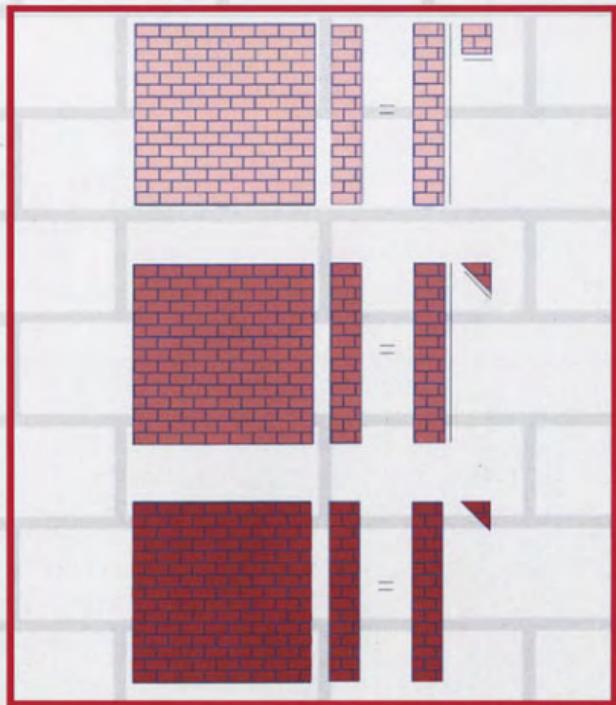


Matrix Algorithms

Volume II: Eigensystems



G. W. Stewart
siam

Matrix Algorithms

This page intentionally left blank

Matrix Algorithms

Volume II: Eigensystems

G. W. Stewart

University of Maryland
College Park, Maryland

siam

Society for Industrial and Applied Mathematics
Philadelphia

Copyright © 2001 by the Society for Industrial and Applied Mathematics.

10 9 8 7 6 5 4 3 2 1

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

MATLAB is a registered trademark of The MathWorks, Inc., 3 Apple Hill Dr., Natick, MA 01760-2098, USA, tel. 508-647-7000, fax 508-647-7001, info@mathworks.com, <http://www.mathworks.com>.

The Library of Congress has catalogued Volume I as follows:

Library of Congress Cataloging-in-Publication Data

Stewart, G. W. (Gilbert W.)

Matrix algorithms / G. W. Stewart.

p. cm.

Includes bibliographical references and index.

Contents: v. 1. Basic decompositions

ISBN 0-89871-414-1 (v. 1 : pbk.)

1. Matrices. I. Title.

QA188.S714 1998

512.9'434--dc21

98-22445

0-89871-414-1 (Volume I)

0-89871-503-2 (Volume II)

0-89871-418-4 (set)

siam is a registered trademark.

CONTENTS

Algorithms	xv
Preface	xvii
1 Eigensystems	1
1 The Algebra of Eigensystems	1
1.1 Eigenvalues and eigenvectors	2
Definitions. Existence of eigenpairs: Special cases. The characteristic equation and the existence of eigenpairs. Matrices of order two. Block triangular matrices. Real matrices.	
1.2 Geometric multiplicity and defective matrices	6
Geometric multiplicity. Defectiveness. Simple eigenvalues.	
1.3 Similarity transformations	8
1.4 Schur Decompositions	10
Unitary similarities. Schur form. Nilpotent matrices. Hermitian matrices. Normal matrices.	
1.5 Block diagonalization and Sylvester's equation	15
From block triangular to block diagonal matrices. Sylvester's equation. An algorithm. Block diagonalization redux.	
1.6 Jordan form	20
1.7 Notes and references	23
General references. History and Nomenclature. Schur form. Sylvester's equation. Block diagonalization. Jordan canonical form.	
2 Norms, Spectral Radii, and Matrix Powers	25
2.1 Matrix and vector norms	25
Definition. Examples of norms. Operator norms. Norms and rounding error. Limitations of norms. Consistency. Norms and convergence.	
2.2 The spectral radius	30
Definition. Norms and the spectral radius.	
2.3 Matrix powers	33
Asymptotic bounds. Limitations of the bounds.	

