave a sufficient level of orthogonality for backward stability. When the Arnoldi process is used for eigenvalue computations orthogonality to working precision should be enforced in the Arnoldi vectors. This can be achieved, e.g., by performing classical Gram–Schmidt twice; see [256].

Example 11.6.1.

A matrix of order 100 with entries normally distributed with zero mean and standard deviation one was generated. Figure 11.6.1 shows the eigenvalues of A and

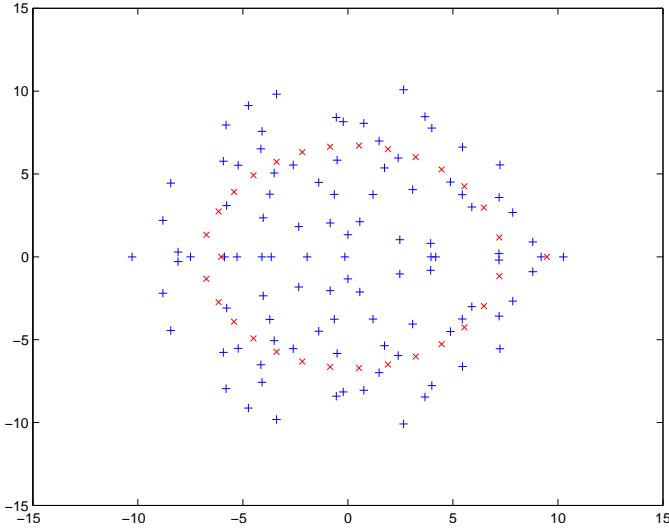


Figure 11.6.1. Eigenvalues (+) and Ritz values (x) for a 100×100 matrix with normally distributed entries after 30 steps of Arnoldi.

the Ritz values obtained after 30 steps of the the Arnoldi method have been carried out using a random starting vector. Note that the Ritz values tend to approximate eigenvalues in the exterior of the spectrum better than interior eigenvalues. The Ritz values have not yet converged to any of the eigenvalues.

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 $O(nk)$ 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. A natural way to choose an the new starting vector is to select the current $k \leq m$ Ritz vectors y_i that best approximate the wanted eigenvalues and take

$$v_1^+ = \sum_{i=1}^k \gamma_i y_i, \quad (11.6.14)$$

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 (11.6.14), we have

$$v_1^+ = \psi(A)v_1, \quad (11.6.15)$$

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 are not of interest, the corresponding eigenvalues will be deemphasized by the restart.

When in the Arnoldi method an eigenvalue has converged this should be eliminated from the rest of the process using some form of deflation. The matrix A can be deflated explicitly as discussed in Section 10.3.2. If k eigenvalues and the corresponding Schur vectors $V_k = (v_1, \dots, v_k)$ are known, then we can continue to work with the deflated matrix

$$A - U_k \Lambda_k U_k^H.$$

Another possibility is to integrate the deflation process in the Arnoldi algorithm as follows. Then we works with a basis $v_1, \dots, v_k, v_{k+1}, \dots, v_m$, where the first k vectors are Schur vectors, which are fixed once they have been computed. At a restart the 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.

A much 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 we have computed an Arnoldi decomposition (11.6.11) 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 straightforward. Let us perform one step of the implicit QR algorithm,described in Section 10.4.3 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, setting $\tilde{H}_m = Q^H H_m Q$ and $\tilde{V}_m = V_m Q$,

$$A\tilde{V}_m = \tilde{V}_m \tilde{H}_m + h_{m+1,m} v_{m+1} e_m^T Q. \quad (11.6.16)$$

Here \tilde{V}_m is unitary and \tilde{H}_m Hessenberg. Further,

$$e_m^T Q = e_m^T G_{m-1,m} = s_m e_{m-1}^T + c_m e_m^T,$$

which shows that only the last two components of $e_m^T Q$ are nonzero. If we drop the last column of each term in (11.6.16), we obtain the truncated decomposition

$$A\tilde{V}_{m-1} = \tilde{V}_{m-1} \tilde{H}_{m-1} + \hat{h}_{m,m-1} \hat{v}_m e_{m-1}^T, \quad (11.6.17)$$

where \tilde{H}_{m-1} is the leading principal submatrix of \tilde{H}_m of order $m - 1$. Using 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 both are orthogonal to \tilde{V}_{m-1} so is \hat{v}_m . Hence, (11.6.17) 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 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

$$\tilde{v}_1 = \frac{p(A)v_1}{\|p(A)v_1\|_2}, \quad p(z) = (z - \theta_1) \cdots (z - \theta_{m-k}). \quad (11.6.18)$$

The normal procedure in implicitly restarted Arnoldi is as follows. Initially $m = k + p$ steps of an Arnoldi decomposition is 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 the \widetilde{H}_k having the k wanted Ritz values as its spectrum. As the restart process is repeated the successive subdiagonals of \widetilde{H}_k will tend to zero and converge to a partial Schur decomposition of A with the corresponding Schur vectors given by \widetilde{V}_k .

11.6.3 The Simple Lanczos Method

For a real symmetric (Hermitian) matrix the Arnoldi method is greatly simplified. (Note that by the invariance with respect to shifting it does not matter if A is positive definite or not.) The Lanczos algorithm (see Section 11.2.5) yields after k steps the Lanczos decomposition

$$AU_k = U_k T_k + \beta_{k+1} u_{k+1} e_k^T. \quad (11.6.19)$$

By construction, it follows that $\text{span}(U_k) = \mathcal{K}_k(A, b)$. Multiplying (11.6.19) by U_k^H and using $U_k^H u_{k+1} = 0$ it follows that the generalized Rayleigh quotient matrix corresponding to $\mathcal{K}_k(A, b)$ is the symmetric tridiagonal matrix

$$T_k = U_k^H A U_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \alpha_{k-1} & \beta_k \\ & & & \beta_k & \alpha_k \end{pmatrix}. \quad (11.6.20)$$

The Ritz values are the eigenvalues $\theta_i^{(k)}$ of T_k , and the Ritz vectors are

$$y_i^{(k)} = U_k z_i,$$

where $z_i^{(k)}$ are the normalized eigenvectors of T_k corresponding to $\theta_i^{(k)}$. The Ritz values and vectors can be computed efficiently using the Hermitian QR algorithm. It follows from Cauchy's interlacing property (Theorem 10.2.11) 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 *the Ritz values approach the eigenvalues of A monotonically from the inside out*.

If $\beta_{k+1} = 0$ the Lanczos process terminates with $AU_k = U_k T_k$. This will happen for some $k \leq n$ in exact arithmetic. This means that U_k spans an invariant subspace of A and that the eigenvalues of T_k are exact eigenvalues of A . The remaining eigenvalues of A can be determined by restarting the Lanczos process with a vector orthogonal to u_1, \dots, u_k . For example, if u_1 happens to be an eigenvector of A , the process stops after one step.

In principle the 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 be assessed from the residual norms

$$\|Ay_i - y_i\theta_i\|_2,$$

and used to decide when the process should be stopped.

Theorem 11.6.2. *Suppose k steps of the symmetric Lanczos method has been performed. Let the Ritz values be θ_i and the Ritz vectors $y_i = U_k z_i$, $i = 1 : k$. Then we have*

$$\|Ay_i - y_i \theta_i\|_2 = \xi_{ki}, \quad \xi_{ki} = \beta_{k+1} |e_k^T z_i|, \quad (11.6.21)$$

and a bound for the error in the Ritz values is

$$\min_{\mu \in \lambda(A)} |\mu - \theta_i| \leq \xi_{ki}. \quad (11.6.22)$$

Proof. Substituting $y_i = U_k z_i$ and using (11.6.19) 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 (11.6.21) and then (11.6.21) follows from Theorem 10.2.10. \square

Hence, the residual norm can be computed from *the bottom element of the normalized eigenvectors of T_k* . Therefore, the matrix U_k need 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 (11.6.21) also explains why some Ritz values can be very accurate approximations even when β_{k+1} is not small.

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 (11.6.9)–(11.6.10) 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 $A\mathcal{K}_k(A, b)$. The harmonic Ritz values are determined by (11.6.7), which using the Lanczos decomposition becomes

$$(AU_k)^T (A - \bar{\lambda}I) U_k y_k = ((AU_k)^T (AU_k) - \bar{\lambda}T_k) y_k = 0. \quad (11.6.23)$$

Squaring the Lanczos decomposition gives

$$(AU_k)^T (AU_k) = T_k^T U_k^T U_k T_k + \beta_{k+1} e_k u_{k+1}^T u_{k+1} e_k^T = T_k^2 + \beta_{k+1} e_k e_k^T,$$

and using the symmetry of T_k and $U_k^T u_{k+1} \neq 0$. Substituting this in (11.6.23) we find that

$$(T_k^2 + \beta_{k+1} e_k e_k^T - \bar{\lambda}T_k) y_k = 0.$$

Finally, multiplying with T_k^{-1} shows that the harmonic Ritz values and Ritz vectors are the eigenpairs of the symmetric matrix

$$T_k + \beta_{k+1} T_k^{-1} e_k e_k^T. \quad (11.6.24)$$

This is a tridiagonal matrix modified by a rank one matrix. By Theorem 10.2.12 the harmonic Ritz values interlace the standard Ritz values and can be computed by solving a secular equation.

Eigenvalues close to zero of A 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, 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, since they are the eigenvalues of the matrix

$$(T_k - \mu I) + \beta_{k+1}(T_k - \mu I)^{-1}e_k e_k^T. \quad (11.6.25)$$

The Ritz values together with the harmonic Ritz values of the shifted matrix $(A - \mu I)$ form the so-called Lehmann intervals, which are optimal inclusion intervals for eigenvalues of A .

11.6.4 Lanczos Method in Finite Precision

So far we have discussed the Lanczos process 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. This will cause the generated Lanczos vectors to gradually lose orthogonality. The Lanczos decomposition becomes

$$AU_j = U_j T_j + \beta_{j+1}u_{j+1}e_j^T + F_j,$$

where $\|F_j\|_2 \leq cu\|A\|_2$, u the unit roundoff, and c 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.

A satisfactory analysis of the numerical properties of the Lanczos process was not given in 1971 by Paige [462, 1971]. Paige observed that the loss of orthogonality in the Lanczos process was related to the convergence of a Ritz pair to an eigenpair of A . A famous result by Paige shows that after k steps of the Lanczos process, the newly generated Lanczos vector satisfies a relation of the form

$$|q_{k+1}y_i| \approx u \frac{\|A\|_2}{|\beta_{k+1}| |e_k^T z_i|},$$

where u is the unit roundoff. For the error in the Ritz value This can be compared with the error estimate corresponding to y given by (11.6.21), which is

$$\|Ay_i - y_i \theta_i\|_2 = \beta_{k+1} |e_k^T z_i|,$$

This 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.

The reintroduction of components of converged eigenvectors leads to duplications of converged eigenpairs in the reduced tridiagonal matrix. This does not prevent convergence to other eigenpairs or cause a loss of accuracy in the computed approximations. The simple Lanczos method has been shown to be very effective in computing accurate approximations to a few of the extreme eigenvalues of A , even in the face of total loss of orthogonality. Similarly, when used to solve linear systems the main effect of finite precision is to delay convergence, but not prevent accurate approximations to be found. However, the loss of orthogonality delays convergence and causes possible duplication copies of eigenvalues to appear. The convergence of the Ritz values can become quite irregular and sometimes almost stagnate. It is difficult to know if a converging Ritz value is a new close eigenvalue or a copy of an eigenvalue already found.

A possible remedy to these effects is to reorthogonalize each new Lanczos vector q_{k+1} to *all* previous vectors u_j, \dots, u_1 . This requires that all Lanczos vectors are stored and is clearly very costly both in terms of storage and operations. An alternative to this scheme of *full* reorthogonalization is to employ *selective* or *periodic* reorthogonalization of the Lanczos vectors. The loss of orthogonality among vectors in the matrix $U_k \in \mathbf{R}^{n \times k}$ can be 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. A key result from the error analysis says that as long as the loss of orthogonality in U_k is bounded by

$$\omega \leq \sqrt{u/k}, \tag{11.6.26}$$

the Lanczos process will behave sufficiently close 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 $O(u\|A\|_2)$. If (11.6.26) is satisfied, we say that semiorthogonality holds. The aim of a reorthogonalization process is to maintain semiorthogonality. Several schemes that maintain semiorthogonality have been suggested; see, e.g., Simon [529].

An different approach is to use the simple Lanczos method. i.e., with no reorthogonalization steps, and use bounds for the errors in the Ritz values to locate and delete spurious copies of eigenvalues. At the end the Lanczos basis vectors are regenerated and the Ritz vectors reconstructed.

Even when selective reorthogonalization is used the Lanczos method can take many iterations before approximations to the desired eigenvalues have converged. A remedy is to use implicit restarts as in IRAM. The implicitly restarted Lanczos method (IRLM) avoids the 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

$$AU_m = U_m T_m + \beta_{m+1} u_{m+1} e_m^T, \quad (11.6.27)$$

has been computed. Here T_m is a real symmetric tridiagonal matrix and U_k a matrix with orthogonal columns spanning the Krylov subspace $\mathcal{K}_m(u_1, A)$. 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

$$A\tilde{U}_m = \tilde{U}_m \tilde{T}_m + \beta_{m+1} v_{m+1} e_m^T Q, \quad (11.6.28)$$

where $\tilde{T}_m = Q_m^T T_m Q$ and $\tilde{U}_m = U_m Q$. Deleting the last column will produce a truncated decomposition of the form

$$A\tilde{U}_{m-1} = \tilde{U}_{m-1} \tilde{T}_{m-1} + \hat{\beta}_m \hat{u}_m e_{m-1}^T, \quad (11.6.29)$$

where \tilde{T}_{m-1} is the leading principal submatrix of \tilde{T}_m of order $m - 1$. Only the last two components in $e_m^T Q$ will be nonzero and using the tridiagonal structure of \tilde{T}_m , we obtain

$$\hat{\beta}_m \hat{u}_m = \tilde{\beta}_m \tilde{u}_m + s_m \beta_{m+1} u_{m+1}.$$

This shows that \hat{u}_m is orthogonal to \tilde{U}_{m-1} and thus (11.6.29) 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.

The Lanczos bidiagonalization described in Section 11.3.4, can be used for computing singular values and singular vectors of a matrix $A \in \mathbf{R}^{m \times n}$. This is mathematically equivalent to applying the symmetric Lanczos process to the matrix

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}, \quad A \in \mathbf{R}^{m \times n}, \quad (11.6.30)$$

using a special starting vector. This halves the work compared to if the Lanczos method was applied naively to (11.6.30). Let B_k be the bidiagonal matrix obtained after k steps of the bidiagonalization process applied to A . Let the SVD of B_k be.

$$B_k = P_{k+1} \Omega_k Q_k^T, \quad \Omega = \text{diag}(\sigma_1, \dots, \sigma_k). \quad (11.6.31)$$

Then σ_i , $i = 1 : k$ are approximations to singular values of A and the vectors

$$\hat{U}_k = U_k P_{k+1}, \quad \hat{V}_k = V_k Q_k,$$

where U_k and V_k are the Lanczos vectors.

This is a simple way of realizing the Ritz–Galerkin projection process on the subspaces $\mathcal{K}_j(A^T A, v_1)$ and $\mathcal{K}_j(A A^T, A v_1)$. The corresponding approximations are called Ritz values and Ritz vectors.

Lanczos bidiagonalization algorithms have been successfully used in numerous large-scale applications, e.g., in information retrieval, seismic tomography, regularization methods, and other applications. In these, typically, the 100–200 largest singular values and vectors for matrices having up to 30,000 rows and 20,000 columns are required.

In Section 11.3.4 we discussed the use of Lanczos bidiagonalization and LSQR for solving least squares problems. In this application the loss of orthogonality in the left and right Lanczos vectors could be disregarded. It is also true that the extreme (largest and smallest) singular values of the truncated bidiagonal matrix $B_k \in \mathbf{R}^{(k+1) \times k}$ tend to be quite good approximations to the corresponding singular values of A , even for $k \ll n$. Often the smaller singular values are clustered, which will delay convergence for this part of the spectrum.

When using the Lanczos bidiagonalization for computing low-rank approximations, it is desirable to avoid the spurious singular values that will appear when orthogonality is lost. If just a few triplets are wanted, full reorthogonalization can be used. Otherwise a partial reorthogonalization scheme is preferable. Such a scheme, which ensures the semiorthogonality of the left and right Lanczos vectors has been developed by Simon and Zha [530]. An interesting aspect of this is that the orthogonality of the left and right Lanczos vectors in V_k and U_k are closely coupled. If the matrix A is “skinny”, i.e., $n \ll m$, then it can suffice to enforce orthogonality among the short right Lanczos vectors, with a considerable saving in storage and operations.

The implicitly restarted Lanczos method can also be applied to Lanczos bidiagonalization. Similar strategies as in IRAM may be invoked in order to determine a fixed number k of the largest singular values and vectors. In particular, the “exact” shifts can be adopted in a straightforward fashion. Recall (see (11.3.36) in Section sec10.3.4) that the Lanczos bidiagonalization process yields the two decompositions

$$AV_m = U_{m+1}B_m, \tag{11.6.32}$$

$$A^T U_{m+1} = V_m B_m^T + \alpha_{m+1} v_{m+1} e_{m+1}^T, \tag{11.6.33}$$

The Golub and Reinsch bulge chasing implicit shift QR–SVD algorithm can now be applied to the lower bidiagonal matrix $B_m \in \mathbf{R}^{(m+1) \times m}$.⁸¹

We now describe the application of one shift in the implicitly restarted Lanczos

⁸¹Note that in Section 10.5.3 we worked with an upper bidiagonal matrix (10.5.19). This gives rise to certain minor differences in the QR–SVD algorithm.

SVD method. Given a shift μ^2 , a Givens rotation G_{12} is determined such that

$$\begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix} \begin{pmatrix} \alpha_1^2 - \mu^2 \\ \alpha_1\beta_1 \end{pmatrix} \begin{pmatrix} \tau \\ 0 \end{pmatrix}.$$

The matrix $G_{12}B_m$ is then brought back to bidiagonal form by applying further rotations alternately from left and right in bulge chasing step. The result is a new bidiagonal matrix

$$\tilde{B}_m = Q^T B_m P,$$

such that the tridiagonal matrix

$$\tilde{T}_m = \tilde{B}_m^T \tilde{B}_m = P^T B_m^T Q Q^T B P = P^T T_m P$$

is the result of one QR step applied to $T_m = B^T B$ with shift equal to μ .

The QR-SVD step transforms the two Lanczos decompositions in (11.6.32)–(11.6.33) into

$$A\tilde{V}_m = \tilde{U}_{m+1}\tilde{B}_m, \quad (11.6.34)$$

$$A^T\tilde{U}_{m+1} = \tilde{V}_m\tilde{B}_m^T + \alpha_{m+1}v_{m+1}e_{m+1}^T Q, \quad (11.6.35)$$

where $\tilde{V}_m = V_m P$ and $\tilde{U}_{m+1} = U_{m+1}Q$. By construction, only the last two elements of $e_{m+1}^T Q$ are nonzero

$$e_m^T Q = e_m^T G_{m-1,m} = s_m e_{m-1}^T + c_m e_m^T,$$

Deleting the last column of each term in (11.6.34) and in (11.6.35) gives

$$A\tilde{V}_{m-1} = \tilde{U}_m\tilde{B}_{m-1} \quad (11.6.36)$$

$$\begin{aligned} A^T\tilde{U}_m &= \tilde{V}_m \begin{pmatrix} \tilde{B}_{m-1}^T \\ \tilde{\alpha}_m e_m^T \end{pmatrix} + s_m \alpha_{m+1} v_{m+1} e_m^T \\ &= \tilde{V}_{m-1}\tilde{B}_{m-1}^T + \hat{\alpha}_m \hat{v}_m \end{aligned} \quad (11.6.37)$$

where

$$\hat{\alpha}_m \hat{v}_m = \tilde{\alpha}_m \tilde{v}_m + s_m \alpha_{m+1} v_{m+1} \quad (11.6.38)$$

Since \hat{v}_m is orthogonal to \tilde{V}_{m-1} (11.6.36) and (11.6.37) 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 Lanczos bidiagonalization with starting vector $\tilde{u}_1 = (AA^T - \mu^2)u_1$. If this is repeated for p shifts μ_1, \dots, μ_p , the result corresponds to $m-p$ steps of Lanczos bidiagonalization using the starting vector

$$\prod_{i=1}^p (AA^T - \mu_i^2) u_1.$$

It is often the case that the k largest (or smallest) singular value triples are wanted. The shifts in the implicitly restarted Lanczos SVD method are used to enhance the convergence to the wanted part of the SVD spectrum. For example, the shifts μ_1 can be taken as the p smallest (largest) singular values of B_m and the restart process repeated cyclically.

11.6.5 Convergence Analysis

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 of A be approximated by a vector in $\mathcal{K}(A, v)$? The answer is closely related to the theory of polynomial approximation. Let Π_k denote the orthogonal projector onto the Krylov subspace $\mathcal{K}_k(A, v)$. The following lemma bounds the distance $\|u_i - \Pi_k u_i\|_2$, where u_i is a particular eigenvector of A .

Theorem 11.6.3.

Assume that the matrix A is diagonalizable with eigenvalues λ_i and normalized eigenvectors x_i , $i = 1 : n$. Let the starting vector v in Lanczos method have the expansion

$$v = \sum_{k=1}^n \alpha_k x_k, \quad \alpha_i \neq 0. \quad (11.6.39)$$

Let P_{k-1} be the set of polynomials of degree at most $k-1$ normalized so that $p(\lambda_i) = 1$. Then the inequality

$$\|x_i - \Pi_k x_i\|_2 \leq \xi_i \epsilon_i^{(k)}, \quad \xi_i = \sum_{j \neq i} |\alpha_j| / |\alpha_i|, \quad (11.6.40)$$

holds, where Π_k is the orthogonal projector onto \mathcal{K}_k and

$$\epsilon_i^{(k)} = \min_{p \in P_{k-1}} \max_{\lambda \in \lambda(A) - \lambda_i} |p(\lambda)|. \quad (11.6.41)$$

where $\lambda(A)$ denotes the spectrum of A .

Proof. We note that any vector in \mathcal{K}_k can be written $q(A)v$, where q is a polynomial $q \in P_{k-1}$. Since Π_k is the orthogonal projector onto \mathcal{K}_k we have

$$\|(I - \Pi_k)u_i\|_2 \leq \|u_i - q(A)v\|_2.$$

Using the expansion (11.6.39) of v it follows that for any polynomial $p \in P_{k-1}$ with $p(\lambda_i) = 1$ we have

$$\|(I - \Pi_k)\alpha_i u_i\|_2 \leq \left\| \alpha_i u_i - \sum_{j=1}^n \alpha_j p(\lambda_j) u_j \right\|_2 \leq \max_{j \neq i} |p(\lambda_j)| \sum_{j \neq i} |\alpha_j|.$$

The last inequality follows by noticing that the component in the eigenvector u_i is zero and using the triangle inequality. Finally, dividing by $|\alpha_i|$ establishes the result. \square

The minimization problem in (11.6.41) is hard to solve in general. Therefore we now specialize and consider the Hermitian case. We assume that the eigenvalues are simple and ordered so that $\lambda_1 > \lambda_2 > \dots > \lambda_n$. Let $T_k(x)$ be the Chebyshev polynomial of the first kind of degree k . Then $|T_k(x)| \leq 1$ for $|x| \leq 1$, and for $|x| \geq 1$

$$T_k(x) = \frac{1}{2} \left[\left(x + \sqrt{x^2 - 1} \right)^k + \left(x - \sqrt{x^2 - 1} \right)^k \right]. \quad (11.6.42)$$

This shows the steep climb of $T_n(x)$ outside the interval $[-1, 1]$. As shown by Theorem 11.1.20 the minimax property of the Chebyshev polynomials give the solution to the minimization problem. For the eigenvalue λ_1 the linear transformation

$$x = l_1(\lambda) = 1 + 2 \frac{\lambda - \lambda_2}{\lambda_2 - \lambda_n}, \quad \gamma_1 = l_1(\lambda_1) = 1 + 2 \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}. \quad (11.6.43)$$

maps the interval $\lambda = [\lambda_2, \lambda_n]$ onto $x = [-1, 1]$, and $\gamma_1 > 1$. If we take

$$p(\lambda) = \frac{T_{k-1}(l_1(\lambda))}{T_{k-1}(\gamma_1)},$$

then $p(\lambda_1) = 1$ as required. When k is large we have

$$\epsilon_1^{(k)} \leq \max_{\lambda \in \lambda(A) - \lambda_1} |p(\lambda)| \leq \frac{1}{T_{k-1}(\gamma_1)} \approx 2 / \left(\gamma_1 + \sqrt{\gamma_1^2 - 1} \right)^{k-1}. \quad (11.6.44)$$

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. However, often the distribution of the eigenvalues is such that only those at one end of the spectrum can be found.