2.4 Notes and references	36
Norms. Norms and the spectral radius. Matrix powers.	
3 Perturbation Theory	37
3.1 The perturbation of eigenvalues	37
The continuity of eigenvalues. Gershgorin theory. Hermitian matrices.	
3.2 Simple eigenpairs	44
Left and right eigenvectors. First-order perturbation expansions.	
Rigorous bounds. Qualitative statements. The condition of an eigenvalue. The condition of an eigenvector. Properties of sep. Hermitian matrices.	
3.3 Notes and references	52
General references. Continuity of eigenvalues. Gershgorin theory. Eigenvalues of Hermitian matrices. Simple eigenpairs.	
Left and right eigenvectors. First-order perturbation expansions.	
Rigorous bounds. Condition numbers. Sep. Relative and structured perturbation theory.	
2 The QR Algorithm	55
1 The Power and Inverse Power Methods	56
1.1 The power method	56
Convergence. Computation and normalization. Choice of a starting vector. Convergence criteria and the residual. Optimal residuals and the Rayleigh quotient. Shifts and spectral enhancement. Implementation. The effects of rounding error. Assessment of the power method.	
1.2 The inverse power method	66
Shift-and-invert enhancement. The inverse power method. An example. The effects of rounding error. The Rayleigh quotient method.	
1.3 Notes and references	69
The power method. Residuals and backward error. The Rayleigh quotient. Shift of origin. The inverse power and Rayleigh quotient methods.	
2 The Explicitly Shifted QR Algorithm	71
2.1 The QR algorithm and the inverse power method	72
Schur from the bottom up. The QR choice. Error bounds. Rates of convergence. General comments on convergence.	
2.2 The unshifted QR algorithm	75
The QR decomposition of a matrix polynomial. The QR algorithm and the power method. Convergence of the unshifted QR algorithm.	
2.3 Hessenberg form	80

Householder transformations. Reduction to Hessenberg form.	
Plane rotations. Invariance of Hessenberg form.	
2.4 The explicitly shifted algorithm	94
Detecting negligible subdiagonals. Deflation. Back searching.	
The Wilkinson shift. Reduction to Schur form. Eigenvectors.	
2.5 Bits and pieces	104
Ad hoc shifts. Aggressive deflation. Cheap eigenvalues.	
Balancing. Graded matrices.	
2.6 Notes and references	110
Historical. Variants of the QR algorithm. The QR algorithm, the	
inverse power method, and the power method. Local	
convergence. Householder transformations and plane rotations.	
Hessenberg form. The Hessenberg QR algorithm. Deflation	
criteria. The Wilkinson shift. Aggressive deflation.	
Preprocessing. Graded matrices.	
3 The Implicitly Shifted QR Algorithm	113
3.1 Real Schur form	113
Real Schur form. A preliminary algorithm.	
3.2 The uniqueness of Hessenberg reduction	116
3.3 The implicit double shift	117
A general strategy. Getting started. Reduction back to Hessenberg	
form. Deflation. Computing the real Schur form. Eigenvectors.	
3.4 Notes and references	128
Implicit shifting. Processing 2×2 blocks. The multishifted QR	
algorithm.	
4 The Generalized Eigenvalue Problem	129
4.1 The theoretical background	130
Definitions. Regular pencils and infinite eigenvalues.	
Equivalence transformations. Generalized Schur form.	
Multiplicity of eigenvalues. Projective representation of	
eigenvalues. Generalized shifting. Left and right eigenvectors of	
a simple eigenvalue. Perturbation theory. First-order expansions:	
Eigenvalues. The chordal metric. The condition of an eigenvalue.	
Perturbation expansions: Eigenvectors. The condition of an	
eigenvector.	
4.2 Real Schur and Hessenberg-triangular forms	143
Generalized real Schur form. Hessenberg-triangular form.	
4.3 The doubly shifted QZ algorithm	147
Overview. Getting started. The QZ step.	
4.4 Important miscellanea	152
Balancing. Back searching. Infinite eigenvalues. The 2×2	
problem. Eigenvectors.	

4.5 Notes and References	155
The algebraic background. Perturbation theory. The QZ algorithm.	
3 The Symmetric Eigenvalue Problem	157
1 The QR Algorithm	158
1.1 Reduction to tridiagonal form	158
The storage of symmetric matrices. Reduction to tridiagonal form. Making Hermitian tridiagonal matrices real.	
1.2 The symmetric tridiagonal QR algorithm	163
The implicitly shifted QR step. Choice of shift. Local convergence. Deflation. Graded matrices. Variations.	
1.3 Notes and references	169
General references. Reduction to tridiagonal form. The tridiagonal QR algorithm.	
2 A Clutch of Algorithms	171
2.1 Rank-one updating	171
Reduction to standard form. Deflation: Small components of z . Deflation: Nearly equal d_i . When to deflate. The secular equation. Solving the secular equation. Computing eigenvectors. Concluding comments.	
2.2 A Divide-and-conquer algorithm	181
Generalities. Derivation of the algorithm. Complexity.	
2.3 Eigenvalues and eigenvectors of band matrices	186
The storage of band matrices. Tridiagonalization of a symmetric band matrix. Inertia. Inertia and the LU decomposition. The inertia of a tridiagonal matrix. Calculation of a selected eigenvalue. Selected eigenvectors.	
2.4 Notes and References	201
Updating the spectral decomposition. The divide-and-conquer algorithm. Reduction to tridiagonal form. Inertia and bisection. Eigenvectors by the inverse power method. Jacobi's method.	
3 The Singular Value Decomposition	203
3.1 Background	203
Existence and nomenclature. Uniqueness. Relation to the spectral decomposition. Characterization and perturbation of singular values. First-order perturbation expansions. The condition of singular vectors.	
3.2 The cross-product algorithm	210
Inaccurate singular values. Inaccurate right singular vectors. Inaccurate left singular vectors. Assessment of the algorithm.	
3.3 Reduction to bidiagonal form	215
The reduction. Real bidiagonal form.	

3.4	The QR algorithm	217
	The bidiagonal QR step. Computing the shift. Negligible superdiagonal elements. Back searching. Final comments.	
3.5	A hybrid QRD-SVD algorithm	226
3.6	Notes and references	226
	The singular value decomposition. Perturbation theory. The cross-product algorithm. Reduction to bidiagonal form and the QR algorithm. The QRD-SVD algorithm. The differential qd algorithm. Divide-and-conquer algorithms. Downdating a singular value decomposition. Jacobi methods.	
4	The Symmetric Positive Definite (S/PD) Generalized Eigenvalue Problem	229
4.1	Theory	229
	The fundamental theorem. Condition of eigenvalues.	
4.2	Wilkinson's algorithm	231
	The algorithm. Computation of $R^{-T} A R^{-1}$. Ill-conditioned B .	
4.3	Notes and references	234
	Perturbation theory. Wilkinson's algorithm. Band matrices. The definite generalized eigenvalue problem. The generalized singular value and CS decompositions.	
4	Eigenspaces and Their Approximation	239
1	Eigenspaces	240
1.1	Definitions	240
	Eigenbases. Existence of eigenspaces. Deflation.	
1.2	Simple eigenspaces	244
	Definition. Block diagonalization and spectral representations.	
	Uniqueness of simple eigenspaces.	
1.3	Notes and references	247
	Eigenspaces. Spectral projections. Uniqueness of simple eigenspaces. The resolvent.	
2	Perturbation Theory	248
2.1	Canonical angles	248
	Definitions. Computing canonical angles. Subspaces of unequal dimensions. Combinations of subspaces.	
2.2	Residual analysis	251
	Optimal residuals. The Rayleigh quotient. Backward error.	
	Residual bounds for eigenvalues of Hermitian matrices. Block deflation. The quantity sep . Residual bounds.	
2.3	Perturbation theory	258
	The perturbation theorem. Choice of subspaces. Bounds in terms of E . The Rayleigh quotient and the condition of L . Angles between the subspaces.	
2.4	Residual bounds for Hermitian matrices	262