Example 11.6.2.

Consider a matrix with eigenvalues equal to $\lambda_1 = 20$, $\lambda_2 = 19$, and $\lambda_n = 0$. In this case we have

$$\gamma_1 = 1 + 2/19, \quad \gamma_1 + \sqrt{\gamma_1^2 - 1} = (21 + \sqrt{80})/19 \approx 1.576.$$

This indicates that as the dimension k of the Krylov space increases the error bound for the best approximation converges linearly with rate 0.635^k . This is much better than for the power method which will converge as 0.95^k . The analysis does not take into account that Krylov subspace method show superlinear convergence, i.e., the rate of convergence will increase with k , which is not the case for the power method.

By considering the matrix $-A$ we get analogous convergence results for the rightmost eigenvalue λ_n of A . More generally, for λ_i , $i > 1$, similar but weaker

results can be proved using polynomials of the form

$$p(\lambda) = q_{i-1}(\lambda) \frac{T_{k-i}(l_1(\lambda))}{T_{k-i}(\gamma_1)}, \quad q_{i-1}(\lambda) = \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda}{\lambda_j - \lambda_1}.$$

Notice that $q_{i-1}(\lambda)$ is a polynomial of degree $i-1$ with $q_{i-1}(\lambda_j) = 0$, $j = 1 : i-1$, and $q_{i-1}(\lambda_1) = 1$. Further,

$$\max_{\lambda \in \lambda(A) - \lambda_1} |q_{i-1}(\lambda)| \leq |q_{i-1}(\lambda_n)| = C_i. \quad (11.6.45)$$

Thus, when k is large we have

$$\epsilon_1^{(k)} \leq C_i / T_{k-i}(\gamma_1). \approx 2 / \left(\gamma_1 + \sqrt{\gamma_1^2 - 1} \right)^{k-1}. \quad (11.6.46)$$

This indicates that interior eigenvalues and eigenvectors can be expected to be less well approximated by Krylov-type methods.

In the non-Hermitian case the eigenvalues of A are complex the analysis is more difficult. Chebyshev polynomials are defined for complex arguments $z = \frac{1}{2}(w+w-1)$

$$T_k(z) = \frac{1}{2}(w^k + w^{-k}), \quad w = z \pm \sqrt{z^2 - 1}. \quad (11.6.47)$$

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

$$\mathcal{E}_R = \{z : |z-1| + |z+1| \leq R + R^{-1}\}, \quad (11.6.48)$$

with foci at 1 and -1 . The axes are, respectively, $R + R^{-1}$ and $R - R^{-1}$, and hence R is the sum of the semiaxes. 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 (11.1.43) that this family of confocal ellipses are level curves of $|w| = |z \pm \sqrt{z^2 - 1}|$. In fact, we can also write

$$\mathcal{E}_R = \left\{ z : 1 \leq |z + \sqrt{z^2 - 1}| \leq R \right\}. \quad (11.6.49)$$

A min-max result similar to Theorem 11.1.20 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 have been studied by Fischer and Freund [204].

Theorem 11.6.4 (Saad [518, Theorem 4.9]).

Consider the ellipse \mathcal{E}_R in (11.6.49) and let γ be a point not enclosed by it. Then

$$\frac{\rho^k}{|w_\gamma|^k} \leq \min_{p \in \Pi_k} \max_{z \in \mathcal{E}_R} |p(z)| \leq \frac{\rho^k + \rho^{-k}}{|w_\gamma|^k + |w_\gamma|^{-k}}, \quad (11.6.50)$$

in which w_γ is the dominant root of the equation $(w + w^{-1})/2 = \gamma$.

11.6.6 Simultaneous Iteration for Hermitian Matrices

In Section 10.3.5 subspace iteration, or orthogonal iteration, was introduced as a block version of the power method. Subspace iteration or **simultaneous iteration** has long been one of the most important methods for solving large sparse eigenvalue problems. In particular it has been used much in structural engineering, and developed to a high standard of refinement.

In simple subspace iteration we start with an initial matrix $Q_0 \in \mathbf{R}^{n \times p}$ ($1 < p \ll n$) with orthogonal columns. From this a sequence of matrices $\{Q_k\}$ are computed from

$$Z_k = AQ_{k-1}, \quad Q_k R_k = Z_k, \quad k = 1, 2, \dots, \quad (11.6.51)$$

where $Q_k R_k$ is the QR decomposition of the matrix Z_k . There is no need for the matrix A to be known explicitly; only an algorithm (subroutine) for computing the matrix-vector product Aq for an arbitrary vector q is required.

The iteration (11.6.51) generates a sequence of subspaces $\mathcal{S}_k = \mathcal{R}(A^k Q_0) = \mathcal{R}(Q_k)$. We seek approximate eigenvectors of A in these subspaces. It can be shown (see Section 10.3.5) 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. The convergence is linear with rate $|\lambda_{p+1}/\lambda_p|$.

For the individual eigenvalues $\lambda_i > \lambda_{i+1}$, $i \leq p$, it holds that

$$|r_{ii}^{(k)} - \lambda_i| = O(|\lambda_{i+1}/\lambda_i|^k), \quad i = 1 : p.$$

where $r_{ii}^{(k)}$ are the diagonal elements in 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 the improved algorithm below.

Algorithm 11.19. *Orthogonal Iteration, Hermitian Case.*

With $Q_0 \in \mathbf{R}^{n \times p}$ compute for $k = 1, 2, \dots$ a sequence of matrices Q_k as follows:

1. Compute $Z_k = AQ_{k-1}$;
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 eigenvalue decomposition $B_k = U_k \Theta_k U_k^T$;
5. Compute the matrix of Ritz vectors $Q_k = \bar{Q}_k U_k$.

It can be shown that

$$|\theta_i^{(k)} - \lambda_i| = O(|\lambda_{p+1}/\lambda_i|^k), \quad \Theta_k = \text{diag}(\theta_1^{(k)}, \dots, \theta_p^{(k)}),$$

which is a much more favorable rate of convergence than without the Rayleigh–Ritz procedure. The columns of Q_k are the Ritz vectors, and they will converge to the corresponding eigenvectors of A .

Example 11.6.3.

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 j th eigenvalue with and without Rayleigh–Ritz acceleration are:

j	without R-R	with R-R
1	0.99	0.1
2	0.99	0.101
3	0.102	0.102

The work in step 1 of Algorithm 11.6.6 consists of p matrix times vector operations with the matrix A . If the modified Gram–Schmidt method is used step 2 requires $p(p+1)n$ flops. To form the Rayleigh quotient matrix requires a further p matrix times vector multiplications 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.

Note that the same subspace \mathcal{S}_k is generated by k consecutive steps of 1, as with the complete Algorithm 11.6.6. Therefore, the rather costly orthogonalization and Rayleigh–Ritz acceleration need not be carried out at every step. However, to be able to check convergence to the individual eigenvalues we need the Rayleigh–Ritz approximations. If we then form the residual vectors

$$r_i = Aq_i^{(k)} - q_i^{(k)}\theta_i = (AQ_k)u_i^{(k)} - q_i^{(k)}\theta_i. \quad (11.6.52)$$

and compute $\|r_i\|_2$ each interval $[\theta_i - \|r_i\|_2, \theta_i + \|r_i\|_2]$ will contain an eigenvalue of A .

Algorithm 11.6.6 can be generalized to unsymmetric matrices, by substituting in step 4 the Schur decomposition

$$B_k = U_k S_k U_k^T,$$

where S_k is upper triangular. The vectors q_i then converge to the Schur vector u_i of A .

11.6.7 Spectral Transformation

Suppose that the smallest eigenvalue λ_n and its separation $|\lambda_n - \lambda_{n-1}|$ both are of order one. If the eigenvalue of largest magnitude is of order 10^8 then the separation relative to the spread is of order 10^{-8} . Then γ_1 in (11.6.43) is also of order 10^{-8} and this is one case when we can expect very slow convergence of Lanczos method. Another is when it is used to find interior eigenvalues of A . In both these cases a cure is to use inverse iteration; see Section 10.3.3). In this the power method was applied to the shifted and inverted matrix $\hat{A} = (A - \mu I)^{-1}$.

Consider now the generalized eigenvalue problem

$$Ax = \lambda Bx, \quad (11.6.53)$$

where we assume that A and B has no common nullspace, i.e.,

$$\mathcal{N}(A) \cap \mathcal{N}(B) = \{0\}. \quad (11.6.54)$$

For solving this eigenvalue problem Ericsson and Ruhe [192] suggested using a similar **spectral transformation**, which we now derive. Subtracting μBx from both sides of (11.6.53) gives

$$(A - \mu B)x = (\lambda - \mu)Bx.$$

From the assumption (11.6.54) it follows that the shift μ can be chosen so that $(A - \mu B)$ is nonsingular. Multiplying both sides by $(A - \mu B)^{-1}$ we obtain the standard eigenvalue problem

$$Cx = \theta x, \quad C = (A - \mu B)^{-1}B, \quad (11.6.55)$$

where $\theta = (\lambda - \mu)^{-1}$. Hence, the eigenvalues of A are related to those of C through

$$\lambda = \mu + 1/\theta. \quad (11.6.56)$$

The eigenvalues θ of C are again related to those of A as in (11.6.56). This transforms the eigenvalues near the shift μ to extremal well separated eigenvalues and rapid convergence to these can be expected. On the other hand, eigenvalues far from μ are mapped near zero and hence damped.

The Arnoldi (or Lanczos) method can be applied to the matrix C as follows. Whenever a matrix-vector product $v = Cv$ is required this is performed in two steps as

$$y = Bx, \quad (A - \mu B)v = y.$$

The solution of the linear system requires that the matrix $C = A - \mu B$ is initially factorized in a sparse LU factorization. Note that this may not be possible for really huge eigenvalue problems. By using a sequence of shifts μ_1, \dots, μ_k , we can obtain eigenvalues in different regions of the complex plane. The eigenvalues of C obtained when transformed back will correspond to eigenvalues in a disk with center at μ .

When A and B are symmetric and B positive semidefinite the eigenvalues are real. To preserve symmetry a small modification of the above procedure is necessary. Assume that the Cholesky factorization

$$B = WW^T, \quad (11.6.57)$$

can be computed. We allow B to be singular so W but may not have full rank, but has linearly independent columns. Then the transformed eigenvalue can be written in symmetric form as

$$Cy = \theta y, \quad C = W^T(A - \mu B)^{-1}W, \quad y = W^T x. \quad (11.6.58)$$

Lanczos method is implicitly applied to the symmetric matrix C in (11.6.58). Matrix-vector products $v = Cx$ are performed as

$$y = Wx, \quad (A - \mu B)z = y, \quad v = W^T z.$$

To solve the linear system in the second step a sparse symmetric factorization $\tilde{A} = LDL^T$ is used. If W is nonsingular, then the eigenvectors of A are obtained by backsubstitution from $x = W^{-T}y$. If W is singular, we have to use use

$$x = (A - \mu B)^{-1}Wy.$$

As a by-product of the factorization of $(A - \mu B)$, we get information about the number of eigenvalues $\lambda < \mu$. If \tilde{A} is positive definite, then D is diagonal and the number is equal to the number of negative elements in D . For the indefinite case, see Section 7.3.4.)

The **rational Krylov method** is a generalization of the shifted and inverted Arnoldi algorithm with several different shifts and factorizations are used; see [506, 507] It 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 a polynomial, operating on a starting vector. The use of several shifts leads to fewer iterations steps. This has to be balances against the costs of computing several factorizations. The rational Krylov method is advantageous to use when many eigenvalues are requested and the eigenvalue problem is illconditioned.

The **Jacobi–Davidson** algorithm is a modification of an algorithm in [136]. Davidson's original method is a projection method with a special choice of how the subspaces are expanded. Let (θ, z) be a Ritz pair approximating and eigenpair of interest. and let $r = (A - \theta I)z$ be the corresponding residual. Then Davidson proposed to increase the current subspace by the vector $(D_A - \theta I)^{-1}r$, where D_A is the diagonal of A . This frequently worked well for the diagonally dominant matrices in certain chemistry problems. On other problems it failed completely.

Sleijpen and van der Vorst [535] suggested restricting the expansion of the current subspace to vectors orthogonal to z . This is similar to an old technique used by Jacobi in 1846, hence the name of the method.

Review Questions

- 6.1** Describe briefly the Rayleigh–Ritz method for computing approximate eigenvalues and eigenvectors from a subspace span (U_k) . What is the connection to the Rayleigh quotient?
- 6.2** The Lanczos method is a particular Rayleigh–Ritz method for real symmetric (Hermitian) matrices. What subspaces are used in this? Describe this algorithms in more detail.

- 6.3** The Ritz values in Lanczos method are usually good approximations for the extreme (largest and smallest) eigenvalues of a symmetric matrix A . To better approximate inner eigenvalues the so-called harmonic Ritz values can be used. What oblique projection method yields these?
- 6.4** (a) When is orthogonality between the Lanczos vectors lost in finite precision? What are the main effects caused by this loss of orthogonality?
 (b) What is the remedy against the loss of orthogonality? How well need the orthogonality be preserved for the Lanczos method in finite precision to behave like the exact algorithm?

Problems

- 6.1** In Lanczos method the harmonic Ritz values are the eigenvalues of the matrix $T_k + \beta_{k+1} T_k^{-1} e_k e_k^T$, Consider the tridiagonal matrix

$$\tilde{T}_k = \begin{pmatrix} T_k & \beta_{k+1} e_k \\ e_k^T \beta_{k+1} & s_k \beta_{k+1}^2 \end{pmatrix}, \quad s_k = e_k^T T_k^{-1} e_k^T, \quad (11.6.59)$$

which 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 \tilde{T}_K ? matrix.

Notes and References

Two classical texts on iterative methods of the 1960s and 1970s are Varga [594] and Young [623]. Among more recent monographs the comprehensive treatment by Saad [518] is recommended. Other outstanding treatments of preconditioners and Krylov space methods are given in Axelsson [19], Greenbaum [281], Meurant [433, 1999], and H. van der Vorst [584, 585]. Domain decomposition methods are surveyed in Toselli and Widlund [570].

An authoritative survey of the development of iterative methods for solving linear systems during the last century is given by Saad and van der Vorst [519]. This contribution also contains references to many classical papers. Barret et al. [32] gives templates for the implementation of many iterative methods. A pdf version of the second edition of this book can be downloaded from <http://www.csm.ornl.gov/rbarret>. Implementations in MATLAB code are also available here.

Section 11.1

The use of overrelaxation techniques was initiated by the work of David Young 1950, who introduced the notion of “consistent orderings” and “property A”. These methods were used successively in the early 1960s to solve linear systems of order 20,000 with a Laplace-type matrix.

An extreme case of blow-up of the SOR method is studied by Hammarling and Wilkinson [304]. A componentwise error analysis for stationary iterative methods

has been given by Higham and Knight [333]. This is extended to singular systems in [334]. A detailed study of using SSOR as a preconditioning was carried out by Axelsson in [17].

Section 11.2

The early history of the conjugate gradient and Lanczos algorithms are detailed in [270]. A comprehensive and up-to-date discussion of Lanczos and conjugate gradient methods for solving linear systems and computing eigenvalues and is given by Meurant [434] (see also Meurent and Strakoš [435]). The superlinear convergence was analyzed in [586]. How to use Krylov subspace methods for linear systems with multiple right-hand sides is discussed by Gutknecht [299]. Krylov subspace solvers are studied by Broyden and Vespucci [80]. The Krylov subspace methods SYMMLQ and MINRES are due to Paige and Saunders [467, 1975].

The CG method was developed independently by Stiefel and Hestenes. Further work was done at the Institute for Numerical Analysis, on the campus of the University of California, at Los Angeles (UCLA). This work was published in 1952 in the seminal paper [321]. In this paper the author acknowledge cooperation with J. B. Rosser, G. E. Forsythe, and L. Paige, who were working at the institute during this period. They also mention that C. Lanczos had developed a closely related method. The CG method was initially considered as a direct method, see Householder [, Sec. 5.7]. As such it is not competitive with Gaussian elimination either in respect of accuracy or number of operations. Therefore, the initial interest in the method soon vanished. It was renewed as Reid [500] showed it was highly efficient if used as an *iterative method for solving large, sparse, well-conditioned linear systems*. The CG method has since had a tremendous impact on scientific computing. Krylov subspace methods, which originated with the conjugate gradient method has been named one of the Top 10 Algorithms of the 20th century.

Section 11.3

The geometrical interpretation of Cimmino's method has inspired many generalizations. For the important problem of finding a solution of a system of linear inequalities $a_i^T x \leq b_i$, $i = 1 : m$, $x \in \mathbf{R}^n$, a Cimmino type iterative method is given in [93].

Iterative methods which at each step require access only to one row of the matrix A are called “row-action methods”. A survey of this class of methods is given by Censor [92]. Kaczmarz's method was rediscovered by around 1970 Gordon et al. [279] and given the name (unconstrained) ART (algebraic reconstruction technique). It is still extensively used in computerized tomography; see Herman [319]. Byrne [88] is unique in combining optimization, convex analysis and approximation theory with a treatment of iterative methods and applications in tomography and imaging.

The implementation of CGLS was first formulated by Stiefel [554]. The stability of CGLS and LSQR methods for least squares problems are studied in [59]. Stopping rules for CGLS method and other Krylov subspace methods when solving ill-posed systems are discussed in the excellent survey by Hanke and Hansen [307], Hanke [305, 1995], and in Hansen [310]. Hybrid methods were first suggested by O'Leary and Simmons [455], and Björck [58].

Section 11.4

The Arnoldi process [13] actually preceded the Lanczos method. The B-Lanczos algorithm was proposed by Lanczos [392] and later in conjugate gradient form by Fletcher [206]. A complete theoretical analysis of the unsymmetric Lanczos process and its relation to the Padé table has been given by Gutknecht [298]. Freund and Nachtigal [220] implement the QMR method with a look-ahead variant of the bi-Lanczos process. The CGS method is due to Sonneveld [539].

Section 11.5

A good introduction to preconditioners for linear systems is given by Saad [518, Chap. 10]. A good survey of the development up to 1985 is given by Axelsson [18]. A more recent survey with focus on algebraic preconditioning methods suitable for general sparse matrices is given by Benzi [46].

Using circulant preconditioners for Toeplitz systems were first proposed by Strang [557]. Interest in this approach took off after the paper by T. Chan [100]. More results are found in R. Chan and Strang [101], and Chan et al. [97]. Preconditioners for Toeplitz systems are surveyed in [98]. An survey of iterative methods for solving Toeplitz methods is given by R. Chan and Jin [96]. A general treatment of circulant matrices is given by Davis [139].

Point and block preconditioners for linear least squares problems are given in Björck [61, Sec. 7.2–7.3]. Of special interest is the two-block case; see Elfving [190].

Section 11.6

The original references to the Rayleigh–Ritz procedure are [496, 503]. How to find approximate solutions and eigenvalue bounds from Krylov subspaces is discussed in [461]. For large eigenvalue problem the Jacobi–Davidson algorithm and the Krylov subspace projection methods are among the most effective. Interest in the Arnoldi's method was revived in the 1980s by a paper of Saad [515].

A priori error bounds for Rayleigh–Ritz approximations were first given by Kaniel [365] and Paige [462] and later improved by Saad [514].

An early sophisticated implementation of simultaneous iteration for symmetric matrices was given by Rutishauser [513].

How to locate spurious copies of eigenvalues in the simple Lanczos method is discussed by Parlett and Reid [480]. Explicit restarts of the Lanczos algorithm was suggested already in 1951 by Karush [370]. Implicit restarts for the Arnoldi (IRAM) and Lanczos methods is due to Sorensen [540]. ARPACK, is an implementation of the implicitly restarted Arnoldi method (IRAM), which has become the most successful and best known public domain software package for large eigenvalue problems. ARPACK can be used for finding a few eigenvalues and eigenvectors of large symmetric or unsymmetric standard or generalized eigenvalue problem; see [40, 541]. A users' guide [404] is available; see also [403]. The Krylov–Schur method by Stewart [551] is mathematically equivalent to IRAM, but has some advantages. First, it is easier to deflate converged Ritz vectors. Second, a potential forward instability of the QR algorithm is avoided.

Selective reorthogonalization of the Lanczos vectors was first suggested by Parlett and Scott [482]. An analysis applicable to several different reorthogonalization schemes in symmetric Lanczos is given by Simon [529]. Implicitly restarted Lanczos

methods for computing a few eigenpairs of large Hermitian eigenvalue problems are treated in [89, 20]. For a description of a MATLAB implementation, see [21].

Implicitly restarted Lanczos bidiagonalization was introduced by Björck et al. in [64]. Rasmus Munk Larsen [395] has developed a software package called PROPACK.⁸² for Lanczos bidiagonalization that incorporates both selective reorthogonalization and implicit restarts. As shown by Hochstenbach [335] the harmonic Ritz values in Lanczos bidiagonalization are the easily obtained from the square bidiagonal matrices in the process. Kokiopoulis et al. [384] 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 [354].

Several software packages are available for solving large-scale symmetric eigenproblems. The block Lanczos code of Grimes *et. al* [283] and its updates are perhaps the most used for structural analysis problems in industrial applications. Many more are freely available on netlib. A selection of those are listed in Sorensen [541].

⁸²<http://soi.stanford.edu/~rmunk/PROPACK/>

Bibliography

- [1] Jan Ole Aasen. On the reduction of a symmetric matrix to tridiagonal form. *BIT*, 11:233–242, 1971. (Cited on p. 84.)
- [2] H. Abou-Kandil, G. Freiling, V. Ionescu, and G. Jank. *Matrix Riccati Equations in Control and Systems Theory*. Birkhäuser, Basel, Switzerland, 2003. (Cited on p. 587.)
- [3] P.-A. Absil, R. Mahony, R. Sepulchre, and P. Van Dooren. A Grassmann–Rayleigh quotient iteration for computing invariant subspaces. *SIAM Review*, 44(1):57–73, 2002. (Cited on p. 588.)
- [4] P.-A. Absil, R. Mahony, R. Sepulchre, and P. Van Dooren. Cubically convergent iterations for invariant subspace computation. *SIAM J. Matrix Anal. Appl.*, 26(1):70–96, 2004. (Cited on p. 588.)
- [5] R. Albaroni and Yair Censor. Block-iterative methods for parallel computation of solutions to convex feasibility problems. *Linear Algebra Appl.*, 120:165–175, 1989. (Cited on p. 643.)
- [6] E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*. Springer Series in Computational Mathematics, Vol. 13. Springer-Verlag, Berlin, 1990. (Cited on pp. 374, 431.)
- [7] G. Ammar and W. Gragg. Superfast solution of real positive definite Toeplitz systems. *SIAM J. Matrix. Anal. Appl.*, 9(1):61–76, 1988. (Cited on p. 187.)
- [8] G. S. Ammar, W. B. Gragg, and L. Reichel. On the eigenproblem for orthogonal matrices. In *Proceedings of the 25th IEEE Conference on Decision and Control*, pages 1963–1966, Piscataway, 1986. IEEE. (Cited on p. 523.)
- [9] Bjarne S. Andersen, Jerzy Waśniewski, and Fred G. Gustavson. A recursive formulation of Cholesky factorization of a packed matrix. *ACM Trans. Math. Software*, 27(2):214–244, 2001. (Cited on p. 145.)
- [10] Edward Anderson, Zhaojun Bai, Christian Bischof, S. Blackford, James W. Demmel, Jack J. Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK Users’ Guide*. SIAM, Philadelphia, PA, third edition, 1999. (Cited on pp. 135, 186, 587.)
- [11] Edward Anderssen, Zhaojun Bai, and Jack J. Dongarra. Generalized QR factorization and its applications. *Linear Algebra Appl.*, 162–164:243–271, 1992. (Cited on p. 351.)
- [12] Mario Arioli, James W. Demmel, and Iain S. Duff. Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Appl.*, 10(2):165–190, 1989. (Cited on pp. 117, 262.)