Residual bounds for eigenspaces. Residual bounds for eigenvalues.	
2.5 Notes and references	264
General references. Canonical angle. Optimal residuals and the Rayleigh quotient. Backward error. Residual eigenvalue bounds for Hermitian matrices. Block deflation and residual bounds.	
Perturbation theory. Residual bounds for eigenspaces and eigenvalues of Hermitian matrices.	
3 Krylov Subspaces	266
3.1 Krylov sequences and Krylov spaces	266
Introduction and definition. Elementary properties. The polynomial connection. Termination.	
3.2 Convergence	269
The general approach. Chebyshev polynomials. Convergence redux. Multiple eigenvalues. Assessment of the bounds. Non-Hermitian matrices.	
3.3 Block Krylov sequences	277
3.4 Notes and references	279
Sources. Krylov sequences. Convergence (Hermitian matrices). Convergence (non-Hermitian matrices). Block Krylov sequences.	
4 Rayleigh–Ritz Approximation	280
4.1 Rayleigh–Ritz methods	281
Two difficulties. Rayleigh–Ritz and generalized eigenvalue problems. The orthogonal Rayleigh–Ritz procedure.	
4.2 Convergence	284
Setting the scene. Convergence of the Ritz value. Convergence of Ritz vectors. Convergence of Ritz values revisited. Hermitian matrices. Convergence of Ritz blocks and bases. Residuals and convergence.	
4.3 Refined Ritz vectors	289
Definition. Convergence of refined Ritz vectors. The computation of refined Ritz vectors.	
4.4 Harmonic Ritz vectors	292
Computation of harmonic Ritz pairs.	
4.5 Notes and references	295
Historical. Convergence. Refined Ritz vectors. Harmonic Ritz vectors.	
5 Krylov Sequence Methods	297
1 Krylov Decompositions	297
1.1 Arnoldi decompositions	298
The Arnoldi decomposition. Uniqueness. The Rayleigh quotient. Reduced Arnoldi decompositions. Computing Arnoldi	

decompositions. Reorthogonalization. Practical considerations.	
Orthogonalization and shift-and-invert enhancement.	
1.2 Lanczos decompositions	306
Lanczos decompositions. The Lanczos recurrence.	
1.3 Krylov decompositions	308
Definition. Similarity transformations. Translation. Equivalence to an Arnoldi decomposition. Reduction to Arnoldi form.	
1.4 Computation of refined and Harmonic Ritz vectors	314
Refined Ritz vectors. Harmonic Ritz vectors.	
1.5 Notes and references	315
Lanczos and Arnoldi. Shift-and-invert enhancement and orthogonalization. Krylov decompositions.	
2 The Restarted Arnoldi Method	316
2.1 The implicitly restarted Arnoldi method	317
Filter polynomials. Implicitly restarted Arnoldi. Implementation. The restarted Arnoldi cycle. Exact shifts. Forward instability.	
2.2 Krylov–Schur restarting	325
Exchanging eigenvalues and eigenblocks. The Krylov–Schur cycle. The equivalence of Arnoldi and Krylov–Schur. Comparison of the methods.	
2.3 Stability, convergence, and deflation	332
Stability. Stability and shift-and-invert enhancement. Convergence criteria. Convergence with shift-and-invert enhancement. Deflation. Stability of deflation. Nonorthogonal bases. Deflation in the Krylov–Schur method. Deflation in an Arnoldi decomposition. Choice of tolerance.	
2.4 Rational Krylov Transformations	342
Krylov sequences, inverses, and shifts. The rational Krylov method.	
2.5 Notes and references	344
Arnoldi’s method, restarting, and filter polynomials. ARPACK. Reverse communication. Exchanging eigenvalues. Forward instability. The Krylov–Schur method. Stability. Shift-and-invert enhancement and the Rayleigh quotient. Convergence. Deflation. The rational Krylov transformation.	
3 The Lanczos Algorithm	347
3.1 Reorthogonalization and the Lanczos algorithm	348
Lanczos with reorthogonalization.	
3.2 Full reorthogonalization and restarting	351
Implicit restarting. Krylov-spectral restarting. Convergence and deflation.	
3.3 Semiorthogonal methods	352