- [13] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951. (Cited on p. 719.)
- [14] Cleve Ashcraft, Roger G. Grimes, and John G. Lewis. Accurate symmetric indefinite linear system solvers. *SIAM J. Matrix Anal. Appl.*, 20(2):513–561, 1998. (Cited on p. 84.)
- [15] Edgar Asplund. Inverses of matrices $\{a_{ij}\}$ which satisfy $a_{ij} = 0$ for $j > i + p$. *Math. Scand.*, 7:57–60, 1959. (Cited on p. 186.)
- [16] L. Autonne. Sur les matrices hypohermitiennes et les unitaires. *Comptes Rendus de l'Academie Sciences, Paris*, 156:858–860, 1913. (Cited on p. 349.)
- [17] Owe Axelsson. A generalized SSOR method. *BIT*, 12(4):443–467, 1972. (Cited on p. 718.)
- [18] Owe Axelsson. A survey of preconditioned iterative methods for linear systems of algebraic equations. *BIT*, 25(1):166–187, 1985. (Cited on p. 719.)
- [19] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, NY, 1994. (Cited on pp. 609, 633, 648, 717.)
- [20] J. Baglama, D. Calvetti, and L. Reichel. Iterative methods for the computation of a few eigenvalues of a large symmetric matrix. *BIT*, 36(3):400–421, 1996. (Cited on p. 720.)
- [21] J. Baglama, D. Calvetti, and L. Reichel. A MATLAB program for computing a few eigenpairs of a large sparse Hermitian matrix. *ACM Trans. Math. Software*, 29(3):337–348, 2003. (Cited on p. 720.)
- [22] Zhaojun Bai and James W. Demmel. Computing the generalized singular value decomposition. *SIAM J. Sci. Comput.*, 14(6):1464–1486, 1993. (Cited on pp. 540, 555.)
- [23] Zhaojun Bai and James W. Demmel. Using the matrix sign function to compute invariant subspaces. *SIAM J. Matrix Anal. Appl.*, 19(1):205–225, 1998. (Cited on p. 577.)
- [24] Zhaojun Bai, James W. Demmel, Jack J. Dongarra, Axel Ruhe, and Henk A. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, PA, 2000. (Cited on p. 587.)
- [25] Zhaojun Bai, James W. Demmel, and A. McKenney. On computing condition numbers for the nonsymmetric eigenproblem. *ACM Trans. Math. Software*, 19(1):202–223, 1993. (Cited on p. 447.)
- [26] Zhaojun Bai and Hongyuan Zha. A new preprocessing algorithm for the computation of the generalized singular value decomposition. *SIAM J. Sci. Comput.*, 14(4):1007–1012, 1993. (Cited on p. 555.)
- [27] Zhong.Zhi Bai, Iain S. Duff, and Andrew J. Wathen. A class of incomplete orthogonal factorization methods I: Methods and theories. *BIT*, 41(1):53–70, 2001. (Cited on p. 684.)
- [28] Vincent A. Barker, L. S. Blackford, J. Dongarra, Jeremy Du Croz, Sven Hammarling, M. Marinova, Jerzy Waśniewski, and Plamen Yalamov. *LAPACK 95 Users' Guide*. SIAM, Philadelphia, PA, 2001. (Cited on p. 186.)
- [29] Jesse Barlow and James W. Demmel. Computing accurate eigensystems of scaled diagonally dominant matrices. *SIAM J. Numer. Anal.*, 27:762–791, 1990. (Cited on p. 507.)

- [30] Jesse L. Barlow. More accurate bidiagonal reduction algorithm for computing the singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 23(3):761–798, 2002. (Cited on p. 352.)
- [31] Jesse L. Barlow, Nela Bosner, and Zlatko Drmač. A new stable bidiagonal reduction algorithm. *Linear Algebra Appl.*, 397:35–84, 2005. (Cited on p. 352.)
- [32] R. Barret, M. W. Berry, Tony F. Chan, James W. Demmel, J. Donato, Jack Dongarra, V. Eijkhout, R. Pozo, C. Romine, and Henk A. van der Vorst, editors. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, second edition, 1993. (Cited on p. 717.)
- [33] Anders Barrlund. Perturbation bounds on the polar decomposition. *BIT*, 20(1):101–113, 1990. (Cited on p. 333.)
- [34] I. Barrodale and F. D. K. Roberts. An improved algorithm for discrete ℓ_1 linear approximation. *SIAM J. Numer. Anal.*, 10:839–848, 1973. (Cited on p. 351.)
- [35] R. H. Bartels, Andrew R. Conn, and J. W. Sinclair. Minimization techniques for piecewise differentiable functions: The ℓ_1 solution to an overdetermined linear system. *SIAM J. Numer. Anal.*, 15:224–241, 1978. (Cited on p. 351.)
- [36] R. H. Bartels, J. Stoer, and Ch. Zenger. A realization of the simplex method based on triangular decompositions. In James H. Wilkinson and C. Reinsch, editors, *Handbook for Automatic Computation. Vol. II, Linear Algebra*, pages 152–190. Springer-Verlag, New York, 1971. (Cited on p. 416.)
- [37] Richard H. Bartels and G. W. Stewart. Algorithm 432: Solution of the equation $AX + XB = C$. *Comm. ACM*, 15:820–826, 1972. (Cited on p. 450.)
- [38] F. L. Bauer. Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischer Eigenwertprobleme. *Z. Angew. Math. Phys.*, 8:214–235, 1957. (Cited on p. 488.)
- [39] F. L. Bauer. Genauigkeitsfragen bei der Lösung linearer Gleichungssysteme. *Z. Angew. Math. Mech.*, 46:7:409–421, 1966. (Cited on p. 186.)
- [40] Christopher A. Beattie, Mark Embree, and D. C. Sorensen. Convergence of polynomial restart Krylov methods for eigenvalue computations. *SIAM Review*, 47:492–515, 2002. (Cited on p. 719.)
- [41] Richard Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, New York, second edition, 1970. Republished by SIAM, Philadelphia, PA, 1997. (Cited on pp. 184, 185.)
- [42] E. Beltrami. Sulle funzioni bilineari. *Giornale di Matematiche ad Uso degli Studenti Delle Università*, 11:98–106, 1873. (Cited on p. 349.)
- [43] Adi Ben-Israel and T. N. E. Greville. Some topics in generalized inverses of matrices. In M. Z. Nashed, editor, *Generalized Inverses and Applications*, pages 125–147. Academic Press, Inc., New York, 1976. (Cited on p. 349.)
- [44] Adi Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications*. Springer-Verlag, Berlin–Heidelberg–New York, 2003. (Cited on p. 349.)
- [45] Commandant Benoit. Sur la méthode de résolution des équations normales, etc. (procédés du commandant Cholesky). *Bull. Géodesique*, 2:67–77, 1924. (Cited on p. 214.)
- [46] Michele Benzi. Preconditioning techniques for large linear systems. *J. Comput. Phys.*, 182(3–6):418–477, 2002. (Cited on p. 719.)

- [47] Michele Benzi. Gianfranco Cimmino's contribution to Numerical Mathematics. In *Conferenza in Memoria di Gianfranco Cimmino*, pages 87–109, Bologna, 2005. Tecnoprint. (Cited on p. 644.)
- [48] Michele Benzi and Carl D. Meyer. A direct projection method for sparse linear systems. *SIAM J. Sci. Comput.*, 16(5):1159–1176, 1995. (Cited on pp. 685, 686.)
- [49] Michele Benzi and M. Túma. A sparse approximate inverse preconditioner for the nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19(3):968–994, 1998. (Cited on p. 686.)
- [50] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. SIAM, Philadelphia, PA, 1994. Corrected republication, with supplement, of work first published in 1979 by Academic Press. (Cited on p. 453.)
- [51] Rajendra Bhatia. *Perturbation Bounds for Matrix Eigenvalues*. Pitman Research Notes in Mathematics. Longman Scientific & Technical, Harlow, Essex, 1987. (Cited on p. 587.)
- [52] Rajendra Bhatia. *Matrix Analysis*. Graduate Texts in Mathematics, 169. Springer-Verlag, New York, 1997. ISBN 0-387-94846-5. (Cited on p. 587.)
- [53] David Bindel, James W. Demmel, and William Kahan. On computing Givens rotations reliably and efficiently. *ACM Trans. Math. Software*, 28(2):206–238, 2002. (Cited on p. 350.)
- [54] R. E. Bixby. Implementing the simplex method: The initial basis. *ORSA J. Comput.*, 4:267–284, 1992. (Cited on p. 418.)
- [55] Arne Bjerhammar. Rectangular reciprocal matrices with special reference to geodetic calculations. *Bulletin Géodésique*, 52:118–220, 1951. (Cited on p. 200.)
- [56] Arne Bjerhammar. *Theory of Errors and Generalized Inverse Matrices*. Elsevier Scientific Publishing Co., Amsterdam, 1973. (Cited on p. 349.)
- [57] Å. Björck. Solving linear least squares problems by Gram–Schmidt orthogonalization. *BIT*, 7(1):1–21, 1967. (Cited on pp. 254, 350.)
- [58] Å. Björck. A bidiagonalization algorithm for solving ill-posed systems of linear equations. *BIT*, 28:659–670, 1988. (Cited on p. 718.)
- [59] Å. Björck, T. Elfving, and Z. Strakoš. Stability of conjugate gradient and Lanczos methods for linear least squares problems. *SIAM J. Matrix Anal. Appl.*, 19:3:720–736, 1998. (Cited on p. 718.)
- [60] Åke Björck. Numerics of Gram–Schmidt orthogonalization. *Linear Algebra Appl.*, 197–198:297–316, 1994. (Cited on p. 350.)
- [61] Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA, 1996. (Cited on pp. 219, 275, 277, 306, 348, 349, 655, 719.)
- [62] Åke Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM J. Numer. Anal.*, 8(2):358–364, 1973. (Cited on p. 574.)
- [63] Åke Björck and Gene H. Golub. Iterative refinement of linear least squares solution by Householder transformation. *BIT*, 7(4):322–337, 1967. (Cited on p. 322.)
- [64] Åke Björck, Eric Grimme, and Paul Van Dooren. An implicit shift bidiagonalization algorithm for ill-posed systems. *BIT*, 34(4):510–534, 1994. (Cited on p. 720.)

- [65] Åke Björck and Christopher C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, 13(1):176–190, 1992. (Cited on p. 258.)
- [66] L. S. Blackford, J. Choi, A. Cleary, E. D’Azevedo, J. Demmel, I. Dhillon, J. Dongarra, Sven Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users’ Guide*. SIAM, Philadelphia, PA, 1997. (Cited on p. 136.)
- [67] Paul T. Boggs, R. H. Byrd, and R. B. Schnabel. A stable and efficient algorithm for nonlinear orthogonal regression. *SIAM J. Sci. Stat. Comput.*, 8:1052–1078, 1987. (Cited on p. 396.)
- [68] Paul T. Boggs, J. R. Donaldson, R. H. Byrd, and R. B. Schnabel. Algorithm 676. ODRPACK: Software for weighted orthogonal distance regression. *ACM Trans. Math. Software*, 15:348–364, 1989. (Cited on p. 396.)
- [69] A. W. Bojanczyk and R. P. Brent. Parallel solution of certain Toeplitz least-squares problems. *Linear Algebra Appl.*, 77:43–60, 1986. (Cited on p. 327.)
- [70] A. W. Bojanczyk, R. P. Brent, P. Van Dooren, and F. de Hoog. A note on downdating the Cholesky factorization. *SIAM J. Sci. Stat. Comput.*, 8:210–221, 1987. (Cited on p. 351.)
- [71] Carl de Boor. An empty exercise. *ACM SIGNUM Newsletter.*, 25:2–6, 1990. (Cited on p. 662.)
- [72] Carl de Boor. On Pták’s derivation of the Jordan normal form. *Linear Algebra Appl.*, 310:9–10, 2000. (Cited on p. 587.)
- [73] Carl de Boor and Allan Pinkus. Backward error analysis for totally positive linear systems. *Numer. Math.*, 27:485–490, 1977. (Cited on p. 121.)
- [74] Tibor Boros, Thomas Kailath, and Vadim Olshevsky. A fast parallel Björck–Pereyra-type algorithm for parallel solution of Cauchy linear systems. *Linear Algebra Appl.*, 302–303:265–293, 1999. (Cited on p. 181.)
- [75] Z. Bothe. Bounds for rounding errors in the Gaussian elimination for band systems. *J. Inst. Maths. Applies.*, 16:133–142, 1975. (Cited on p. 95.)
- [76] K. Braman, Ralph Byers, and R. Mathias. The multishift QR algorithm. Part I. Maintaining well-focused shifts and level 3 performance. *SIAM J. Matrix. Anal. Appl.*, 23(4):929–947, 2002. (Cited on p. 588.)
- [77] K. Braman, Ralph Byers, and R. Mathias. The multishift QR algorithm. Part II. Aggressive early deflation. *SIAM J. Matrix. Anal. Appl.*, 23(4):948–973, 2002. (Cited on p. 588.)
- [78] Richard P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973. Reprinted by Dover Publications, Mineola, NY, 2002. (Cited on p. 432.)
- [79] Richard P. Brent, Colin Percival, and Paul Zimmermann. Error bounds on complex floating-point multiplication. *Math. Comp.*, posted on January 24, 2007, 2007. to appear in print. (Cited on p. 36.)
- [80] Charles Broyden and Maria Theresa Vespucci. *Krylov Solvers for Linear Algebraic Systems*. Studies in Computational Mathematics 11. Elsevier, Amsterdam, 2004. (Cited on p. 718.)

- [81] Kurt Bryan and Tanya Leise. The \$25,000,000,000 eigenvector: The linear algebra behind Google. *SIAM Review.*, 48(3):569–581, 2006. (Cited on p. 477.)
- [82] J. R. Bunch, Ch. P. Nielsen, and D. C. Sorenson. Rank-one modifications of the symmetric tridiagonal eigenproblem. *Numer. Math.*, 31(1):31–48, 1978. (Cited on p. 528.)
- [83] James R. Bunch. Stability of methods for solving Toeplitz systems of equations. *SIAM J. Sci. Stat. Comp.*, 6(2):349–364, 1985. (Cited on p. 187.)
- [84] James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Math. Comp.*, 31:163–179, 1977. (Cited on p. 83.)
- [85] James R. Bunch, Linda Kaufman, and Beresford N. Parlett. Decomposition of a symmetric matrix. *Numer. Math.*, 27(1):95–110, 1976. (Cited on p. 185.)
- [86] Angelica Bunse-Gerstner, Ralph Byers, and Volker Mehrmann. A chart of numerical methods for structured eigenvalue problems. *SIAM J. Matrix. Anal. Appl.*, 13(2):419–453, 1992. (Cited on p. 589.)
- [87] Peter Businger and G. H. Golub. Linear least squares solutions by Householder transformations. *Numer. Math.*, 7:269–276, 1965. Also published as Contribution I/8 in the Handbook. (Cited on p. 350.)
- [88] Charles L. Byrne. *Applied Iterative Methods*. A. K. Peters, New York, second edition, 2007. (Cited on p. 718.)
- [89] D. Calvetti, L. Reichel, and D. Sorensen. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Electron. Trans. Numer. Anal.*, 2:1–21, 1994. (Cited on p. 720.)
- [90] A. Cauchy. *Mémoire sur les fonctions alternées et sur les sommes alternées*. Exercises d’Analyse et de Phys. Math., Vol. 2, Paris, 1841. (Cited on p. 177.)
- [91] Arthur Cayley. A memoir on the theory of matrices. *Philos. Trans. Roy. Soc. London*, 148:17–37, 1858. (Cited on p. 2.)
- [92] Yair Censor. Row-action methods for huge and sparse systems and their applications. *SIAM Review*, 23:444–466, 1981. (Cited on p. 718.)
- [93] Yair Censor and Tommy Elfving. New methods for linear inequalities. *Linear Algebra Appl.*, 42:199–211, 1981. (Cited on p. 718.)
- [94] Yair Censor and S. A. Zenios. *Parallel Optimization. Theory, Algorithms, and Applications*. Oxford University Press, Oxford, 1997. (Cited on p. 432.)
- [95] J. M. Chambers. Regression updating. *J. Amer. Statist. Assoc.*, 66:744–748, 1971. (Cited on p. 320.)
- [96] Raymond H. Chan and Xiao-Qing Jin. *An Introduction to Iterative Toeplitz Solvers*. SIAM, Philadelphia, PA, 2007. (Cited on p. 719.)
- [97] Raymond H. Chan, James G. Nagy, and Robert J. Plemmons. Generalization of Strang’s preconditioner with application to Toeplitz least squares problems. *Numer. Linear Algebra Appl.*, 3(1):45–64, 1996. (Cited on p. 719.)
- [98] Raymond H. Chan and Michael K. Ng. Conjugate gradient methods for Toeplitz systems. *SIAM Review.*, 38(3):427–482, 1996. (Cited on p. 719.)
- [99] Tony F. Chan. Rank revealing QR factorizations. *Linear Algebra Appl.*, 88/89:67–82, 1987. (Cited on p. 273.)

- [100] Tony F. Chan. An optimal circulant preconditioner for Toeplitz systems. *SIAM J. Sci. Stat. Comp.*, 9(4):766–771, 1988. (Cited on pp. 692, 719.)
- [101] Tony F. Chan. Toeplitz equations by conjugate gradients with circulant preconditioner. *SIAM J. Sci. Stat. Comp.*, 10(1):104–119, 1989. (Cited on p. 719.)
- [102] Tony F. Chan and Per Christian Hansen. Some applications of the rank revealing QR factorization. *SIAM J. Sci. Stat. Comput.*, 13(3):727–741, 1992. (Cited on p. 273.)
- [103] Tony F. Chan and Per Christian Hansen. Low-rank revealing QR factorizations. *Numer. Linear Algebra Appl.*, 1:33–44, 1994. (Cited on p. 271.)
- [104] S. Chandrasekaran and Ming Gu. Fast and stable algorithms for banded plus semiseparable systems of linear equations. *SIAM J. Matrix Anal. Appl.*, 25(2):373–384, 1985. (Cited on p. 182.)
- [105] S. Chandrasekaran and I. C. F. Ipsen. On rank-revealing factorizations. *SIAM J. Matrix Anal. Appl.*, 15:592–622, 1994. (Cited on p. 273.)
- [106] S. Chandrasekaran and Ilse C. F. Ipsen. Analysis of a QR algorithm for computing singular values. *SIAM J. Matrix Anal. Appl.*, 16(2):520–535, 1995. (Cited on p. 588.)
- [107] F. Chatelin. Simultaneous Newton’s corrections for the eigenproblem. In *Defect Correction Methods*, Comput. Suppl. 5, pages 67–74. Springer Verlag, Vienna, 1984. (Cited on p. 588.)
- [108] Eleanor Chu and Alan George. A note on estimating the error in Gaussian elimination without pivoting. *ACM SIGNUM Newsletter*, 20:2:2–7, 1985. (Cited on p. 121.)
- [109] M. Chu, R. Funderlic, and Gene H. Golub. A rank-one reduction formula and its application to matrix factorization. *SIAM Review*, 37(3):512–530, 1995. (Cited on p. 41.)
- [110] Philippe G. Ciarlet. *Introduction to Numerical Linear Algebra and Optimization*. Cambridge University Press, Cambridge, UK, 1989. (Cited on p. 26.)
- [111] Gianfranco Cimmino. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari. *Ricerca Sci. II*, 9:326–333, 1938. (Cited on p. 644.)
- [112] D. I. Clark and M. R. Osborne. Finite algorithms for Huber’s M-estimator. *SIAM J. Sci. Statist. Comput.*, 7:72–85, 1986. (Cited on p. 352.)
- [113] A. K. Cline. The transformation of a quadratic programming problem into solvable form. Tech. Report ICASE 75-14, NASA, Langley Research Center, Hampton, VA., 1975. (Cited on pp. 425, 426.)
- [114] Alan K. Cline, Cleve B. Moler, G. W. Stewart, and James H. Wilkinson. An estimate for the condition number of a matrix. *SIAM J. Numer. Anal.*, 16(2):368–375, 1979. (Cited on p. 114.)
- [115] Paul Concus, Gene H. Golub, and Gérard Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.*, 6:220–252, 1985. (Cited on p. 687.)
- [116] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods*. SIAM Publications., Philadelphia, PA, 2000. (Cited on p. 432.)
- [117] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Derivative-Free Optimization*. SIAM Publications., Philadelphia, PA, 2009. (Cited on p. 431.)

- [118] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9:251–280, 1990. (Cited on p. 32.)
- [119] G. Corliss, C. Faure, Andreas Griewank, L. Hascoet, and U. Naumann. *Automatic Differentiation of Algorithm. From Simulation to Optimization*. Springer-Verlag, Berlin, 2002. (Cited on p. 431.)
- [120] Anthony J. Cox and Nicholas J. Higham. Stability of Householder qr factorization for weighted least squares problems. In D. F. Griffiths, D. J. Higham, and G. A. Watson, editors, *Numerical Analysis 1997: Proceedings of the 17th Dundee Biennial Conference*, Pitman Research Notes in mathematics, vol. 380, pages 57–73. Longman Scientific and Technical, Harlow, Essex, UK, 1998. (Cited on p. 317.)
- [121] Anthony J. Cox and Nicholas J. Higham. Backward error bounds for constrained least squares problems. *BIT*, 39(2):210–227, 1999. (Cited on p. 350.)
- [122] Maurice G. Cox. The least-squares solution of linear equations with block-angular observation matrix. In M. G. Cox and S. J. Hammarling, editors, *Reliable Numerical Computation*, pages 227–240. Oxford University Press, UK, 1990. (Cited on p. 297.)
- [123] E. J. Craig. The n -step iteration procedure. *J. Math. Phys.*, 34:65–73, 1955. (Cited on p. 650.)
- [124] C. R. Crawford. Reduction of a band-symmetric generalized eigenvalue problem. *Comm. Assos. Comput. Mach.*, 16:41–44, 1973. (Cited on p. 550.)
- [125] C. R. Crawford and Yiu Sang Moon. Finding a positive definite linear combination of two Hermitian matrices. *Linear Algebra Appl.*, 51:37–48, 1983. (Cited on p. 549.)
- [126] P. D. Crout. A short method for evaluating determinants and solving systems of linear equations with real or complex coefficients. *Trans. Amer. Inst. Elec. Engng.*, 60:1235–1240, 1941. (Cited on p. 61.)
- [127] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36(2):177–195, 1981. (Cited on pp. 526, 544.)
- [128] Alan R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974. (Cited on p. 376.)
- [129] G. Cybenko. The numerical stability of the Levinson–Durbin algorithm for Toeplitz systems of equations. *SIAM J. Sci. Stat. Comp.*, 1(3):303–309, 1980. (Cited on p. 177.)
- [130] G. Dahlquist. *Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations*. PhD thesis, Department of Mathematics, Uppsala University, Uppsala, Sweden, 1958. Also available as Trans. Royal Inst. Technology, Stockholm, No. 130. (Cited on p. 589.)
- [131] G. Dahlquist, S. C. Eisenstat, and Gene H. Golub. Bounds for the error in linear systems of equations using the theory of moments. *J. Math. Anal. Appl.*, 37:151–166, 1972. (Cited on p. 578.)
- [132] Germund Dahlquist and Åke Björck. *Numerical Methods in Scientific Computing*, volume I. SIAM, Philadelphia, PA, 2008. (Cited on pp. 19, 611, 611, 637.)
- [133] Germund Dahlquist, Gene H. Golub, and S. G. Nash. Bounds for the error in linear systems. In R. Hettich, editor, *Workshop on Semi-Infinite Programming*, pages 154–172, Berlin, 1978. Springer-Verlag. (Cited on p. 578.)
- [134] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comp.*, 30:772–795, 1976. (Cited on p. 276.)

- [135] George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, second edition, 1965. (Cited on p. 432.)
- [136] E. R. Davidson. 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. (Cited on p. 716.)
- [137] Philip I. Davies and Nicholas J. Higham. A Schur–Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl.*, 25(2):464–485, 2003. (Cited on p. 565.)
- [138] Philip I. Davies, Nicholas J. Higham, and Françoise Tisseur. 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.
- [139] P. J. Davis. *Circulant Matrices*. Wiley & Sons, New York, 1979. (Cited on p. 719.)
- [140] Timothy A. Davis. *Direct Methods for Sparse Linear Systems*, volume 2 of *Fundamental of Algorithms*. SIAM, Philadelphia, PA, 2006. (Cited on p. 187.)
- [141] F. R. de Hoog and R. M. M. Mattheij. Subset selection for matrices. *Linear Algebra Appl.*, 422:349–359, 2007. (Cited on p. 350.)
- [142] A. de la Garza. An iterative method for solving systems of linear equations. Report K-731, Oak Ridge Gaseous Diffusion Plant, Oak Ridge, TN, 1951. (Cited on p. 646.)
- [143] S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Math. Comput.*, 43:491, 1984. (Cited on p. 684.)
- [144] James W. Demmel. Three ways for refining estimates of invariant subspaces. *Computing*, 38:43–57, 1987. (Cited on p. 588.)
- [145] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997. (Cited on p. 184.)
- [146] James W. Demmel, Ming Gu, Stanley Eisenstat, Ivan Slapničar, Kresimir Verselić, and Zlatko Drmač. Computing the singular value decomposition with high relative accuracy. *Linear Algebra Appl.*, 299:21–80, 1999. (Cited on p. 588.)
- [147] James W. Demmel, Yozo Hida, William Kahan, Xiaoye S. Li, Sonil Mukherjee, and E. Jason Riedy. Error bounds from extra-precise iterative refinement. *ACM Trans. Math. Software*, 32(2):325–351, 2006. (Cited on p. 126.)
- [148] James W. Demmel, Yozo Hida, Xiaoye S. Li, and E. Jason Riedy. Extra-precise iterative refinement for overdetermined least squares problems. Research Report, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, 2007. (Cited on p. 264.)
- [149] James W. Demmel, Nicholas J. Higham, and Robert S. Schreiber. Stability of block LU factorizations. *Numer. Linear Algebra Appl.*, 2:173–190, 1995. (Cited on p. 138.)
- [150] James W. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Statist. Comput.*, 11(5):873–912, 1990. (Cited on pp. 516, 519, 520, 540, 588.)
- [151] James W. Demmel and Plamen Koev. The accurate and efficient solution of a totally positive generalized Vandermonde linear system. *SIAM J. Matrix Anal. Appl.*, 27:142–152, 2005. (Cited on p. 181.)
- [152] James W. Demmel and Krešimir Veselić. Jacobi’s method is more accurate than QR. *SIAM J. Matrix Anal. Appl.*, 13(4):1204–1245, 1992. (Cited on p. 535.)

- [153] Eugene D. Denman and Alex N. Beavers. The matrix sign function and computations in systems. *Appl. Math. Comput.*, 2:63–94, 1976. (Cited on p. 571.)
- [154] John E. Dennis, David M. Gay, and R. E. Welsh. An adaptive nonlinear least-squares algorithm. *ACM. Trans. Math. Software*, 7:348–368, 1981. (Cited on p. 432.)
- [155] John E. Dennis and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Review*, 19:46–89, 1977. (Cited on p. 372.)
- [156] John E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Number 16 in Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1995. (Cited on pp. 361, 389, 431.)
- [157] Henk A. Van der Vorst. 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. (Cited on p. 672.)
- [158] P. Deuflhard. *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*, volume 35 of *Computational Mathematics*. Springer, New York, second edition, 2006. (Cited on pp. 364, 372.)
- [159] Inderjit S. Dhillon. *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*. Ph.D. thesis, University of California, Berkeley, CA, 1997. (Cited on p. 589.)
- [160] Inderjit S. Dhillon. Reliable computation of the condition number of a tridiagonal matrix in $O(n)$ time. *SIAM J. Matrix Anal. Appl.*, 19(3):776–796, 1998. (Cited on p. 186.)
- [161] Inderjit S. Dhillon and Beresford N. Parlett. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra Appl.*, 387:1–28, 2004. (Cited on p. 589.)
- [162] Inderjit S. Dhillon and Beresford N. Parlett. Orthogonal eigenvectors and relative gaps. *SIAM J. Matrix Anal. Appl.*, 25(3):858–899, 2004. (Cited on p. 589.)
- [163] J. Dieudonné. *Foundations of Modern Analysis*. Academic Press, New York, NY, 1961. (Cited on pp. 354, 356.)
- [164] Jack J. Dongarra, James R. Bunch, Cleve B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. SIAM, Philadelphia, PA, 1979. (Cited on pp. 115, 133, 186, 277, 430.)
- [165] Jack J. Dongarra, Jeremy Du Croz, Iain S. Duff, and Sven Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 16:1–17, 1988. (Cited on p. 134.)
- [166] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and R. J. Hanson. A extended set of Fortran Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 14:1–17, 1988. (Cited on p. 133.)
- [167] Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. *Numerical Linear Algebra for High Performance Computers*. SIAM, Philadelphia, PA, 1998. (Cited on p. 186.)
- [168] Jack J. Dongarra and Victor Eijkhout. Numerical linear algebra and software. *J. Comput. Appl. Math.*, 123:489–514, 2000. (Cited on p. 186.)
- [169] Jack J. Dongarra, Sven Hammarling, and Danny C. Sorensen. Block reduction of matrices to condensed forms for eigenvalue computation. *J. Assos. Comput. Mach.*, 27:215–227, 1989. (Cited on p. 496.)

- [170] Jack J. Dongarra and Danny C. Sorensen. A fully parallel algorithmic for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8(2):139–154, 1987. (Cited on p. 526.)
- [171] Jack J. Dongarra and D. W. Walker. Software libraries for linear algebra computations on high performance computers. *SIAM Review.*, 37:151–180, 1995. (Cited on p. 186.)
- [172] M. H. Doolittle. Method employed in the solution of normal equations and the adjustment of a triangularization. In *U. S. Coast and Geodetic Survey Report*, pages 115–120. 1878. (Cited on p. 61.)
- [173] Jeremy Du Croz and Nicholas J. Higham. Stability of methods for matrix inversion. *IMA J. Numer. Anal.*, 12:1–19, 1992. (Cited on p. 67.)
- [174] Iain S. Duff. Sparse numerical linear algebra: Direct methods and preconditioning. In Iain S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 27–62. Oxford University Press, London, UK, 1997. (Cited on p. 187.)
- [175] Iain S. Duff, A. M. Erisman, C. W. Gear, and John K. Reid. Sparsity structure and Gaussian elimination. *SIGNUM Newslett.*, 23(2), 1988. (Cited on p. 98.)
- [176] Iain S. Duff, A. M. Erisman, and John K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, London, 1986. (Cited on pp. 168, 187.)
- [177] Iain S. Duff, Roger G. Grimes, and John G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Software*, 15(1):1–14, 1989. (Cited on p. 151.)
- [178] Iain S. Duff and John K. Reid. An implementation of Tarjan’s algorithm for the block triangularization of a matrix. *ACM Trans. Math. Software*, 4:137–147, 1978. (Cited on p. 172.)
- [179] Iain S. Duff and John K. Reid. The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Trans. Math. Software*, 9:302–325, 1983. (Cited on pp. 167, 307.)
- [180] A. L. Dulmage and N. S. Mendelsohn. Two algorithms for bipartite graphs. *J. Soc. Indust. Appl. Math.*, 11:183–194, 1963. (Cited on p. 351.)
- [181] J. Durbin. The fitting of time-series models. *Rev. Internat. Statist. Inst.*, 28:229–249, 1959. (Cited on p. 175.)
- [182] P. J. Eberlein and C. P. Huang. Global convergence of the QR algorithm for unitary matrices with some results for normal matrices. *SIAM J. Numer. Anal.*, 12(1):97–104, 1975. (Cited on p. 523.)
- [183] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936. (Cited on pp. 206, 349.)
- [184] A. Edelman, T. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix. Anal. Appl.*, 20(2):303–353, 1999. (Cited on pp. 587, 588.)
- [185] Stanley C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Scherman. The Yale sparse matrix package, 1. The symmetric code. *Internat J. Numer. Meth. Engrg.*, 18:1145–1151, 1982. (Cited on p. 186.)
- [186] H. Ekblom. A new algorithm for the Huber estimator in linear models. *BIT*, 28:60–76, 1988. (Cited on p. 352.)
- [187] L. Eldén. A weighted pseudoinverse, generalized singular values, and constrained least squares problems. *BIT*, 22:487–502, 1982. (Cited on pp. 323, 326.)

- [188] Lars Eldén. 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. (Cited on pp. 327, 329.)
- [189] Lars Eldén and Berkant Savas. A Newton–Grassman method for computing the best multi-linear rank- (r_1, r_2, r_3) approximation of a tensor. Technical Report LITH-MAT-R-2007-6, Linköping University, SE-581 83 Linköping, Sweden, April 2007. submitted to SIMAX. (Cited on p. 313.)
- [190] Tommy Elfving. Block-iterative methods for consistent and inconsistent equations. *Numer. Math.*, 35:1–12, 1980. (Cited on pp. 648, 719.)
- [191] Tommy Elfving and Ingegerd Skoglund. A direct method for a special regularized least squares problem. Technical Report LITH-MAT-R-2007-1, Linköping University, SE-581 83 Linköping, Sweden, March 2007. submitted. (Cited on p. 306.)
- [192] Thomas Ericsson and Axel Ruhe. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Math. Comp.*, 35(152):1251–1268, 1980. (Cited on p. 715.)
- [193] Jerry Eriksson. Quasi-Nerwton methods for nonlinear least squares focusing on curvature. *BIT*, 39(2):228–254, 1999. (Cited on p. 431.)
- [194] Vance Faber and Thomas Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21(2):352–362, 1984. (Cited on p. 660.)
- [195] D. K. Faddeev and V. N. Faddeeva. *Computational Methods of Linear Algebra*. W. H. Freeman, San Francisco, CA, 1963. (Cited on p. 185.)
- [196] V. N. Faddeeva. *Computational Methods of Linear Algebra*. Dover, Mineola, NY, 1959. (Cited on p. 626.)
- [197] Ky Fan and Alan J. Hoffman. Some metric inequalities in the space of matrices. *Proc. Amer. Math. Soc.*, 6:111–116, 1955. (Cited on p. 206.)
- [198] Heike Fassbender, D. Steven Mackey, and Niloufer Mackey. Hamilton and Jacobi come full circle: Jacobi algorithms for structured Hamiltonian eigenproblems. *Linear Algebra Appl.*, 332–334:37–80, 2001. (Cited on p. 555.)
- [199] K. V. Fernando. Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices. *SIAM J. Matrix Anal. Appl.*, 20(2):373–399, 1998. (Cited on p. 534.)
- [200] K. V. Fernando and Beresford N. Parlett. Accurate singular values and differential qd algorithms. *Numer. Math.*, 67:191–229, 1994. (Cited on pp. 588, 589.)
- [201] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley, New York, 1968. (Cited on p. 432.)
- [202] Ricardo D. Fierro and Per Christian Hansen. UTV Expansion Pack: Special-purpose rank-revealing algorithms. *Numerical Algorithms*, 40:47–66, 2005. (Cited on p. 350.)
- [203] Ricardo D. Fierro, Per Christian Hansen, and P. S. Hansen. UTV Tools: Matlab templates for rank-revealing UTV decompositions. *Numerical Algorithms*, 20:165–194, 1999. (Cited on p. 271.)
- [204] Bernt Fischer and Roland Freund. On the constrained Chebyshev approximation problem on ellipses. *J. Approx. Theory*, 62:297–315, 1990. (Cited on p. 712.)
- [205] E. Fischer. Über quadratische Formen mit reeller Koeffizienten. *Monatshefte Math. Phys.*, 16:234–249, 1905. (Cited on p. 463.)

- [206] Roger Fletcher. Conjugate gradient methods for indefinite systems. In G. A. Watson, editor, *Numerical Analysis 1975: Proceedings of the Dundee Conference on Numerical Analysis 1975*, Lecture Notes in Mathematics Vol. 506, pages 73–89, Berlin, 1976. Springer-Verlag. (Cited on pp. 185, 719.)
- [207] Roger Fletcher. *Practical Methods of Optimization*. John Wiley, New York, second edition, 1987. (Cited on pp. 423, 432.)
- [208] Roger Fletcher and Danny C. Sorensen. An algorithmic derivation of the Jordan canonical form. *American Mathematical Monthly*, 90, 1983. (Cited on p. 442.)
- [209] IEEE Standard for Binary Floating-Point Arithmetic. ANSI/IEEE Standard 754-1985. *SIGPLAN Notices*, 22(2):9–25, 1987. (Cited on p. 32.)
- [210] Anders Forsgren. An elementray proof of optimality conditions for linear programming. Tech. Report TRITA-MAT-2008-OS6, Department of Mathematics, Royal Institute of Technolgy, Stockholm, 2008. (Cited on p. 409.)
- [211] Anders Forsgren, Philip E. Gill, and Margaret H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44:4:525–597, 2002. (Cited on p. 432.)
- [212] George E. Forsythe. Tentativ classification of methods and bibliography of on solving systems of linear equations. *Nat. Bur. Standards Appl. Math. Ser.*, 29:1–28, 1953. (Cited on p. 185.)
- [213] George E. Forsythe. Algorithms for scientific computation. *Comm. ACM.*, 9:255–256, 1966. (Cited on p. 353.)
- [214] George E. Forsythe and Peter Henrici. The cyclic Jacobi method for computing the principal values of a complex matrix. *Trans. Amer. Math. Soc.*, 94:1–23, 1960. (Cited on p. 537.)
- [215] George E. Forsythe and Cleve B. Moler. *Computer Solution of Linear Algebraic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1967. (Cited on p. 65.)
- [216] Leslie Foster and Rajesh Kommu. Algorithm 853. An efficient algorithm for solving rank-deficient least squares problems. *ACM Trans. Math. Software*, 32(1):157–165, 2006. (Cited on p. 350.)
- [217] J. G. F. Francis. The QR transformation. Part I and II. *Computer J.*, 4:265–271, 332–345, 1961–1962. (Cited on p. 492.)
- [218] W. L. Frank. Computing eigenvalues of complex matrices by determinant evaluation and by methods of danilewski and wielandt. *J. SIAM*, 6:378–392, 1958. (Cited on p. 503.)
- [219] Roland W. Freund, Gene H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1992. (Cited on p. 669.)
- [220] Roland W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991. (Cited on p. 719.)
- [221] Carl-Erik Fröberg. *Numerical Mathematics. Theory and Computer Applications*. Benjamin/Cummings, Menlo Park, CA, 1985. (Cited on pp. 479, 585.)
- [222] Walter Gander. Least squares with a quadratic constraint. *Numer. Math.*, 36:291–307, 1981. (Cited on p. 324.)
- [223] Walter Gander. Algorithms for the polar decomposition. *SIAM J. Sc. Statist. Comput.*, 11(6):1102–1115, 1990. (Cited on p. 586.)

- [224] Walter Gander and Gene H. Golub. Cyclic reduction—history and applications. In Franklin T. Luk and Robert James Plemmons, editors, *Scientific Computing*, pages 73–85. Springer Verlag, Singapore, 1997. (Cited on p. 186.)
- [225] Walter Gander, Gene H. Golub, and Rolf Strebel. Least squares fitting of circles and ellipses. *BIT*, 34(4):558–578, 1994. (Cited on p. 432.)
- [226] Walter Gander and Jiří Hřebíček. *Solving Problems in Scientific Computing using Maple and Matlab*. Springer-Verlag, Berlin, fourth edition, 2004. (Cited on p. 330.)
- [227] F. R. Gantmacher. *The Theory of Matrices. Vol. I.* Chelsea Publishing Co, New York, 1959. (Cited on pp. 184, 573.)
- [228] F. R. Gantmacher. *The Theory of Matrices. Vol. II.* Chelsea Publishing Co, New York, 1959. (Cited on pp. 184, 589.)
- [229] B. S. Garbow, J. M. Boyle, Jack J. Dongarra, and G. W. Stewart. *Matrix Eigen-systems Routines: EISPACK Guide Extension*. Springer-Verlag, New York, 1977. (Cited on pp. 133, 587.)
- [230] M. Gasca and J. M. Peña. On factorization of totally positive matrices. In M. Gasca and C. A. Micchelli, editors, *Total Positivity*, pages 109–130. Kluwer Academic Publ., Dordrecht, 1996. (Cited on p. 181.)
- [231] N. Gastinel. *Linear Numerical Analysis*. Hermann, Paris, 1970. (Cited on p. 185.)
- [232] C. F. Gauss. *The Theory of the Combination of Observations Least Subject to Errors. Pars Prior*. SIAM, Philadelphia, PA, G. W. Stewart, Translation 1995, 1821. (Cited on pp. 193, 349.)
- [233] Carl Friedrich Gauss. *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections*. Dover, New York, (1963), C. H. Davis, Translation, 1809. (Cited on p. 348.)
- [234] Walter Gautschi. *Numerical Analysis. An Introduction*. Birkhäuser, Boston, MA, 1997. (Cited on p. 107.)
- [235] Alan George. Nested dissection of a regular finite element grid. *SIAM J. Numer. Anal.*, 10:345–363, 1973. (Cited on p. 166.)
- [236] Alan George and Michael T. Heath. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra Appl.*, 34:69–83, 1980. (Cited on pp. 306, 351.)
- [237] Alan George, Kkakim D. Ikramov, and Andrey B. Kucherov. On the growth factor in Gaussian elimination for generalized Higham matrices. *Numer. Linear Algebra Appl.*, 9:107–114, 2002. (Cited on p. 79.)
- [238] Alan George and Joseph W. H. Liu. User guide for SPARSPAK, Waterloo Sparse Linear Equation Package. Tech. Report Res. CS 78–30, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1980. (Cited on p. 186.)
- [239] Alan George and Joseph W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981. (Cited on pp. 154, 162, 162, 163, 167, 187.)
- [240] Alan George and Joseph W. H. Liu. Householder reflections versus Givens rotations in sparse orthogonal decomposition. *Linear Algebra Appl.*, 88/89:223–238, 1987. (Cited on pp. 307, 351.)
- [241] Alan George and Joseph W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1–19, 1989. (Cited on pp. 163, 186.)

- [242] Alan George and Esmond Ng. On row and column orderings for sparse least squares problems. *SIAM J. Numer. Anal.*, 20:326–344, 1983. (Cited on p. 305.)
- [243] J. A. George and E. G. Ng. An implementation of Gaussian elimination with partial pivoting for sparse systems. *SIAM J. Sci. Statist. Comput.*, 6(2):390–409, 1985. (Cited on p. 169.)
- [244] J. A. George and E. G. Ng. Symbolic factorization for sparse Gaussian elimination with partial pivoting. *SIAM J. Sci. Statist. Comput.*, 8(6):877–898, 1987. (Cited on pp. 169, 303.)
- [245] S. A. Gershgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Akademia Nauk SSSR, Mathematics and Natural Sciences*, 6:749–754, 1931. (Cited on p. 456.)
- [246] John R. Gilbert. Predicting structure in sparse matrix computations. *SIAM J. Matrix Anal. Appl.*, 15(1):62–79, 1994. (Cited on p. 186.)
- [247] John R. Gilbert, Cleve Moler, and Robert Schreiber. Sparse matrices in Matlab: Design and implementation. *SIAM J. Matrix Anal. Appl.*, 13(1):333–356, 1992. (Cited on pp. 151, 170, 171.)
- [248] John R. Gilbert, Esmond Ng, and B. W. Peyton. Separators and structure prediction in sparse orthogonal factorization. *Linear Algebra Appl.*, 262:83–97, 1997. (Cited on p. 306.)
- [249] John R. Gilbert and Tim Peierls. Sparse partial pivoting in time proportional to arithmetic operations. *SIAM J. Sc. Statist. Comput.*, 9(5):862–874, 1988. (Cited on p. 169.)
- [250] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright. User’s guide for LSSOL (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. (Cited on p. 432.)
- [251] Philip E. Gill, Gene H. Golub, Walter Murray, and Michael Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28:505–535, 1974. (Cited on p. 382.)
- [252] Philip E. Gill and Walter Murray. Algorithms for the solution of the nonlinear least squares problem. *SIAM J. Numer. Anal.*, 15:977–992, 1978. (Cited on p. 390.)
- [253] Philip E. Gill, Walter Murray, D. B. Ponceleón, and Michael A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix. Anal. Appl.*, 13:292–311, 1992. (Cited on p. 676.)
- [254] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, London, UK, 1981. (Cited on pp. 385, 427, 429, 432.)
- [255] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Numerical Linear Algebra and Optimization. Volume 1*. Addison-Wesley, London, UK, 1991. (Cited on pp. 408, 415, 416.)
- [256] L. Giraud, J. Langou, and M. Rozložník. On the round-off error of the gram–schmidt algorithm with reorthogonalization. Tech. Report TR/PA/02/33, CERFACS, Toulouse, France, 2002. (Cited on p. 699.)
- [257] Wallace G. Givens. Computation of plane unitary rotations transforming a general matrix to triangular form. *SIAM J. Appl. Math.*, 6(1):26–50, 1958. (Cited on pp. 236, 350.)
- [258] I. Gohberg, Thomas Kailath, and I. Kohlstrach. Linear complexity algorithms for semiseparable matrices. *J. Integral. Equations Operator Theory.*, 8:779–804, 1985. (Cited on p. 182.)

- [259] I. Gohberg, Thomas Kailath, and Vadim Olshevsky. Fast Gaussian elimination with partial pivoting for matrices with displacement structure. *Math. Comp.*, 64:1557–1576, 1995. (Cited on pp. 179, 180.)
- [260] G. H. Golub and R. J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra Appl.*, 34:3–28, 1980. (Cited on pp. 219, 351.)
- [261] Gene H. Golub. Numerical methods for solving least squares problems. *Numer. Math.*, 7:206–216, 1965. (Cited on p. 350.)
- [262] Gene H. Golub. Least squares, singular values and matrix approximations. *Aplikace Matematiky*, 13:44–51, 1968. (Cited on p. 588.)
- [263] Gene H. Golub. Matrix computations and the theory of moments. In S. D. Chatterji, editor, *Proceedings of the International Congress of Mathematicians, Zürich 1994*, volume 2, pages 1440–1448, Basel, Switzerland, 1995. Birkhäuser-Verlag. (Cited on p. 578.)
- [264] Gene H. Golub and Carl D. Meyer 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, 1975. (Cited on pp. 579, 581.)
- [265] Gene H. Golub and W. Kahan. Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal. Ser. B*, 2:205–224, 1965. (Cited on pp. 351, 588, 651.)
- [266] Gene H. Golub, Virginia Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Tech. Report STAN-CS-76-559, August 1976, Computer Science Department, Stanford University, CA, 1976. (Cited on p. 272.)
- [267] Gene H. Golub and Randy J. LeVeque. 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.*, pages 1–12. White Sands Missile Range, White Sands, NM, ARO Report 79-3, 1979. (Cited on p. 432.)
- [268] Gene H. Golub and Gérard Meurant. Matrices, moments and quadrature. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1993: Proceedings of the 13th Dundee Biennial Conference*, Pitman Research Notes in mathematics, pages 105–156. Longman Scientific and Technical, Harlow, Essex, UK, 1994. (Cited on p. 578.)
- [269] Gene H. Golub, Stephen Nash, and Charles F. Van Loan. A Hessenberg–Schur method for the the matrix problem $AX + XB = C$. *IEEE Trans. Automat. Control.*, AC-24:909–913, 1972. (Cited on p. 450.)
- [270] Gene H. Golub and Dianne P. O’Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Review*, 31:50–102, 1989. (Cited on p. 718.)
- [271] Gene H. Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.*, 10(2):413–432, 1973. (Cited on p. 392.)
- [272] Gene H. Golub and Victor Pereyra. Separable nonlinear least squares: the variable projection method and its application. *Inverse Problems*, 19:R1–R26, 2003. (Cited on p. 431.)
- [273] Gene H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numer. Math.*, 14:403–420, 1970. (Cited on pp. 515, 588.)

- [274] Gene H. Golub and Henk A. van der Vorst. Closer to the solution: iterative linear solvers. In I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 63–92. Clarendon Press, Oxford, 1997. (Cited on p. 669.)
- [275] Gene H. Golub and Henk A. van der Vorst. Eigebvalue computations in the 20th century. *J. Comput. Appl. Math.*, 123:35–65, 2000. (Cited on p. 587.)
- [276] Gene H. Golub and Charles F. Van Loan. An analysis of the total least squares problem. *SIAM J. Numer. Anal.*, 17(6):883–893, 1980. (Cited on pp. 335, 351.)
- [277] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996. (Cited on pp. 30, 184, 208, 239, 272, 349, 489, 523, 544.)
- [278] Gene H. Golub and James H. Wilkinson. Ill-conditioned eigensystems and the computation of the Jordan canonical form. *SIAM Review.*, 18(4):578–619, 1976. (Cited on p. 587.)
- [279] R. Gordon, R. Bender, and Gabor T. Herman. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *J. Theor. Biology*, 29:471–481, 1970. (Cited on p. 718.)
- [280] W. B. Gragg. The QR algorithm for unitary Hessenberg matrices. *J. Comp. Appl. Math.*, 16:1–8, 1986. (Cited on p. 523.)
- [281] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, PA, 1997. (Cited on p. 717.)
- [282] Andreas Griewank and Andrea Walther. *Automatic Differentiation of Algorithm. From Simulation to Optimization*. Springer-Verlag, Berlin, second edition, 2008. (Cited on p. 431.)
- [283] Roger G. Grimes, John G. Lewis, and H. D. Simon. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM J. Matrix Anal. Appl.*, 15(1):228–272, 1994. (Cited on p. 720.)
- [284] Igor Griva, Stephen G. Nash, and Ariela Sofer. *Linear and Nonlinear Optimization*. SIAM, Philadelphia, PA, second edition, 2008. (Cited on p. 432.)
- [285] Benedikt Grosser and Bruno Lang. An $O(n^2)$ algorithm for the bidiagonal SVD. *Linear Algebra Appl.*, 358:45–70, 2003. (Cited on p. 589.)
- [286] M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18(3):838–853, 1997. (Cited on p. 685.)
- [287] Ming Gu. Backward perturbation bounds for linear least squares problems. *SIAM J. Matrix Anal. Appl.*, 20(2):363–372, 1998. (Cited on p. 349.)
- [288] Ming Gu. Stable and efficient algorithms for structured systems of linear equations. *SIAM J. Matrix Anal. Appl.*, 19(2):279–306, 1998. (Cited on p. 180.)
- [289] Ming Gu, James W. Demmel, and Inderjit Dhillon. Efficient computation of the singular value decomposition with applications to least squares problems. Tech. Report TR/PA/02/33, Department of Mathematics and Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720, 1994. (Cited on p. 529.)
- [290] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal svd. *SIAM J. Matrix. Anal. Appl.*, 16(1):79–92, 1995. (Cited on p. 531.)
- [291] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix. Anal. Appl.*, 16(1):172–191, 1995. (Cited on pp. 526, 529.)

- [292] Ming Gu and Stanley C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996. (Cited on p. 273.)
- [293] M. E. Gulliksson, Inge Söderkvist, and Per-Åke Wedin. Algorithms for constrained and weighted nonlinear least squares. *SIAM J. Optim.*, 7:208–224, 1997. (Cited on p. 431.)
- [294] M. E. Gulliksson and Per-Åke Wedin. Perturbation theory for generalized and constrained linear least squares. *Numer. Linear Algebra Appl.*, 7:181–196, 2000. (Cited on p. 349.)
- [295] Chun-Hua Guo, Nicholas J. Higham, and Françoise Tisseur. An improved arc algorithm for detecting definite Hermitian pairs. *SIAM J. Matrix Anal. Appl.*, to appear, 2009. (Cited on p. 549.)
- [296] Ivar Gustafsson. A class of first order factorization methods. *BIT*, 18(2):142–156, 1978. (Cited on p. 682.)
- [297] Fred G. Gustavson. Recursion leads to automatic variable blocking for dense linear-algebra algorithms. *IBM J. Res. Develop.*, 41(6):737–754, 1997. (Cited on pp. 142, 146.)
- [298] Martin H. Gutknecht. A complemeted theory of the unsymmetric Lanczos process and related algorithms. *SIAM J. Matrix Anal. Appl.*, 13:594–639, 1992. (Cited on p. 719.)
- [299] Martin H. Gutknecht. Block Krylov space methods for linear systems with multiple right hand-sides: An introduction. In A. H. Siddiqi, I. S. Duff, and O. Christensen, editors, *Modern Mathematical Models, Methods, and Algorithms for Real World Systems*, pages 420–447. Anarnaya Publishers, New Dehli, India, 2007. (Cited on p. 718.)
- [300] William W. Hager. Condition estimates. *SIAM J. Sci. Stat. Comput.*, 5(2):311–316, 1984. (Cited on pp. 115, 116.)
- [301] William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989. (Cited on p. 13.)
- [302] Nicholas Hale, Nicholas J. Higham, and Lloyd N. Trefethen. Computing A^α , $\log(A)$ and related matrix functions by contour integrals. *SIAM J. Matrix Anal. Appl.*, to appear, 2008. (Cited on p. 589.)
- [303] S. J. Hammarling. Numerical solution of the stable non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982. (Cited on p. 450.)
- [304] S. J. Hammarling and J. H. Wilkinson. The practical behaviour of linear iterative methods with particular reference to S.O.R. Tech. Report NAC 69, National Physical Laboratory, Teddington, UK, 1976. (Cited on p. 717.)
- [305] M. Hanke. *Conjugate Gradient Type Methods for Ill-posed Problems*. Pitman Research Notes in Mathematics. Longman Scientific and Technical, Harlow, UK, 1995. (Cited on pp. 648, 650, 718.)
- [306] M. Hanke. Regularization methods for large-scale problems. *BIT*, 41:1008–1018, 2001. (Cited on p. 287.)
- [307] M. Hanke and P. C. Hansen. Regularization methods for large-scale problems. *Surveys Math. Indust.*, 3:253–315, 1993. (Cited on p. 718.)

- [308] Eldon Hansen. Interval arithmetic in matrix computations. ii. *SIAM J. Numer. Anal.*, 4(1):1–9, 1965. (Cited on p. 131.)
- [309] Eldon Hansen. *Topics in Interval Analysis*. Oxford University Press, Oxford, 1969. (Cited on p. 130.)
- [310] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion*. SIAM, Philadelphia, 1998. (Cited on pp. 348, 351, 718.)
- [311] Richard J. Hanson and Michael J. Norris. Analysis of measurements based on the singular value decomposition. *SIAM J. Sci. Stat. Comput.*, 2(3):363–373, 1981. (Cited on p. 333.)
- [312] G. I. Hargreaves. Interval analysis in MATLAB. Numer. anal. report 418, Department of Mathematics, University of Manchester, 2002. (Cited on p. 130.)
- [313] Thomas Hawkins. The theory of matrices in the 19th century. In R. D. James, editor, *Proceedings of the International Congress of Mathematics, Vancouver 1974*, volume 2, pages 561–570, 1975. (Cited on p. 185.)
- [314] M. D. Hebdon. An algorithm for minimization using exact second derivatives. Tech. Report T. P. 515, Atomic Energy Research Establishment, Harwell, England, 1973. (Cited on p. 328.)
- [315] G. Heinig. Inversion of generalized Cauchy matrices and other classes of structured matrices. *IMA Vol. Math. Appl.*, 69:95–114, 1994. (Cited on p. 178.)
- [316] F. R. Helmert. *Die Mathematischen und Physikalischen Theorien der höheren Geodäsie, 1 Teil*. B. G. Teubner Verlag, Leipzig, 1880. (Cited on p. 300.)
- [317] H. V. Henderson and S. R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23(1):53–60, 1981. (Cited on p. 12.)
- [318] H. V. Henderson and S. R. Searle. The vec-permutation matrix, the vec operator and Kronecker products; a review. *Linear Multilinear Algebra*, 9:271–288, 1981. (Cited on p. 186.)
- [319] Gabor T. Herman. *Image Reconstruction from Projections. The Fundamentals of Computerized Tomography*. Academic Press, New York, 1980. (Cited on p. 718.)
- [320] Karl Hessenberg. Behandlung linearer Eigenvertaufgaben mit Hilfe der Hamilton–Cayleyschen Gleichung. Technical Report IPM-1, Institute of Practical Mathematics, Technische Hochschule, Darmstadt, 1940. (Cited on p. 87.)
- [321] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear system. *J. Res. Nat. Bur. Standards, Sect. B*, 49:409–436, 1952. (Cited on pp. 629, 718.)
- [322] K. L. Hiebert. An evaluation of mathematical software that solves systems of nonlinear equations. *ACM Trans. Math. Software*, 8:5–20, 1982. (Cited on p. 431.)
- [323] Nicholas J. Higham. Computing the polar decomposition—with applications. *SIAM J. Sci. Stat. Comput.*, 7(4):1160–1174, 1986. (Cited on pp. 206, 576, 585.)
- [324] Nicholas J. Higham. Error analysis of the Björck–Pereyra algorithm for solving Vandermonde systems. *Numer. Math.*, 50:613–632, 1987. (Cited on p. 180.)
- [325] Nicholas J. Higham. FORTRAN codes for estimating the one-norm of a real or complex matrix, with application to condition estimation. *ACM Trans. Math. Software*, 14(4):381–396, 1988. (Cited on p. 115.)

- [326] Nicholas J. Higham. Stability of the diagonal pivoting method with partial pivoting. *SIAM J. Matrix Anal. Appl.*, 18(1):52–65, 1997. (Cited on p. 186.)
- [327] Nicholas J. Higham. Evaluating Padé approximants of the matrix logarithm. *SIAM J. Matrix Anal. Appl.*, 22:4:1126–1135, 2001. (Cited on p. 570.)
- [328] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, second edition, 2002. (Cited on pp. 17, 33, 36, 67, 78, 82, 83, 117, 132, 142, 181, 184, 185, 185, 249, 262.)
- [329] Nicholas J. Higham. The matrix computation toolbox for MATLAB (Version 1.0). Numerical Analysis Report 410, Department of Mathematics, The University of Manchester, 2002. (Cited on p. 558.)
- [330] Nicholas J. Higham. *J*-Orthogonal matrices: Properties and generation. *SIAM Review*, 45(3):504–519, 2003. (Cited on pp. 319, 351.)
- [331] Nicholas J. Higham. The scaling and squaring method for the matrix exponential function. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193, 2005. (Cited on p. 569.)
- [332] Nicholas J. Higham. *Functions of Matrices. Theory and Computation*. SIAM, Philadelphia, PA, 2008. (Cited on p. 589.)
- [333] Nicholas J. Higham and Philip A. Knight. Componentwise error analysis for stationary iterative methods. In Carl D. Meyer and Robert J. Plemmons, editors, *Linear Algebra, Markov Chains, and Queueing Models*, volume 48, pages 29–46, Berlin, Germany, 1993. Springer-Verlag. (Cited on p. 718.)
- [334] Nicholas J. Higham and Philip A. Knight. Finite precision behavior of stationary iteration for solving singular systems. *Linear Algebra Appl.*, 192:165–186, 1993. (Cited on p. 718.)
- [335] Michiel E. Hochstenbach. Harmonic and refined extraction methods for the singular value problem, with applications in least squares problems. *BIT*, 44(4):721–754, 2004. (Cited on p. 720.)
- [336] A. J. Hoffman and H. W. Wielandt. The variation of the spectrum of a normal matrix. *Duke Math. J.*, 20:37–39, 1953. (Cited on p. 465.)
- [337] Y. T. Hong and C. T. Pan. Rank-revealing QR decompositions and the singular value decomposition. *Math. Comp.*, 58:213–232, 1992. (Cited on pp. 272, 352.)
- [338] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985. ISBN 0-521-30586-1. (Cited on pp. 20, 175, 184.)
- [339] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1991. ISBN 0-521-30587-X. (Cited on pp. 184, 558.)
- [340] Harold Hotelling. Some new methods in matrix calculus. *Ann. Math. Statist.*, 14:1–34, 1943. (Cited on p. 185.)
- [341] A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.*, 5:339–342, 1958. (Cited on pp. 233, 350.)
- [342] A. S. Householder and F. L. Bauer. On certain iterative methods for solving linear systems. *Numer. Math.*, 2:55–59, 1960. (Cited on p. 646.)
- [343] Alston S. Householder. *The Theory of Matrices in Numerical Analysis*. Dover, New York, 1975. Corrected republication of work first published in 1964 by Blaisdell Publ. Co., New York. (Cited on pp. 22, 184, 589.)

- [344] P. J. Huber. *Robust Statistics*. John Wiley, New York, 1981. (Cited on p. 351.)
- [345] Bruno Iannazzo. A note on computing the matrix square root. *Calcolo*, 40:273–283, 2003. (Cited on p. 573.)
- [346] Bruno Iannazzo. A the Newton method for the matrix p th root. *SIAM J. Matrix Anal. Appl.*, 28(2):503–523, 2006. (Cited on p. 573.)
- [347] Yasuhiko Ikebe. On inverses of Hessenberg matrices. *Linear Algebra Appl.*, 24:93–97, 1979. (Cited on p. 98.)
- [348] Ilse Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 39:254–291, 1997. (Cited on p. 588.)
- [349] B. M. Irons. A frontal solution program for finite element analysis. *Internat. J. Numer. Methods Engrg.*, 2:5–32, 1970. (Cited on p. 167.)
- [350] C. G. J. Jacobi. Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden linearen Gleichungen. *Astron. Nachrichten*, 22:297–306, 1845. (Cited on p. 350.)
- [351] C. G. J. Jacobi. Über ein leichtes Verfahren die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen. *Crelle J. reine angew. Math.*, 30:51–94, 1846. (Cited on p. 535.)
- [352] A. Jennings and M. A. Ajiz. Incomplete methods for solving $A^T Ax = A^T b$. *SIAM J. Matrix Anal. Appl.*, 5:978–987, 1984. (Cited on p. 684.)
- [353] E. R. Jessup and D. C. Sorenson. A parallel algorithm for computing the singular value decomposition of a matrix. *SIAM J. Matrix Anal. Appl.*, 15(2):530–548, 1994. (Cited on p. 529.)
- [354] Z. Jia and D. Niu. An implicitly restarted refined bidiagonalization Lanczos method for computing a partial singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 25(1):246–265, 2004. (Cited on p. 720.)
- [355] L. W. Johnson and D. R. Scholz. On Steffensen's method. *SIAM J. Numer. Anal.*, 5(2):296–302, 1968. (Cited on p. 369.)
- [356] Camille Jordan. Mémoires sur les formes bilinéaires. *J. Meth. Pures. Appl., Deuxième Série.*, 19:35–54, 1874. (Cited on p. 349.)
- [357] S. Kaczmarz. Angenäherte auflösung von systemen linearen gleichungen. *Acad. Polon. Sciences et Lettres*, pages 355–357, 1937. (Cited on p. 647.)
- [358] Bo Kågström, Per Ling, and Charles F. Van Loan. GEMM-based level 3 BLAS high performance model implementation and performance evaluation benchmarks. *ACM Trans. Math. Software*, 24(3):268–302, 1998. (Cited on p. 135.)
- [359] Bo Kågström and Axel Ruhe. Algorithm 560 JNF: An algorithms for numerical computaton of the Jordan normal form of a complex matrix. *ACM Trans. Math. Software*, 16(3):437–443, 1980. (Cited on p. 587.)
- [360] Bo Kågström and Axel Ruhe. An algorithm for numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Software*, 16(3):398–419, 1980. (Cited on p. 587.)
- [361] W. Kahan. Inclusion theorems for clusters of eigenvalues of Hermitian matrices. Tech. Report CS42, Computer Science Department, University of Toronto, 1967. (Cited on p. 698.)

- [362] W. M. Kahan. Accurate eigenvalues of a symmetric tri-diagonal matrix. Tech. Report No. CS-41, Revised June 1968, Computer Science Department, Stanford University, CA, 1966. (Cited on p. 533.)
- [363] W. M. Kahan. Numerical linear algebra. *Canad. Math. Bull.*, 9:757–801, 1966. (Cited on pp. 57, 107.)
- [364] Thomas Kailath and Ali H. Sayed. Displacement structures: Theory and applications. *SIAM Review*, 37:297–386, 1995. (Cited on p. 187.)
- [365] S. Kaniel. Estimates for some computational techniques in linear algebra. *Math. Comp.*, 20:369–378, 1966. (Cited on p. 719.)
- [366] L. V. Kantorovich. On Newton’s method. *Trudy Mat. Inst. Steklov*, 28:104–144, 1949. in Russian. (Cited on pp. 364, 431.)
- [367] L. V. Kantorovich and G. P. Akilov. *Functional Analysis in Normed Spaces*. Macmillan, New York, 1964. (Cited on p. 431.)
- [368] Rune Karlsson and Bertil Waldén. Estimation of optimal backward perturbation bounds for the linear least squares problem. *BIT*, 37:4:862–869, 1997. (Cited on p. 349.)
- [369] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984. (Cited on p. 432.)
- [370] W. Karush. An iteration method for finding characteristic vectors of a symmetric matrix. *Pacific J. Math.*, 1:233–248, 1951. (Cited on p. 719.)
- [371] T. Kato. *Perturbation Theory for Linear Operators*. Springer-Verlag, New York, second edition, 1976. (Cited on p. 587.)
- [372] L. Kaufman. Variable projection methods for solving separable nonlinear least squares problems. *BIT*, 15:49–57, 1975. (Cited on p. 393.)
- [373] L. Kaufman and G. Sylvester. Separable nonlinear least squares problems with multiple right hand sides. *SIAM J. Matrix Anal. Appl.*, 13:68–89, 1992. (Cited on p. 432.)
- [374] Charles S. Kenney and Alan J. Laub. Padé error estimates for the logarithm of a matrix. *Internat J. Control.*, 10:707–730, 1989. (Cited on p. 569.)
- [375] Charles S. Kenney and Alan J. Laub. Rational iterative methods for the matrix sign function. *SIAM J. Matrix Anal. Approx.*, 12(2):273–291, 1991. (Cited on p. 586.)
- [376] Charles S. Kenney and Alan J. Laub. On scaling Newton’s method for polar decomposition and the matrix sign function. *SIAM J. Matrix Anal. Approx.*, 13(3):688–706, 1992. (Cited on p. 576.)
- [377] Philip A. Knight and Daniel Ruiz. A fast method for matrix balancing. In Andreas Frommer, Michael W. Mahoney, and Daniel B. Szyld, editors, *Web Information Retrieval and Linear Algebra Algorithms*, Schloss Dagstuhl, Germany, 2007. Internationales Begegnungs- aufn Forschungszentrum für Informatik. (Cited on p. 504.)
- [378] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Reading, MA, third edition, 1998. (Cited on p. 143.)
- [379] E. G. Kogbetlantz. Solution of linear equations by diagonalization of coefficients matrix. *Quart. Appl. Math.*, 13:123–132, 1955. (Cited on p. 540.)
- [380] Tamara G. Kolda. Multilinear operators for higher-order decompositions. Technical Report SAND 2006–2081, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, April 2006. (Cited on p. 313.)

-
- [381] Tamara G. Kolda and R. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:to appear, 2009. (Cited on p. 313.)
 - [382] Tamara G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482, 2003. (Cited on p. 432.)
 - [383] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditioning I. Theory. *SIAM J. Matrix. Anal. Appl.*, 14(1):45–58, 1998. (Cited on p. 685.)
 - [384] E. Koukiopoulou, C. Bekas, and E. Gallopoulos. Computing smallest singular value triplets with implicitly restarted Lanczos bidiagonalization. *Appl. Numer. Math.*, 49(1):39–61, 2004. (Cited on p. 720.)
 - [385] S. Kourouklis and Christopher C.Paige. A constrained approach to the general Gauss-Markov linear model. *J. Amer. Statist. Assoc.*, 76:620–625, 1981. (Cited on p. 197.)
 - [386] H. O. Kreiss. Über die Stabilitätsdefinition für Differenzengleichungen die Partielle Differentialgleichungen approximieren. *BIT*, 2:153–181, 1962. (Cited on p. 561.)
 - [387] Vera N. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *USSR Comput. Math. Phys.*, 3:637–657, 1961. (Cited on p. 492.)
 - [388] H. P. Künzi, H. G. Tzschach, and C. A. Zehnder. *Numerical Methods of Mathematical Optimization*. Academic Press, New York, 1971. (Cited on p. 432.)
 - [389] P. Lancaster and L. Rodman. *The Algebraic Riccati Equation*. Oxford University Press, Oxford, 1995. (Cited on p. 587.)
 - [390] Peter Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, New York, 1985. (Cited on pp. 184, 589.)
 - [391] Cornelius Lanczos. 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. (Cited on pp. 634, 665.)
 - [392] Cornelius Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards, Sect. B*, 49:33–53, 1952. (Cited on p. 719.)
 - [393] L. Landweber. An iterative formula for Fredholm integral equations of the first kind. *Amer. J. Math.*, 73:615–624, 1951. (Cited on p. 643.)
 - [394] P. S. Laplace. *Théorie analytique des probabilités. Premier supplément*. Courcier, Paris, 3rd edition, 1820. with an introduction and three supplements. (Cited on p. 218.)
 - [395] Rasmus Munk Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical Report, Department of Computer Science, University of Aarhus, DK-8000 Aarhus, Denmark, 1998. (Cited on p. 720.)
 - [396] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000. (Cited on p. 313.)
 - [397] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank (R_1, R_2, \dots, R_N) approximation of hogher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000. (Cited on p. 313.)
 - [398] P. Läuchli. Jordan-Elimination und Ausgleichung nach kleinsten Quadraten. *Numer. Math.*, 3:226–240, 1961. (Cited on p. 223.)

- [399] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1974. Reprinted by SIAM, Philadelphia, PA, 1995. (Cited on pp. 246, 280, 291, 291, 348, 426, 431.)
- [400] Charles L. Lawson, Richard J. Hanson, D. R. Kincaid, and Fred T. Krogh. Basic Linear Algebra Subprograms for Fortran usage. *ACM Trans. Math. Software*, 5:308–323, 1979. (Cited on p. 133.)
- [401] Adrien-Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. Courcier, Paris, 1805. (Cited on p. 348.)
- [402] R. B. Lehoucq. The computations of elementary unitary matrices. *ACM Trans. Math. Software*, 22:393–400, 1996. (Cited on p. 350.)
- [403] R. B. Lehoucq. Implicitly restarted Arnoldi methods and subspace iterations. *SIAM J. Matrix Anal. Appl.*, 23:551–562, 2001. (Cited on p. 719.)
- [404] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, PA, 1998. (Cited on p. 719.)
- [405] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944. (Cited on p. 388.)
- [406] N. Levinson. The Wiener rms root-mean square error criterion in filter design and prediction. *J. Math. Phys.*, 25:261–278, 1947. (Cited on p. 175.)
- [407] Ren-Cang Li. Solving secular equations stably and efficiently. Technical Report UCB/CSD-94-851, Computer Science Department, University of California, Berkeley CA 94720, 1994. (Cited on p. 528.)
- [408] Ren-Cang Li. Relative perturbation theory: I. Eigenvalue and singular value variations. *SIAM J. Matrix Anal. Appl.*, 19(4):956–982, 1998. (Cited on p. 587.)
- [409] Ren-Cang Li. Relative perturbation theory: II. Eigenspace and singular subspace variations. *SIAM J. Matrix Anal. Appl.*, 20(2):471–492, 1998. (Cited on p. 587.)
- [410] X. S. Li, James W. Demmel, David H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, A. Kapur, M. C. Martin, T. Tung, and D. J. Yoo. Design, implementation and testing of extended and mixed precision BLAS. LAPACK working note 149 Tech. Report CS-00-451, Department of Computer Science, University of Tennessee, Knoxville, TN, 2000. (Cited on p. 126.)
- [411] R. J. Lipton, D. J. Rode, and R. E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16:346–358, 1979. (Cited on p. 167.)
- [412] Joseph W. H. Liu. On general row merging schemes for sparse Givens transformations. *SIAM J. Sci. Stat. Comput.*, 7:1190–1211, 1986. (Cited on pp. 307, 308, 351.)
- [413] Joseph W. H. Liu. The role of elimination trees in sparse matrix factorization. *SIAM J. Matrix Anal. Appl.*, 11(1):134–172, 1990. (Cited on pp. 159, 160, 160, 186, 310, 351.)
- [414] Joseph W. H. Liu. The multifrontal method for sparse matrix solution: theory and practice. *SIAM Review.*, 34(1):82–109, 1992. (Cited on p. 186.)
- [415] P. Lötstedt. Perturbation bounds for the linear least squares problem subject to linear inequality constraints. *BIT*, 23:500–519, 1983. (Cited on p. 423.)
- [416] P. Lötstedt. Solving the minimal least squares problem subject to bounds on the variables. *BIT*, 24:206–224, 1984. (Cited on p. 424.)

- [417] S. M. Lozinskii. Error estimate for the numerical integration of ordinary differential equations. *Izv. Vyssh. Učebn. Zaved. Matematika*, 6:52–90, 1958. in Russian. (Cited on p. 589.)
- [418] S.-M. Lu and J. L. Barlow. Multifrontal computation with the orthogonal factors of sparse matrices. *SIAM J. Matrix Anal. Appl.*, 17:658–679, 1996. (Cited on p. 311.)
- [419] D. G. Luenberger. *Introduction to Dynamic Systems. Theory, Models and Applications*. John Wiley, New York, 1979. (Cited on p. 432.)
- [420] D. Steven Mackey, Niloufer Mackey, and Françoise Tisseur. Structured tools for structured matrices. *Electron. J. Linear Algebra*, 332–334:106–145, 2003. (Cited on p. 555.)
- [421] Kaj Madsen and Hans Bruun Nielsen. Finite algorithms for robust linear regression. *BIT*, 30(4):682–699, 1990. (Cited on p. 352.)
- [422] Kaj Madsen and Hans Bruun Nielsen. A finite smoothing algorithm for linear ℓ_1 estimation. *SIAM J. Opt.*, 3(2):223–235, 1993. (Cited on p. 346.)
- [423] Marvin Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities*. Allyn and Bacon, Boston, MA, 1964. Reprinted by Dover, Mineola, NY, 1992. (Cited on pp. 16, 184.)
- [424] A. A. Markov. *Wahrscheinlichkeitsrechnung*. Liebmann, Leipzig, second edition, 1912. (Cited on p. 349.)
- [425] H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Sci.*, 3:255–269, 1957. (Cited on pp. 168, 186.)
- [426] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11:431–441, 1963. (Cited on p. 388.)
- [427] R. S. Martin and James H. Wilkinson. Reduction of the symmetric eigenproblem $Ax = \lambda Bx$ and related problems to standard form. *Numer. Math.*, 11:99–110, 1968. Also in [614, pp. 303–314]. (Cited on p. 550.)
- [428] Roy Mathias and G. W. Stewart. A block QR algorithm and the singular value decompositions. *Linear Algebra Appl.*, 182:91–100, 1993. (Cited on p. 271.)
- [429] Pontus Matstoms. Sparse QR factorization in MATLAB. *ACM Trans. Math. Software*, 20:136–159, 1994. (Cited on p. 351.)
- [430] Pontus Matstoms. *Sparse QR Factorization with Applications to Linear Least Squares Problems*. Ph.D. thesis, Department of Mathematics, Linköping University, Sweden, 1994. (Cited on p. 351.)
- [431] J. A. Meijerink and Henk A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31:148–162, 1977. (Cited on p. 683.)
- [432] Jean Meinguet. Refined error analysis of Cholesky factorization. *SIAM J. Numer. Anal.*, 20(6):1243–1250, 1983. (Cited on p. 122.)
- [433] Gérard Meurant. *Computer Solution of Large Linear Systems*. North Holland Elsevier, Amsterdam, 1999. (Cited on p. 717.)
- [434] Gérard Meurant. *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations*. SIAM, Philadelphia, PA, 2006. (Cited on p. 718.)
- [435] Gérard Meurant and Zdeněk Strakoš. The Lanczos and conjugate gradient algorithms. in finite precision arithmetic. *Acta Numerica*, 15:471–542, 2006. (Cited on p. 718.)

- [436] Carl D. Meyer. The role of the group generalized inverse in the theory of finite Markov chains. *SIAM Review*, 17:443–464, 1975. (Cited on p. 581.)
- [437] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, 2000. (Cited on p. 579.)
- [438] Kenneth S. Miller. Complex least squares. *SIAM Review*, 15(4):706–726, 1973. (Cited on p. 195.)
- [439] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford*, 11:50–59, 1960. (Cited on p. 206.)
- [440] Cleve Moler. The world’s largest matrix computation. *MATLAB News and Notes*, pages 12–13, October 2002. (Cited on p. 477.)
- [441] Cleve Moler and Charles F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45:3–49, 2003. (Cited on pp. 566, 569.)
- [442] Cleve Moler and G. W. Stewart. An algorithm for generalized eigenvalue problems. *SIAM J. Numer. Anal.*, 10:241–256, 1973. (Cited on p. 551.)
- [443] E. H. Moore. On the reciprocal of the general algebraic matrix. *Bull. Amer. Math. Soc.*, 26:394–395, 1920. (Cited on p. 200.)
- [444] J. J. Moré and M. Y. Cosnard. Numerical solution of nonlinear equations. *ACM Trans. Math. Software*, 5:64–85, 1979. (Cited on p. 431.)
- [445] J. J. Moré and Stephen J. Wright. *Optimization Software Guide*. SIAM, Philadelphia, PA, 1993. (Cited on p. 432.)
- [446] N. M. Nachtigal, S. C. Reddy, and Lloyd N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM J. Matrix Anal. Appl.*, 13:778–795, 1992. (Cited on p. 669.)
- [447] Larry Neal and George Poole. A geometric analysis of Gaussian elimination. ii. *Linear Algebra Appl.*, 173:239–264, 1992. (Cited on p. 185.)
- [448] Larry Neal and George Poole. The rook’s pivoting strategy. *J. Comput. Appl. Math.*, 123:353–369, 2000. (Cited on p. 185.)
- [449] John von Neumann. Some matrix inequalities in the space of matrices. *Tomsk Univ. Rev.*, 1:286–299, 1937. in Russian. (Cited on p. 27.)
- [450] John von Neumann and Herman H. Goldstein. Numerical inverting of matrices of high order. *Bull. Amer. Math. Soc.*, 53:1021–1099, 1947. (Cited on p. 185.)
- [451] Ben Noble. Methods for computing the moore–penrose generalized inverse and related matters. In M. Z. Nashed, editor, *Generalized Inverses and Applications*, pages 245–302. Academic Press, Inc., New York, 1976. (Cited on pp. 228, 349.)
- [452] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, New York, second edition, 2006. (Cited on p. 432.)
- [453] W. Oettli and W. Prager. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numer. Math.*, 6:404–409, 1964. (Cited on p. 112.)
- [454] D. P. O’Leary. Robust regression computation using iteratively reweighted least squares. *SIAM J. Matrix Anal. Appl.*, 11:466–480, 1990. (Cited on p. 346.)

- [455] D. P. O'Leary and J. A. Simmons. A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems. *SIAM J. Sci. Statist. Comput.*, 2:474–489, 1981. (Cited on pp. 690, 718.)
- [456] James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970. (Cited on pp. 361, 364.)
- [457] E. E. Osborne. On pre-conditioning of matrices. *J. Assoc. Comput. Mach.*, 7:338–345, 1960. (Cited on p. 504.)
- [458] M. R. Osborne. *Finite Algorithms in Optimization and Data Analysis*. John Wiley, New York, 1985. (Cited on p. 346.)
- [459] A. M. Ostrowski. On the convergence of the Rayleigh quotient iteration for computation of the characteristic roots and vectors I–VI. *Arch. Rational Mech. Anal.*, pages 1:233–241, 2:423–428, 3:325–340, 3:341–347, 3:472–481, 4:153–165., 1958–59. (Cited on p. 588.)
- [460] A. M. Ostrowski. *Solution of Equations in Euclidian and Banach Spaces*. Academic Press, New York, third edition, 1973. (Cited on p. 432.)
- [461] Chris C. Paige, Beresford N. Parlett, and Henk A. van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numer. Linear Algebra Appl.*, 2:115–133, 1995. (Cited on p. 719.)
- [462] Christopher C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*. PhD thesis, University of London, 1971. (Cited on pp. 705, 719.)
- [463] Christopher C. Paige. Computer solution and perturbation analysis of generalized linear least squares problems. *Math. Comp.*, 33:171–184, 1979. (Cited on p. 317.)
- [464] Christopher C. Paige. Some aspects of generalized QR factorizations. In M. G. Cox and Sven J. Hammarling, editors, *Reliable Numerical Computation*, pages 71–91. Clarendon Press, Oxford, UK, 1990. (Cited on p. 351.)
- [465] Christopher C. Paige, Miroslav Rozložník, and Z. Strakoš. Modified Gram–Schmidt (MGS), least squares, and backward stability of MGS–GMRES. *SIAM J. Matrix Anal. Appl.*, 28(1):264–284, 2006. (Cited on p. 662.)
- [466] Christopher C. Paige and M. A. Saunders. Least squares estimation of discrete linear dynamic systems using orthogonal transformations. *SIAM J. Numer. Anal.*, 14:180–193, 1977. (Cited on p. 293.)
- [467] Christopher C. Paige and Michael Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975. (Cited on p. 718.)
- [468] Christopher C. Paige and Michael A. Saunders. Toward a generalized singular value decomposition. *SIAM J. Numer. Anal.*, 18:398–405, 1981. (Cited on pp. 211, 555.)
- [469] Christopher C. Paige and Michael A. Saunders. Algorithm 583: LSQR. sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8:195–209, 1982. (Cited on p. 287.)
- [470] Christopher C. Paige and Michael A. Saunders. LSQR. an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8:43–71, 1982. (Cited on pp. 287, 653, 658.)

- [471] Christopher C. Paige and Z. Strakoš. Unifying least squares, total least squares and data least squares. In S. Van Huffel and P. Lemmerling, editors, *Total Least Squares and Errors-in-Variables Modeling*, pages 25–34. Kluwer Academic Publishers, Dordrecht, 2002. (Cited on p. 286.)
- [472] Christopher C. Paige and Zdeněk Strakoš. Scaled total least squares fundamentals. *Numer. Math.*, 91:1:117–146, 2002a. (Cited on p. 351.)
- [473] Christopher C. Paige and M. Wei. History and generality of the CS decomposition. *Linear Algebra Appl.*, 208/209:303–326, 1994. (Cited on p. 211.)
- [474] A. T. Papadopoulos, Iain S. Duff, and Andrew J. Wathen. A class of incomplete orthogonal factorization methods II: Implementation and results. *BIT*, 45(1):159–179, 2005. (Cited on p. 684.)
- [475] Beresford N. Parlett. A recurrence among the elements of functions of triangular matrices. *Linear Algebra Appl.*, 14:117–121, 1976. (Cited on p. 565.)
- [476] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ, 1980. (Cited on pp. 487, 587, 698.)
- [477] Beresford N. Parlett. The new qd algorithm. *Acta Numerica*, 4:459–491, 1995. (Cited on p. 589.)
- [478] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics 20. SIAM, Philadelphia, PA, 1998. Republication of work published in 1980 by Prentice-Hall, Englewood Cliffs. (Cited on pp. 234, 255, 464, 508, 510, 511, 548, 587.)
- [479] Beresford N. Parlett and Osni A. Marques. An implementation of the dqds algorithm (positive case). *Linear Algebra Appl.*, 309:217–259, 2000. (Cited on p. 589.)
- [480] Beresford N. Parlett and J. K. Reid. Tracking the progress of the Lanczos algorithm for large symmetric eigenvalue problems. *IMA J. Numer. Anal.*, 1:135–155, 1981. (Cited on p. 719.)
- [481] Beresford N. Parlett and Christian Reinsch. Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numer. Math.*, 13:293–304, 1969. (Cited on p. 504.)
- [482] Beresford N. Parlett and D. S. Scott. The Lanczos algorithm with selective orthogonalization. *Math. Comput.*, 33:217–238, 1979. (Cited on p. 719.)
- [483] S. V. Parter. The use of linear graphs in Gauss elimination. *SIAM Review*, 3:119–130, 1961. (Cited on p. 186.)
- [484] G. Peters and James H. Wilkinson. The least squares problem and pseudo-inverses. *Comput. J.*, 13:309–316, 1970. (Cited on pp. 227, 349.)
- [485] G. Peters and James H. Wilkinson. On the stability of Gauss–Jordan elimination with pivoting. *Comm. Assos. Comput. Mach.*, 18:20–24, 1975. (Cited on p. 68.)
- [486] E. Picard. 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. (Cited on p. 349.)
- [487] R. L. Plackett. A historical note on the method of least squares. *Biometrika*, 36:456–460, 1949. (Cited on p. 349.)
- [488] R. L. Plackett. The discovery of the method of least squares. *Biometrika*, 59:239–251, 1972. (Cited on p. 349.)

- [489] George Poole and Larry Neal. Gaussian elimination: When is scaling beneficial? *Linear Algebra Appl.*, 173:309–324, 1992. (Cited on p. 186.)
- [490] A. Pothen and C. J. Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Software*, 16:303–324, 1990. (Cited on p. 302.)
- [491] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations*. Gordon and Breach, London, UK, 1970. (Cited on p. 431.)
- [492] M. J. D. Powell and J. K. Reid. On applying Householder’s method to linear least squares problems. In A. J. H. Morell, editor, *Information Processing 68. Proceedings of the IFIP Congress 68*, pages 122–126. North-Holland, Amsterdam, 1969. (Cited on p. 316.)
- [493] V. Pták. A remarkon the Jordan normal form of matrices. *Linear Algebra Appl.*, 310:5–7, 2000. (Cited on p. 587.)
- [494] Philip Rabinowitz. *Numerical Methods for Non-Linear Algebraic Equations*. Gordon and Breach, London, UK, 1970. (Cited on p. 432.)
- [495] Louis B. Rall. *Automatic Differentiation. Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1981. (Cited on p. 431.)
- [496] Lord Rayleigh. On the calculation of the frequence of vibration of a system in its gravest mode with an example from hydrodynamics. *Philos. Mag.*, 47:556–572, 1899. (Cited on p. 719.)
- [497] L. Reichel and W. B. Gragg. FORTRAN subroutines for updating the QR decomposition. *ACM Trans. Math. Software*, 16:369–377, 1990. (Cited on p. 276.)
- [498] J. K. Reid. A note on the least squares solution of a band system of linear equations by Householder reductions. *Comput J.*, 10:188–189, 1967. (Cited on p. 351.)
- [499] J. K. Reid. On the method of conjugate gradients for the solution of large systems of linear equations. In J. K. Reid, editor, *Large Sparse Sets of Linear Equations*, pages 231–254. Academic Press, New York, 1971. (Cited on p. 629.)
- [500] John K. Reid. A note on the stability of Gaussian elimination. *J. Inst. Maths. Applics.*, 8:374–375, 1971. (Cited on p. 718.)
- [501] Christian H. Reinsch. Smoothing by spline functions. *Numer. Math.*, 16:451–454, 1971. (Cited on p. 328.)
- [502] J. L. Rigal and J. Gaches. On the compatibility of a given solution with the data of a linear system. *J. Assoc. Comput. Mach.*, 14(3):543–548, 1967. (Cited on p. 112.)
- [503] W. Ritz. Über eine neue Method zur Lösung gewisser Variationsprobleme der Mathematischen Physik. *J. Rein. Angew. Math.*, 136:1–61, 1909. (Cited on p. 719.)
- [504] Sara Robinson. Toward an optimal algorithm for matrix multiplication. *SIAM News*, 38(4):2–3, 2005. (Cited on p. 32.)
- [505] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Read, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, New York, 1972. (Cited on p. 186.)
- [506] Axel Ruhe. Rational Krylov algorithms for nonsymmetric eigenvalue problems. In G. H. Golub, A. Greenbaum, and M. Luskin, editors, *Recent Advances in Iterative Methods*, IMA Series in Mathematics and its Applications 60, pages 149–164. Springer Verlag, New York, 1993. (Cited on p. 716.)

- [507] Axel Ruhe. Rational Krylov: A practical algorithms for large sparse nonsymmetric matrix pencils. *SIAM J. Sci. Comput.*, 10(5):1535–1551, 1998. (Cited on p. 716.)
- [508] Axel Ruhe and Per-Åke Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Rev.*, 22:3:318–337, 1980. (Cited on p. 432.)
- [509] Sigfried M. Rump. Fast and parallel interval arithmetic. *BIT*, 39(3):534–554, 1999. (Cited on p. 131.)
- [510] Sigfried M. Rump. INTLAB—INTerval LABoratory. In T. Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. (Cited on pp. 129, 131, 131.)
- [511] Heinz Rutishauser. Solution of eigenvalue problems with the LR-transformation. *Nat. Bur. Standards Appl. Math. Ser.*, 49:47–81, 1958. (Cited on p. 491.)
- [512] Heinz Rutishauser. The Jacobi method for real symmetric matrices. *Numer. Math.*, 9:1–10, 1966. (Cited on p. 536.)
- [513] Heinz Rutishauser. Simultaneous iteration method for symmetric matrices. *Numer. Math.*, 16:205–223, 1970. Also published as Contribution II/9 in the Handbook. (Cited on pp. 588, 719.)
- [514] Yosef Saad. On the rates of convergence of the Lanczos and block-Lanczos methods. *SIAM J. Numer. Anal.*, 17(5):687–706, 1980. (Cited on p. 719.)
- [515] Yosef Saad. Variations on Arnoldi’s method for computing eigenelements of large unsymmetric matrices. *Linear Algebra Appl.*, 34:269–295, 1980. (Cited on p. 719.)
- [516] Yosef Saad. *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York, 1992. (Cited on pp. 468, 587.)
- [517] Yosef Saad. ILUT: A dual threshold incomplete LU factorization. *Numer. Linear Algebra Appl.*, 1:387–402, 1994. (Cited on p. 684.)
- [518] Yosef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM Publications., Philadelphia, PA, second edition, 2003. (Cited on pp. 648, 650, 712, 717, 719.)
- [519] Yosef Saad and Henk A. van der Vorst. Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.*, 123:1–33, 2000. (Cited on p. 717.)
- [520] Werner Sautter. Fehleranalyse für die Gauss-Elimination zur Berechnung der Lösung minimaler Länge. *Numer. Math.*, 30:165–184, 1978. (Cited on p. 349.)
- [521] Erhard Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. 1 Teil. Entwicklung willkürlichen Funktionen nach Systemen vorgeschribener. *Math. Ann.*, 63:433–476, 1907. (Cited on p. 349.)
- [522] W. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31:1–10, 1966. (Cited on p. 332.)
- [523] Robert S. Schreiber. A new implementation of sparse Gaussian elimination. *ACM Trans. Math. Software*, 8(3):256–276, 1982. (Cited on p. 186.)
- [524] Günther Schulz. Iterativ Berechnung der reciproken Matrize. *Z. Angew. Math. Mech.*, 13:57–59, 1933. (Cited on p. 572.)
- [525] Issai Schur. Über die characteristischen Wurzeln einer linearen Substitution mit einer Anwendung auf die Theorie der Integral Gleichungen. *Math. Ann.*, 66:448–510, 1909. (Cited on p. 444.)
- [526] Issai Schur. Über potenzreihen, die in Innern des Einheitskreise beschränkt sind. *J. reine angew. Math.*, 147:205–232, 1917. (Cited on p. 185.)

- [527] Hubert Schwetlick and V. Tiller. Numerical methods for estimating parameters in nonlinear models with errors in the variables. *Technometrics*, 27:17–24, 1985. (Cited on p. 396.)
- [528] Jennifer A. Scott and Yifan Hu. Experiences of sparse direct symmetric solvers. Technical Report RAL-TR-2005-014, Computational Science and Engineering Department, Atlas Centre, Rutherford Appleton Laboratory, Oxfordshire OX11 0QX, England, 2005. (Cited on p. 187.)
- [529] H. D. Simon. Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear Algebra Appl.*, 61:101–131, 1984. (Cited on pp. 706, 719.)
- [530] H. D. Simon and Hongyan Zha. Low-rank matrix approximation using the Lanczos bidiagonalization process with applications. *SIAM J. Sci. Comput.*, 21(6):2257–2274, 2000. (Cited on p. 708.)
- [531] Lennart Simonsson. *Subspace Computations via Matrix Decompositions and Geometric Optimization*. PhD thesis, Linköping Studies in Science and Technology No. 1052, Linköping, Sweden, 2006. (Cited on p. 588.)
- [532] Sanja Singer and Saša Singer. Skew-symmetric differential qd algorithm. *Appl. Num. Anal. Comp. Math.*, 2(1):134–151, 2005. (Cited on p. 589.)
- [533] Robert D. Skeel. Scaling for stability in Gaussian elimination. *J. Assoc. Comput. Mach.*, 26:494–526, 1979. (Cited on pp. 124, 186.)
- [534] Robert D. Skeel. Iterative refinement implies numerical stability for Gaussian elimination. *Math. Comput.*, 35:817–832, 1980. (Cited on pp. 128, 186.)
- [535] G. L. G. Sleijpen and Henk A. van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17:401–425, 1996. (Cited on p. 716.)
- [536] B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystems Routines—EISPACK Guide*. Springer-Verlag, New York, second edition, 1976. (Cited on pp. 133, 186, 587.)
- [537] Inge Söderkvist. Perturbation analysis of the orthogonal Procrustes problem. *BIT*, 33(4):687–694, 1993. (Cited on p. 333.)
- [538] Torsten Söderström and G. W. Stewart. On the numerical properties of an iterative method for computing the Moore–Penrose generalized inverse. *SIAM J. Numer. Anal.*, 11(1):61–74, 1974. (Cited on p. 573.)
- [539] Peter Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989. (Cited on p. 719.)
- [540] D. C. Sorensen. Implicit application of polynomial filters in a k -step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992. (Cited on p. 719.)
- [541] D. C. Sorensen. Numerical methods for large eigenvalue problems. *Acta Numerica*, 11:519–584, 2002. (Cited on pp. 719, 720.)
- [542] G. W. Stewart. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Review*, 15(4):727–764, 1973. (Cited on p. 473.)
- [543] G. W. Stewart. *Introduction to Matrix Computations*. Academic Press, New York, 1973. (Cited on pp. 185, 204, 474, 487, 552.)
- [544] G. W. Stewart. The economical storage of plane rotations. *Numer. Math.*, 25:137–138., 1976. (Cited on p. 239.)

- [545] G. W. Stewart. On the perturbation of pseudo-inverses, projections, and linear least squares problems. *SIAM Review*, 19(4):634–662, 1977. (Cited on p. 202.)
- [546] G. W. Stewart. An updating algorithm for subspace tracking. *IEEE Trans. Signal Process.*, 40:1535–1541, 1992. (Cited on p. 351.)
- [547] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35(4):551–556, 1993. (Cited on p. 349.)
- [548] G. W. Stewart. Updating a rank-revealing ULV decomposition. *SIAM J. Matrix Anal. Appl.*, 14:494–499, 1993. (Cited on p. 351.)
- [549] G. W. Stewart. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, Philadelphia, PA, 1998. (Cited on pp. 30, 184, 184.)
- [550] G. W. Stewart. The decompositional approach to matrix computation. *Computing in Science and Engineering*, 2(1):50–59, 2000. (Cited on p. 2.)
- [551] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems*. SIAM, Philadelphia, PA, 2001. (Cited on pp. 546, 587, 719.)
- [552] G. W. Stewart and Ji guang. Sun. *Matrix Perturbation Theory*. Academic Press, Boston, MA, 1990. (Cited on pp. 20, 27, 184, 463, 549, 549, 587, 587, 589.)
- [553] M. Stewart and G. W. Stewart. On hyperbolic triangularization: Stability and pivoting. *SIAM J. Matrix Anal. Appl.*, 19:4:8471–860, 1998. (Cited on p. 334.)
- [554] E. Stiefel. Ausgleichung ohne Aufstellung der Gausschen Normalgleichungen. *Wiss. Z. Tech. Hochsch. Dresden*, 2:441–442, 1952/53. (Cited on p. 718.)
- [555] S. M. Stiegler. Gauss and the invention of least squares. *Ann. Statist.*, 9:465–474, 1981. (Cited on pp. 231, 348.)
- [556] J. Stoer. On the numerical solution of constrained least squares problems. *SIAM J. Numer. Anal.*, 8:382–411, 1971. (Cited on p. 427.)
- [557] G. Strang. A proposal for Toeplitz matrix computations. *Stud. Appl. Math.*, 74:171–176, 1986. (Cited on p. 719.)
- [558] Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969. (Cited on pp. 32, 142.)
- [559] Torsten Ström. On logarithmic norms. *SIAM J. Numer. Anal.*, 12(5):741–753, 1975. (Cited on p. 566.)
- [560] Ji-guang Sun and Zheng Sun. Optimal backward perturbation bounds for under-determined systems. *SIAM J. Matrix Anal. Appl.*, 18(2):393–402, 1997. (Cited on p. 350.)
- [561] J. J. Sylvester. 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. *Philosophical Magazine*, 2:138–142, 1852. (Cited on p. 80.)
- [562] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1:146–159, 1972. (Cited on p. 172.)
- [563] A. H. Taub. *John von Neumann Collected Works*, volume V, Design of Computers, Theory of Automata and Numerical Analysis. Pergamon Press, Oxford, 1963. (Cited on p. 185.)
- [564] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963. (Cited on p. 323.)

- [565] W. F. Tinney and J. W. Walker. Direct solution of sparse network equations by optimally ordered triangular factorization. *Proc. IEEE*, 55:1801–1809, 1967. (Cited on p. 186.)
- [566] Françoise Tisseur. Newton’s method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 22(4):1038–1057, 2001.
- [567] Françoise Tisseur and Karl Meerbergen. The quadratic eigenvalue problem. *SIAM Review*, 43(2):235–286, 2001. (Cited on p. 589.)
- [568] Otto Toeplitz. Das algebraische Analogon zu einem Satze von Fejér. *Math. Z.*, 2:187–197, 1918. (Cited on p. 589.)
- [569] Sivan Toledo. Locality of reference in LU decomposition with partial pivoting. *SIAM J. Matrix Anal. Appl.*, 18(4):1065–1081, 1997. (Cited on p. 146.)
- [570] Andrea Toselli and Olof Widlund. *Domain Decomposition Methods—Algorithms and Theory*. Number 34 in Computational Mathematics. Springer, New York, 2005. xv+450 pages. ISBN 978-3-540-20696-5. (Cited on p. 717.)
- [571] Lloyd N. Trefethen. Three mysteries of Gaussian elimination. *SIGNUM Newsletter*, 20(4):2–5, 1985. (Cited on pp. 1, 42.)
- [572] Lloyd N. Trefethen. Pseudospectra of linear operators. *SIAM Review*, 39(3):383–406, 1997. (Cited on p. 589.)
- [573] Lloyd N. Trefethen. Calculation of pseudospectra. *Acta Numerica*, 8:247–295, 1999. (Cited on p. 589.)
- [574] Lloyd N. Trefethen. *Spectral Methods in MATLAB*. SIAM, Philadelphia, 2000. (Cited on p. 587.)
- [575] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997. (Cited on p. 290.)
- [576] Lloyd N. Trefethen and Mark Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton Univ. Press, Princeton, MA, 2006. (Cited on p. 589.)
- [577] Lloyd N. Trefethen and Robert S. Schreiber. Average-case stability of Gaussian elimination. *SIAM J. Matrix Anal. Appl.*, 11(3):335–360, 1990. (Cited on pp. 57, 119.)
- [578] W. F. Trench. An algorithm for the inversion of finite Toeplitz matrices. *J. SIAM*, 12:515–522, 1964. (Cited on p. 187.)
- [579] M. J. Tsatsomeros. Principal pivot transforms. *Linear Algebra Appl.*, 307:151–165, 2000. (Cited on p. 351.)
- [580] A. M. Turing. Rounding-off errors in matrix processes. *Quart. J. Mech. Appl. Math.*, 1:287–308, 1948. (Cited on p. 185.)
- [581] M. Van Barel and Nicola Mastronardi. A note on the representation and definition of semiseparable matrices. *Numer. Linear Algebra Appl.*, 12:839–858, 2005. (Cited on p. 187.)
- [582] A. van der Sluis. Condition numbers and equilibration of matrices. *Numer. Math.*, 14:14–23, 1969. (Cited on p. 680.)
- [583] A. van der Sluis and G. Veltkamp. Restoring rank and consistency by orthogonal projection. *Linear Algebra Appl.*, 28:257–278, 1979. (Cited on p. 203.)

- [584] Henk A. van der Vorst. Computational methods for large eigenvalue problems. In Philippe G. Ciarlet and F. Cucker, editors, *Handbook of Numerical Analysis*, volume VIII, pages 3–179. North Holland Elsevier, Amsterdam, 2002. (Cited on p. 717.)
- [585] Henk A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, Cambridge, UK, 2003. (Cited on p. 717.)
- [586] Henk A. van der Vorst and A. van der Sluis. The rate of convergence of conjugate gradients. *Numer. Math.*, 48:543–560, 1986. (Cited on p. 718.)
- [587] Sabine Van Huffel and Joos Vandewalle. *The Total Least Squares Problem; Computational Aspects and Analysis*. SIAM, Philadelphia, PA, 1991. (Cited on pp. 337, 351, 520.)
- [588] Charles F. Van Loan. Generalizing the singular value decomposition. *SIAM J. Numer. Anal.*, 13:76–83, 1976. (Cited on p. 555.)
- [589] Charles F. Van Loan. The ubiquitous Kronecker product. *J. Comput. Appl. Math.*, 123:85–100, 2000. (Cited on p. 186.)
- [590] Raf Vandebril, Marc Van Barel, and Nicola Mastronardi. *Matrix Computations and Semiseparable Matrices*, volume 1; Linear Systems. The Johns Hopkins University Press, Baltimore, MD, 2007. (Cited on p. 187.)
- [591] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer’s International Series. Springer, New York, third edition, 2008. (Cited on p. 432.)
- [592] J. M. Varah. On the solution of block-tridiagonal systems arising from certain finite-difference equations. *Math. Comp.*, 26(120):859–869, 1972. (Cited on p. 141.)
- [593] J. M. Varah. On the separation of two matrices. *SIAM J. Numer. Anal.*, 16(2):216–222, 1979. (Cited on p. 473.)
- [594] Richard S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, 1962. (Cited on pp. 453, 602, 717.)
- [595] Richard S. Varga. *Geršgorin and his Circles*. Number 36 in Series in Computational Mathematics. Springer-Verlag, Berlin, Heidelberg, New York, 2004. (Cited on p. 587.)
- [596] V. V. Voevodin. The problem of a non-selfadjoint generalization of the conjugate gradient method has been closed. *USSR Comput. Math. Mash. Phys.*, 23:143–144, 1983. (Cited on p. 660.)
- [597] Urs von Matt. The orthogonal qd-algorithms. *SIAM J. Sci. Comput.*, 18(4):1163–1186, 1997. (Cited on p. 588.)
- [598] Bertil Waldén, Rune Karlsson, and Ji guang. Sun. Optimal backward perturbation bounds for the linear least squares problem. *Numer. Linear Algebra Appl.*, 2:271–286, 1995. (Cited on p. 349.)
- [599] X. Wang, K. A. Gallivan, and R. Bramley. CIMGS: An incomplete orthogonal factorization preconditioner. *SIAM J. Sci. Comput.*, 18:516–536, 1997. (Cited on p. 684.)
- [600] Robert C. Ward. Eigensystem computation for skew-symmetric matrices and a class of symmetric matrices. *ACM Trans. Math. Software*, 4(3):278–285, 1978. (Cited on p. 589.)
- [601] David S. Watkins. Understanding the QR algorithm. *SIAM Review*, 24:427–440, 1982. (Cited on p. 494.)

- [602] David S. Watkins. *Fundamentals of Matrix Computation*. Wiley-Interscience, New York, second edition, 2002. (Cited on pp. 184, 587.)
- [603] David S. Watkins. *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*. SIAM, Philadelphia, PA, 2007. (Cited on p. 589.)
- [604] David S. Watkins. The QR algorithm revisited. *SIAM Review*, 50(1):133–145, 2008. (Cited on p. 588.)
- [605] Per-Åke Wedin. On pseudoinverses of perturbed matrices. Tech. Report, Department of Computer Science, Lund University, Sweden, 1969. (Cited on p. 203.)
- [606] Per-Åke Wedin. Perturbation theory for pseudo-inverses. *BIT*, 9:217–232, 1973. (Cited on pp. 203, 221.)
- [607] H. Weyl. Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen. *Math. Ann.*, 71:441–469, 1911. (Cited on p. 349.)
- [608] H. Wielandt. Das Iterationsverfahren bei nicht selbstadjungierten linearen Eigenwertaufgaben. *Math. Z.*, 50:93–143, 1944. (Cited on pp. 480, 482, 588.)
- [609] James H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. Assos. Comput. Mach.*, 8:281–330, 1961. (Cited on pp. 56, 59, 118.)
- [610] James H. Wilkinson. *Rounding Errors in Algebraic Processes*. Notes on Applied Science No. 32. Her Majesty's Stationery Office, London, UK, 1963. Republished in 1994 by Dover, Mineola, NY. (Cited on p. 33.)
- [611] James H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965a. (Cited on pp. 33, 118, 350, 350, 461, 465, 481, 507, 527, 587, 588.)
- [612] James H. Wilkinson. Global convergence of tridiagonal QR algorithm with origin shifts. *Linear Algebra Appl.*, 1:409–420, 1968. (Cited on p. 510.)
- [613] James H. Wilkinson. A priori error analysis of algebraic processes. In *Proceedings International Congress Math.*, pages 629–639. Izdat. Mir, Moscow, 1968. (Cited on p. 122.)
- [614] James H. Wilkinson and C. Reinsch, editors. *Handbook for Automatic Computation. Vol. II, Linear Algebra*. Springer-Verlag, New York, 1971. (Cited on pp. 133, 150, 504, 508, 550, 550, 587, 745.)
- [615] Herman Wold. Estimation of principal components and related models by iterative least squares. In P. R. Krishnaiah, editor, *Multivariate Analysis*, pages 391–420. Academic Press, New York, 1966. (Cited on p. 350.)
- [616] Svante Wold, Axel Ruhe, Herman Wold, and W. J. Dunn. The collinearity problem in linear regression, the partial least squares (pls) approach to generalized inverses. *SIAM J. Sci. Stat. Comput.*, 5:735–743, 1984. (Cited on pp. 350, 351.)
- [617] Max A. Woodbury. Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton, 1950. (Cited on p. 12.)
- [618] Margaret H. Wright. *Numerical Methods for Nonlinearly Constrained Optimization*. Ph.D. thesis, Stanford University, Stanford, CA, 1976. (Cited on p. 432.)
- [619] Stephen J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, PA, 1997. (Cited on p. 432.)
- [620] Thomas George Wright. *Algorithms and Software for Pseudospectra*. PhD thesis, Numerical Analysis Group, Oxford University Computing Laboratory, Oxford, UK, 2002. (Cited on p. 589.)

- [621] M. Yannakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Math.*, 2(1):77–79, 1990. (Cited on p. 161.)
- [622] David M. Young. *Iterative methods for solving partial differential equations of elliptic type*. Ph.D. thesis, Harward University, Cambridge, MA, 1950. (Cited on pp. 602, 605.)
- [623] David M. Young. *Iterative Solution of Large Linear Systems*. Academic Press, New York, 1971. (Cited on pp. 605, 608, 613, 717.)
- [624] Fuzhen Zhang, editor. *The Schur Complement and Its Application*. Number 4 in Numerical Methods and Algorithms. Springer-Verlag, New York, 2005. (Cited on p. 185.)

Index

A -orthogonal vectors, 624
 J -orthogonal, 319

A -norm, 621
adjacency set, 156
adjoint matrix, 13
algorithm
 LDL^T , 71
 1-norm estimator, 116
 back substitution, 43
 band, 91
 band LU, 90
 band-Cholesky, 92
 Bi-CG, 668
 Bi-CGS, 671, 672
 block Cholesky, 139
 block LU factorization, 138
 block-Cholesky factorization, 139
 CG, 629
 CGLS, 649
 CGME, 650
 Euler–Newton method, 374
 forward substitution
 band, 91
 Gaussian elimination, 48, 56
 Givens rotations, 236
 Householder QR, 242
 Householder reflection, 234
 incomplete Cholesky, 683
 incomplete LU, 681
 IRLS, 346
 Lanczos, 635, 666
 LSQR, 653
 MGS
 least squares by, 259, 260
 minimum norm solution, 261
 modified Gram–Schmidt, 251
 orthogonal iteration, 713
 preconditioned CG, 675
 preconditioned CGLS, 677

preconditioned CGME, 677
Rayleigh–Ritz procedure, 696
recursive Cholesky factorization, 144
recursive LU factorization, 146
singular values by spectrum slicing,
 534
svd, 536, 542
tridiagonal spectrum slicing, 532
unstable, 111
analytic function
 of matrix, 562
angles between subspaces, 207–209
arithmetic
 complex, 35
 floating-point, 32–35
 standard model, 33
Armijo–Goldstein criterion, 366
Arnoldi’s method, 698–703
 polynomial restarting, 700–703
Arnoldi’s process, 660–665
array operations, 5
arrowhead matrix, 173, 298, 527
ART, 718
artificial .ill-conditioning, 110
artificial ill-conditioning, 108
augmented system, 192, 220, 248, 261,
 262
back substitution, 42
 band, 91
backward error, 225–226
 componentwise, 112
 normwise, 111
 optimal, 225
balancing a matrix, 503
Banach space, 361
Banachiewicz
 inversion formula, 11
band
 matrix, 86–99

- standard form of, 291
- bandwidth
 - of LU factors, 89
 - row, 216
- barrier function, 420
- Bauer–Fike’s theorem, 459
- BFGS update, 381
- bi-CG method, 668
- Bi-CGS, 670
- Bi-CGSTAB, 672
- bi-conjugate gradient, *see* Bi-CG method
- Bi-Lanczos method, 667
- bidiagonal matrix, 87
- bidiagonal reduction, 279–281
- bidiagonalization
 - for TLS, 342–344
- bilinear
 - function, 354
- biographical note
 - Banachiewicz, 11
 - Beltrami, 349
 - Cayley, 2
 - Cholesky, 75
 - Cimmino, 644
 - Frobenius, 22
 - Galerkin, 620
 - Geršgorin, 456
 - Golub, 515
 - Gram, 250
 - Grassmann, 475
 - Hadamard, 18
 - Hamilton, 554
 - Hessenberg, 87
 - Hestenes, 627
 - Householder, 3
 - Jacobi, 535
 - Jordan, C., 441
 - Jordan, W., 67
 - Kaczmarz, 647
 - Kronecker, 147
 - Krylov, A. N., 626
 - Lanczos, 634
 - Lord Rayleigh, 486
 - Markov, 579
 - Perron, 452
 - Riccati, 450
 - Ritz, 696
 - Schmidt, 250
 - Schur, 10
 - Stiefel, 627
 - Sylvester, 2
 - Tikhonov, 323
 - Toeplitz, 174
 - Wilkinson, 52
 - Young, 602
 - BLAS, 126
 - block
 - algorithms, 136–139
 - angular form, 297–300
 - doubly bordered, 298
 - angular problem
 - QR algorithm, 299
 - diagonally dominant, 138
 - triangular form, 171–172, 300–302
 - coarse decomposition, 301
 - fine decomposition, 301
 - block tridiagonal matrices, 686–688
 - bordered matrix, 149
 - bordering method, 63
 - Broyden’s method, 371
 - cancellation, 304
 - canonical form
 - Jordan, 441–443
 - Kronecker, 546
 - Cauchy
 - interlacing theorem, 465
 - matrix, 177
 - sequence, 359
 - systems, 177–180
 - Cauchy–Schwarz inequality, 20
 - Cauchy-like matrix, 178
 - Cayley
 - parameterization, 446
 - transform, 446
 - Cayley transform, 39
 - Cayley–Hamilton theorem, 443
 - central path, 420
 - CG, *see* conjugate gradient
 - CG method
 - preconditioned, 675–678
 - rate of convergence, 632
 - CGS, *see* classical Gram–Schmidt
 - chain rule, 355
 - characteristic equation, 434
 - characteristic polynomial, 435
 - Chebyshev
 - acceleration, 610–613
 - polynomial, 611–613
 - complex, 712

- semi-iterative method, 612
- Cholesky factorization, 214
 - backward error, 121, 122
 - band, 75–94
 - block incomplete, 687
 - incomplete, 683
 - sparse, 303
 - symbolic, 163
- Cimmino’s method, 644
- circulant matrix, 690
- clique, 156, 164, 303
- colon notation, 9
- column pivoting
 - in Gram–Schmidt, 252
 - in Householder QR, 244
- column scaling, 222
 - optimal, 224
- column space, *see* range
- complete QR factorization, 269–271
- complete space, 359
- complex
 - arithmetic, 35
- complex symmetric matrix, 83
- componentwise perturbation bound, 109
- computer memory, 134
- condition estimation, 114–117, 261–262
 - Hager’s, 115
 - LINPACK’s, 115
- condition estimator
 - Hager’s, 473
- condition number
 - general matrix, 220
 - of matrix, 105
- conditional adjustment, 191
- conjugate gradient method, 627–634
- conjugate residual method, 630, 638
- consistently ordered, 604
- constrained problem
 - bounded, 430–431
 - linear inequality, 422–431
 - quadratic inequality, 323–327
- continuation methods, 372–374
- contraction mapping, 359
- contraction mapping theorem, 360
- convergence
 - asymptotic rate, 599
 - average rate, 599
 - conditions for, 597
 - of iterative methods, 597–602
 - of matrix power series, 560–562
- of vectors and matrices, 24
- convergent matrix, 597
- convex
 - function, 26
 - set, 405
- core subproblem, 286
- cost vector, 405
- covariance matrix, 194, 196, 318
 - block angular problem, 299
 - computing, 216–219
 - method, 275
- Craig’s method, 650
- Cramer’s rule, 17
- Crawford number, 548
- Crout’s algorithm, 61
- CS decomposition, 209–211
- Cuthill–McKee, 162
- cycle in graph, 156
- cyclic
 - matrix, 453
 - reduction, 95
- data least squares problem, 336
- decomposition
 - block diagonal, 448
 - CS, 209–211
 - Schur, 443–448
 - SVD, 25
- defect matrix, 687
- deflation, 701
 - of eigenproblem, 470, 480–482
 - of eigenvalue, 444
- degeneracy, 415–417
- derivative
 - directional, 356
 - higher, vector-valued, 356
 - partial, 355
- descent direction, 366, 379
- determinant identity
 - Sylvester’s, 18
- diagonal matrix, 5
- diagonal scaling
 - optimal, 680
- diagonally dominant, 70
- differentiable function, 355
- differential, 355
- differentiation
 - algorithmic, 371
 - automatic, 371
 - symbolic, 371

- direct elimination
 - method of, 322
- direct methods
 - fast, 688–690
- directional derivative, 356
- discrepancy principle, 282
- discrepancy principle, 326
- displacement equation, 178
- distance
 - between subspaces, 208
- distance between subspaces, 474
- distance to singularity, 107
- distribution function, 193
- divide and conquer
 - tridiagonal eigenproblem, 526–531
- domain decomposition, 675
- dominant eigenvalue, 437
- dominant invariant subspace, 489
- Doolittle's algorithm, 61
- double pendulum, 376
- downdating, 274
- Drazin inverse, 580
- drop tolerance, 684
- dual
 - norm, 20
 - vector, 20
- Dulmage–Mendelsohn form, 171, 301
- eigenspace, 439
- eigenvalue
 - algebraic multiplicity, 440
 - by spectrum slicing, 531–533
 - defective, 441
 - geometric multiplicity, 440
 - Jacobi's method, 535–538
 - large problems, 695–716
 - of Kronecker product, 593
 - of Kronecker sum, 594
 - of matrix, 433
 - perturbation, 459–468
 - power method, 477–484
 - problem
 - unsymmetric, 661, 698
 - residual error bound, 462–463, 466–471
 - semisimple, 441
 - subspace iteration, 487–490
- eigenvalue problem
 - generalized, 544–556, 714–716
- eigenvector
 - of matrix, 433
 - perturbation, 459–468
 - element growth, 56, 59
 - in complete pivoting, 118
 - in partial pivoting, 118
 - partial pivoting, 118
 - elementary
 - orthogonal matrix, 232–239
 - unitary rotation, 543
 - elimination
 - left-looking, 140
 - right-looking, 140
 - elimination tree, 310
 - post ordering of, 310
 - topological ordering of, 310
 - elliptic norm, 20
 - empty matrix, 662
 - energy norm, 621
 - envelope
 - method, 99–100
 - of matrix, 100, 162
 - equivalence transformation, 545
 - equivalent orderings, 160
 - error
 - analysis
 - backward, 111
 - forward, 111
 - componentwise estimate, 262
 - floating-point rounding, 35
 - error bounds
 - a posteriori, 112
 - backward, 111
 - errors-in-variable model, 335
 - estimable parameter, 194
 - Euler
 - angles, 238
 - expansion, 25, 694
 - exchange operator, 319
 - expansion
 - Euler, 25, 694
 - Neumann, 25, 694
 - expected value, 193
 - exponential
 - fitting, 391
 - of matrix, 566–569
 - extended precision BLAS, 126, 264
 - Farkas' Lemma, 408
 - feasible point, 405
 - basic, 410

- degenerate, 410
feasible region, 405
FFT, 689
field of values, 548, 558
fill, 152
filter factor, 326, 656
Fischer's theorem, 204, 463
fixed point iteration, 358–360
flam, 30
flop count
 LDL^T factorization, 72
 band back substitution, 91
 band LU, 91
 condition estimation, 115
 Gauss–Jordan elimination, 68
 Gaussian elimination, 48
 Gram–Schmidt, 252
 Hessenberg system, 92
 Householder QR, 266
 inverse matrix, 67
 normal equations, 215
 QR algorithm for SVD, 518
 QR factorization, 243, 288
 banded, 291
 reduction to bidiagonal form, 280
 reduction to Hessenberg form, 496, 498
 triangular system, 48
 tridiagonal system, 95
forest, 156
forward substitution, 43
 band, 91
Fréchet derivative, 355
Frank matrix, 503
Fredholm integral equation, 29
FSAI, 685
functional equation, 353
functions
 matrix-valued, 562–577
fundamental subspaces, 27, 200

gap in spectrum, 698
Gateaux derivative, 356
Gauss–Jordan Elimination, 67–68
Gauss–Markov's theorem, 194
Gauss–Newton method, 383–388
 rate of convergence, 387
Gauss–Seidel's method, 594, 623
Gaussian elimination, 44–68
 backward error, 57
compact schemes, 61–63
rounding error analysis, 117–122
scaling invariance, 122
GE, *see* Gaussian elimination
generalized
 eigenvalue problem, 544–556, 714–716
 inverse, 201–202
 least squares, 314–332
 QR factorization, 318
 SVD, 555–556
geometric fitting, 400
Gershgorin
 disks, 456
 theorem, 70, 456–458
Givens rotation, 236–239
 unitary, 238
global convergence, 365–368
GMRES, 662–665
 preconditioned, 678
 restarted, 665
Golub–Kahan bidiagonalization, 279–281
GQR, *see* generalized QR factorization
grade
 of vector, 435
graded matrix, 503, 507
gradient vector, 355
Gram–Schmidt
 classical, 250
 modified, 251
 orthogonalization, 249–261
Gram–Schmidt
 flop count, 252
graph
 bipartite, 172
 clique in, 164, 303
 connected, 156, 171
 cycle in, 156
 directed, 155
 elimination, 157
 filled, 157
 labeled, 155
 ordered, 155, 156
 path in, 156
 planar, 167
 representation of matrix, 155–159
 separator, 165
 strongly connected, 156
 undirected, 156
Grassmann manifold, 475

- group inverse, 580
 growth ratio, 56
 GSVD, *see* generalized SVD, *see* generalized SVD
 Hölder inequality, 20
 Hadamard matrix, 40
 Hadamard product, 5
 Hadamard's inequality, 18, 86
 Hall property, 304
 Hamiltonian matrix, 554–555
 Hankel matrix, 174
 harmonic Ritz values, 697, 704
 Hermitian definite pair, 548
 Hermitian matrix, 13
 Hessenberg form
 reduction to, 495–498
 Hessenberg matrix, 91
 unreduced, 497
 Hessian matrix, 356, 358
 hierarchical memory, 134
 Hilbert matrix, 107
 condition of, 107
 homotopy, 372
 Hotelling, 480
 Householder
 reflector, 233–235
 vector, 233
 Huber's M-estimator, 345
 hyperbolic rotations, 320
 identity matrix, 6
 ill-conditioned
 artificial, 108, 110
 ill-posed problem, 29, 323–324
 ILU, *see* incomplete LU factorization 681
 ILUT factorization, 684
 implicit function, 356
 incomplete factorization
 block, 686–688
 Cholesky, 683
 incomplete LU factorization, 681–684
 incomplete QR factorization, 684
 incremental loading, 373
 indefinite least squares, 319–321
 indefinite least squares problem, 319
 inertia
 of symmetric matrices, 79–81
 inertia of matrix, 80
 initial basis, 417–418
 inner inverse, 201
 inner iteration, 687
 inner product
 accurate, 126
 error analysis, 33
 standard, 4
 instability
 irrelevant, 497
 INTLAB, 131
 invariant subspace, 439
 invariant subspaces
 perturbation of, 469–474
 inverse
 approximative, 25
 Drazin, 580
 generalized, 201–202
 group, 580
 inner, 201
 left, 213
 matrix, 7
 of band matrix, 98
 outer, 201
 product form of, 68
 inverse function, 356
 inverse iteration, 482–485
 shifted, 483
 inverse matrix, 572
 involutory matrix, 577
 IRLS, *see* iteratively reweighted least squares
 irreducible, 10
 iteration matrix, 596
 Gauss–Seidel, 596
 Jacobi, 596
 SOR, 603
 SSOR, 608
 iterative method
 block, 596–597, 609–610
 convergent, 597
 error reducing, 646
 least squares, 641–648
 residual reducing, 645
 rounding errors in, 613–616
 semiconvergent, 655
 stationary, 596
 symmetrizable, 609
 terminating, 613–617
 Toeplitz system, 690–693
 iterative method
 symmetrizable, 609
 iterative methods

- preconditioned, 673–693
- iterative refinement
 - error bound, 128
 - of solutions, 125–128
- iterative regularization, 655–659
- iteratively reweighted least squares, 344–347
- Jacobi transformation, 536
- Jacobi's method, 594, 645
 - classical, 537
 - cyclic, 537
 - for SVD, 538–542
 - sweep, 537
 - threshold, 537
- Jacobian, 355
- Jacobian matrix, 355
- Jordan canonical form, 441–443
- Kaczmarz's method, 647
- Kalman gain vector, 275
- Kantorovich inequality, 625
- kernel, *see* null space
- Krawczyck's method, 131
- Kronecker
 - least squares problem, 311–312
 - product, 147, 593
 - sum, 593, 594
 - symbol, 6
 - system, 147–149
- Kronecker product, 449
- Kronecker's canonical form, 546
- Krylov
 - subspace methods, 626
- Krylov matrix, 626
- Krylov subspace, 626–627
- Lagrange multipliers, 193
- Lanczos
 - bi-orthogonalization, 665–667
 - bidiagonalization, 651–652
 - decomposition, 636, 707
 - process, 634–638, 703–709
 - loss of orthogonality, 705
- Lanczos bidiagonalization, 707
- Landweber's method, 643–644, 655
- Laplace equation, 92, 592, 633
- latent root regression, 335
- least squares
 - banded problems, 216, 290–292
 - basic solution, 268
 - characterization of solution, 190–193
 - general problem, 199
 - generalized, 314–332
 - indefinite, 319–321
 - principle of, 189
 - problem, 189
 - solution, 189
 - total, 335–344
 - weighted, 314–317
 - with linear equality constraints, 321
- least squares fitting
 - of circles, 397–402
 - of ellipses, 397–402
- least squares method
 - nonlinear, 382–396
 - separable problem, 391–394
- least squares problem
 - indefinite, 319
 - Kronecker, 311–312
 - stiff, 223
 - weighted, 315
- left-inverse, 212
- left-looking, 140
- left-preconditioned system, 673
- Levenberg–Marquardt method, 388
- Levinson–Durbin recursion, 175
- line search, 366
- linear inequality constraints
 - active set algorithms, 426–431
 - basic transformations, 425–426
 - classification, 422–424
- linear model
 - general univariate, 196
 - standard, 194
- linear optimization, 405–421
 - dual problem, 418
 - duality, 418–419
 - duality gap, 419
 - interior point method, 420–421
 - primal problem, 418
 - standard form, 409
- linear programming, *see* linear optimization
- linear regression, 215
 - multiple, 215
- linear system
 - consistent, 47
 - ill-scaling, 125
 - overdetermined, 47

- scaling, 122–125
- scaling rule, 124
- underdetermined, 47
- linear transformation, 7
- linearly independent vectors, 6
- Lipschitz
 - condition, 359
 - constant, 359
- local minimum
 - necessary conditions, 379
- logarithm of matrix, 569–570
- logarithmic norm, 566
- look-ahead strategy, 668
- LSQI, *see* quadratic inequality constraints
- LSQR, 653
 - LU factorization, 49–52, 227
 - Doolittle's algorithm, 61
 - incomplete, 681–684
 - of rectangular matrix, 65
 - theorem, 50
 - Lyapunov's equation, 450
- M*-matrix, 601, 683
- magnitude
 - of interval, 129
- Markov chain, 579–583
- mathematically equivalent algorithms, 251
- matrix
 - adjoint, 13
 - arrowhead, 173
 - band, 86–99
 - bidiagonal, 87
 - block, 8
 - bordered, 149
 - complex symmetric, 71, 83
 - congruent, 80
 - consistently ordered, 604
 - defective, 441
 - derogatory, 442
 - diagonal, 5
 - diagonalizable, 440
 - diagonally dominant, 59–61, 138, 600
 - eigenvalue of, 433
 - eigenvector of, 433
 - elementary divisors, 443
 - elementary elimination, 52
 - elementary orthogonal, 232
 - empty, 662
 - exponential, 566–569
 - functions, 562–577
 - graded, 503, 507
 - Hamiltonian, 554–555
 - Hermitian, 13
 - Hessenberg, 91
 - idempotent, 197
 - identity, 6
 - ill-conditioned, 107
 - indefinite, 70
 - inverse, 7, 25, 65–68
 - involutory, 577
 - irreducible, 10, 94, 600
 - logarithm, 569–570
 - non-negative irreducible, 452
 - nonnegative, 451–453
 - normal, 4, 445
 - orthogonal, 7
 - permutation, 14
 - persymmetric, 4
 - positive definite, 59, 69–74
 - primitive, 452
 - property A, 604
 - quasi-triangular, 447
 - rank deficient, 7
 - Rayleigh quotient, 470, 471
 - reducible, 10, 156, 171
 - row stochastic, 579
 - scaled diagonally dominant, 507
 - semidefinite, 70
 - semiseparable, 181–183
 - sign function, 576–577
 - skew-Hermitian, 13
 - skew-symmetric, 84, 573
 - sparse, 150, 592
 - splitting, 595
 - square root, 571–572
 - symmetric, 70
 - symplectic, 554–555
 - totally positive, 121
 - trace, 435
 - trapezoidal form, 46
 - tridiagonal, 87
 - unitary, 41
 - variable-band, 99
 - well-conditioned, 106
- matrix approximation, 332
- matrix exponential
 - hump, 567
- matrix functionals, 578
- matrix multiplication, 30–32
 - error bound, 34

- fast, 31
- matrix pencil, 545
 - congruent, 546
 - equivalent, 545
 - regular, 545
 - singular, 545
- matrix splitting
 - Gauss–Seidel, 596
 - Jacobi, 596
- maximum matching, 172
- mean, 344
- median, 344
- method of steepest descent, 624–626
- MGS, *see* modified Gram–Schmidt
- MGS factorization
 - backward stability, 254
- midrange, 344
- minimal polynomial, 435, 443
 - of vector, 435
- minimal residual algorithm, 639
- minimax characterization
 - of eigenvalues, 463
 - of singular values, 204
- minimax property, 611
- minimum
 - global, 378
 - local, 378
- minimum degree ordering, 163–165
- minimum distance
 - between matrices, 205
- minimum norm solution, 191, 261
- MINRES, *see* minimal residual algorithm
- Moore–Penrose inverse, 200
- multifrontal method, 167, 307–311
 - for QR decomposition
 - update matrix, 309
- for QR factorization
 - data management, 310
- multigrid iteration, 675
- multilinear
 - symmetric mapping, 358
 - function, 354
 - mapping, 354
- nested dissection, 165–167
- Neumann expansion, 25, 694
- Newton step, 380
- Newton’s interpolation formula
 - for matrix functions, 586
- Newton’s method, 361–365
- damped, 366
- discretized, 368
- for least squares, 389–390
- for minimization, 380
- Newton–based methods, 469–475
- Newton–Kantorovich theorem, 363
- no-cancellation assumption, 152, 157
- node(s)
 - adjacent, 156
 - amalgamation of, 311
 - connected, 156
 - degree, 156
 - indistinguishable, 165
 - supernode, 311
- nonnegative
 - matrix, 451–453
- nonnegative least squares, 423
- norm
 - absolute, 22
 - consistent, 21
 - dual, 20
 - Frobenius, 22
 - logarithmic, 566
 - matrix, 21
 - monotone, 22
 - operator, 21
 - scaled, 20
 - spectral, 22
 - submultiplicative, 21
 - subordinate, 21
 - unitarily invariant, 23
 - vector, 19
 - weighted, 20
- normal
 - curvature matrix, 386
- normal equations, 190
 - accuracy of, 222–227
 - factored form, 641
 - iterative refinement, 226
 - method of, 213–216
 - modified, 191
 - of second kind, 642, 646
 - scaling of, 224
- normality
 - departure from, 446
- normalized residuals, 217
- null space, 27
 - numerical, 28
 - from SVD, 28
- null space (of matrix), 27

- null space method, 322
- numerical
 - radius, 562
 - numerical abscissa, 558, 567
 - numerical cancellation, 304
 - numerical differentiation, 369
 - errors, 369
 - optimal, 369
 - numerical radius, 558
 - numerical rank, 28, 30
 - by SVD, 28
 - oblique projection method, 697
 - odd-even reduction, 95
 - Oettli–Prager error bounds, 112
 - one-sided Jacobi SVD, 540–542
 - one-way dissection method, 165
 - operation count, 31
 - ordering
 - Markowitz, 168
 - minimum degree, 163–165
 - nested dissection, 165–167
 - reverse Cuthill–McKee, 162
 - orthogonal, 7
 - complement, 7
 - distance, 395
 - iteration, 488, 713
 - matrix, 7
 - Procrustes problem, 332
 - projection, 200
 - projector, 197
 - regression, 329–332
 - orthogonal basis problem, 255
 - orthogonal matrix
 - eigenvalues, 523
 - orthogonality
 - loss of, 253–256
 - orthonormal, 7
 - outer inverse, 201
 - outer product, 5
 - overdetermined, 47
 - packed storage, 79
 - Padé approximant, 568
 - Paige’s method, 317
 - pairwise pivoting, 58
 - partial
 - least squares, 282–287
 - partitioned
 - algorithms, 136–139
 - matrix, 8
 - path, 156
 - PCG, *see* preconditioned CG
 - PCGLS, *see* preconditioned CGLS
 - PCGME, *see* preconditioned CGME
 - Penrose conditions, 200
 - perfect ordering, 157
 - permanent of matrix, 16
 - permutation
 - even, 15
 - matrix, 14
 - odd, 15
 - sign of, 15
 - Perron–Frobenius theorem, 452
 - personnel-assignment problem, 412
 - perturbation
 - componentwise, 220
 - of eigenvalue, 459–468
 - of eigenvector, 459–468
 - of invariant subspaces, 469–474
 - of least squares solution, 219–222
 - of linear systems, 103–111
 - of pseudo-inverse, 202–203
 - perturbation bound
 - for linear system, 105
 - componentwise, 110
 - Peters–Wilkinson method, 227–229
 - Petrov–Galerkin conditions, 620, 696
 - Picard condition, 281
 - pivotal elements, 45
 - pivoting
 - Bunch–Kaufman, 83
 - complete, 55
 - for sparsity, 168–171
 - partial, 55
 - rook, 58
 - planar graph, 167
 - plane rotation, 236–239
 - plane rotations
 - hyperbolic, 320
 - PLS
 - see* partial least squares, 282
 - polar decomposition, 206, 332, 573–575, 586
 - polynomial acceleration, 610
 - polynomial restarting, 700
 - positive semidefinite matrix, 77–78
 - postordered tree, 167
 - postordering, 157
 - Powell’s hybrid method, 367

- power method, 477–484
precision
 double, 32
 single, 32
preconditioned CG, 675
preconditioned CGLS, 677
preconditioned CGME, 677
preconditioning, 673–693
 Schur complement, 687
predicting
 structure of R , 304
primitive
 matrix, 452
principal
 angles, 207
 radius of curvature, 386
 vectors, 207, 443
projection, 197–198
 method, 620–651
 one-dimensional, 622–626
projection method
 oblique, 697
projector
 oblique, 198
 orthogonal, 197
Prony’s method, 391
property A, 604, 686
pseudo-inverse, 199
 Bjerhammar, 213
 derivative, 203
 Kronecker product, 311
 solution, 199, 200
pseudospectra of matrices, 556–589
PSVD algorithm, 520
QMR, *see* Quasi-Minimal Residual method
QMR method, 669
QR algorithm, 491–518
 explicit-shift, 498
 for SVD, 512–518
 Hessenberg matrix, 498–503
 implicit shift, 499
 perfect shifts, 512
 rational, 511
 Rayleigh quotient shift, 499
 symmetric tridiagonal matrix, 508–512
 Wilkinson shift, 510
QR decomposition
 multifrontal, 307–311
QR factorization, 240, 251
 and Cholesky factorization, 240
 backward stability, 243, 249
 column pivoting, 244
 complete, 269–271
 deleting a row, 277–278
 flop count, 243
 generalized, 318
 incomplete, 684
 Kronecker product, 312
 of banded matrix, 292–293
 rank deficient problems, 267–269
 rank one change, 276–277
 rank revealing, 272
 recursive, 296
 row ordering for, 304–306
 row pivoting, 317
 row sequential, 305–307
 row sorting, 317
quadratic inequality constraints, 323–327
quadratic model, 380
quasi-minimal residual, *see* QMR method
Quasi-Minimal Residual method, 669
quasi-Newton
 condition, 381, 389
 method, 371, 381
QZ algorithm, 551
radius of convergence, 560, 561
range, 27
rank, 7
 numerical, 28
 structural, 172
rational Krylov method, 716
Rayleigh quotient, 462, 463
 iteration, 485–487, 551
 matrix, 470, 471, 696, 703
 singular value, 469
Rayleigh–Ritz procedure, 696–698
real Schur form, 447, 482
recursive
 QR factorization, 296
recursive least squares, 274–275
reducible, 10
reducible matrix, 156, 171
reduction to
 Hessenberg form, 495–498
 standard form, 547–549
 symmetric tridiagonal form, 506–508
regression

- orthogonal distance, 394–396
- robust, 345–346
- regression analysis, 249
- regular splitting, 602, 683
- regularization
 - filter factor, 326
 - hybrid method, 658
 - iterated, 656
 - Krylov subspace methods, 657–659
 - Landweber, 655
 - semiconvergence, 656
 - Tikhonov, 323
- relaxation methods, 592
- relaxation parameter, 602, 646
- reorthogonalization, 255
- residual
 - normalized, 217
 - polynomial, 610
 - vector, 462
- residual reducing methods, 645
- resolvent, 557
- resolvent operator, 562
- Riccati equation, 450–451, 474, 587
 - algebraic, 450
 - Newton’s method, 456
- Richardson’s method, 595, 643
- ridge estimate, 324
- right-inverse, 212
- right-looking, 140
- right-preconditioned system, 673
- Ritz values, 696, 699
 - harmonic, 697, 704
- Ritz vectors, 696, 699
- rook pivoting, 58
- row stochastic matrix, 579
- row-action methods, 718
- RQI, *see* Rayleigh quotient iteration
- saddle point, 379
- scaled diagonally dominant, 507
- Schulz iteration, 572
- Schur
 - complement, 11, 319
 - decomposition, 443–448
 - generalized, 546
 - reordering, 446
 - vectors, 444
- Schur–Parlett method, 565, 569
- search direction, 379, 623
- secant equation, 371
- secular equation, 325, 367, 466, 528
- semi-iterative method
 - Chebyshev, 612
- semiconvergence, 655
- seminormal equations, 306
- semiorthogonality, 706
- semiseparable matrix, 99, 181–183
 - generator, 181
- semisimple eigenvalue, 441
- separation
 - of matrices, 473
- Sherman–Morrison formula, 12
- shift of origin, 479
- sign function
 - of matrix, 576–577
- signature matrix, 319
- similarity transformation, 437
- simple bounds, 409
- simplex, 407
 - simplex method, 412–418
 - cycling, 416
 - optimality criterion, 414
 - pricing, 414
 - reduced costs, 414
 - steepest edge strategy, 415
 - tableau, 414
 - textbook strategy, 415
 - simultaneous
 - iteration, 713–714
 - single precision, 32
 - singular value, 25
 - decomposition, 25–28, 204–206
 - singular values
 - by spectrum slicing, 533–534
 - relative gap, 519
 - singular vector, 26
 - skew-symmetric
 - eigenproblem, 521–522
 - matrix, 4, 522, 573
 - factorization, 84
 - sliding window method, 274
 - SOR
 - method, 602–609, 624
 - convergence, 624
 - optimal relaxation parameter, 605
 - SPAI, 685
 - sparse matrix, 592
 - block angular form, 297–300
 - block triangular form, 171–172, 300–302

- spectral
 abscissa, 437, 563
 decomposition, 440, 451
 projector, 451, 461
 radius, 437, 597, 598
 transformation, 482
spectral transformation, 714–716
spectrum
 of matrix, 435
 slicing, 531–533
split preconditioner, 674
splitting, 595
 regular, 602, 683
 standard, 596
square root of matrix, 571–572
SSOR method, 608
standard
 basis, 7
standard form
 of LSQI, 325
 transformation to, 326
stationary iterative method, 595
stationary point, 378
steepest descent method, 624
Steffensen’s method, 369
step length, 379
Stiefel manifold, 475
Stieltjes matrix, 601
storage scheme
 compressed form, 153
 dynamic, 155
 static, 155
Strassen’s algorithm, 142
structural cancellation, 304
structure
 of Cholesky factor, 304
submatrix, 8
 principal, 8
subset selection, 271–273
subspace
 invariant, 439
subspaces
 dimension, 6
successive overrelaxation method, *see* SOR
sum convention, 354
superlinear convergence, 633
supernode, 165
SVD, *see* singular value decomposition
 and pseudo-inverse, 198–202
 compact form, 26
 generalized, 555–556
 of Kronecker product, 312
sweep method, 64
Sylvester
 criterion, 73
 determinant identity, 18
 equation, 178, 341, 449–451, 565, 577
 law of inertia, 80, 531, 551
symmetric
 gauge functions, 28
 indefinite matrix, 81–85
 matrix, 70
 pivoting, 77
 symmetric tridiagonal form
 reduction to, 506–508
symmetrizable iterative method, 609
SYMMLQ, 639
symplectic matrix, 554–555
Taylor’s formula, 357, 358
tensor, 313, 354
tensor decomposition, 313
theorem
 Cayley–Hamilton, 443
 implicit Q , 497, 509
Tikhonov
 method iterated, 656
 regularization, 323
TLS, *see* total least squares
Toeplitz matrix, 174, 655
 fast multiplication, 692
 upper triangular, 327
Toeplitz preconditioners, 690–692
Toeplitz systems, 174–177
 circulant preconditioner, 693
 iterative solvers, 690–693
topological ordering, 157
total least squares, 335–344
 by SVD, 336–338
 conditioning, 338
 generalized, 339–342
 mixed, 341
 multidimensional, 341
 scaled, 336
totally positive, 177
totally positive matrix, 181
trace, 23
transformation
 congruence, 80

- similarity, 437
- transportation problem, 411–412
- transpose (of matrix), 4
- transposition, 14
 - matrix, 14, 50
- tree, 156
 - postordered, 167
- Treppeniteration, 488
- triangular
 - factorization, *see* LU factorization
 - matrix, 42
 - systems of equations, 42–44
- tridiagonal matrix, 87, 476
 - periodic, 97
 - symmetric indefinite, 98
- truncated SVD, 281–282
- trust region method, 367, 388–389
- TSVD, *see* truncated SVD
- two-cyclic matrix, 514
- two-sided Jacobi SVD, 540–542
- ULV decomposition, 271
- unbiased estimate, 194
 - of σ^2 , 217
- underdetermined, 47
- underdetermined system, 191
 - minimum norm solution, 248, 260
- unit matrix, 6
- unitary matrix
 - eigenvalues, 522–523
- unreduced matrix, 497
- updating, 274
- Vandermonde
 - matrix, 180
 - system, 180–181
- variable projection algorithm, 203, 392
- variables
 - basic, 413
 - nonbasic, 413
- variance, 193
- variance-covariance matrix, *see* covariance matrix
- vector
 - bi-orthogonal, 665
 - grade of, 627
 - orthogonal, 7
 - orthonormal, 7
 - principal, 443
- vertex