Semiorthogonal bases. The QR factorization of a semiorthogonal basis. The size of the reorthogonalization coefficients. The effects on the Rayleigh quotient. The effects on the Ritz vectors.	
Estimating the loss of orthogonality. Periodic reorthogonalization.	
Updating H_k . Computing residual norms. An example.	
3.4 Notes and references	365
The Lanczos algorithm. Full reorthogonalization. Filter polynomials. Semiorthogonal methods. No reorthogonalization.	
The singular value decomposition. Block Lanczos algorithms.	
The bi-Lanczos algorithm.	
4 The Generalized Eigenvalue Problem	367
4.1 Transformation and convergence	368
Transformation to an ordinary eigenproblem. Residuals and backward error. Bounding the residual.	
4.2 Positive definite B	370
The B inner product. Orthogonality. B -Orthogonalization. The B -Arnoldi method. The restarted B -Lanczos method. The B -Lanczos method with periodic reorthogonalization.	
Semidefinite B .	
4.3 Notes and references	379
The generalized shift-and-invert transformation and the Lanczos algorithm. Residuals and backward error. Semidefinite B .	
6 Alternatives	381
1 Subspace Iteration	381
1.1 Theory	382
Convergence. The QR connection. Schur–Rayleigh–Ritz refinement.	
1.2 Implementation	386
A general algorithm. Convergence. Deflation. When to perform an SRR step. When to orthogonalize. Real matrices.	
1.3 Symmetric matrices	391
Economization of storage. Chebyshev acceleration.	
1.4 Notes and references	394
Subspace iteration. Chebyshev acceleration. Software.	
2 Newton-Based Methods	395
2.1 Approximate Newton methods	396
A general approximate Newton method. The correction equation.	
Orthogonal corrections. Analysis. Local convergence. Three approximations.	
2.2 The Jacobi–Davidson method	404
The basic Jacobi–Davidson method. Extending Schur decompositions. Restarting. Harmonic Ritz vectors. Hermitian	

matrices. Solving projected systems. The correction equation: Exact solution. The correction equation: Iterative solution.	
2.3 Notes and references	418
Newton's method and the eigenvalue problem. Inexact Newton methods. The Jacobi–Davidson method. Hermitian matrices.	
Appendix: Background	421
1 Scalars, vectors, and matrices	421
2 Linear algebra	424
3 Positive definite matrices	425
4 The LU decomposition	425
5 The Cholesky decomposition	426
6 The QR decomposition	426
7 Projectors	427
References	429
Index	451

This page intentionally left blank

ALGORITHMS

Chapter 1. Eigensystems	
1.1 Solution of Sylvester's Equation	18
Chapter 2. The QR Algorithm	
1.1 The shifted power method	64
1.2 The inverse power method	67
2.1 Generation of Householder transformations	83
2.2 Reduction to upper Hessenberg form	85
2.3 Generation of a plane rotation	90
2.4 Application of a plane rotation	90
2.5 Basic QR step for a Hessenberg matrix	93
2.6 Finding deflation rows in an upper Hessenberg matrix	96
2.7 Computation of the Wilkinson shift	98
2.8 Schur form of an upper Hessenberg matrix	99
2.9 Right eigenvectors of an upper triangular matrix	103
3.1 Starting an implicit QR double shift	120
3.2 Doubly shifted QR step	122
3.3 Finding deflation rows in a real upper Hessenberg matrix	124
3.4 Real Schur form of a real upper Hessenberg matrix	125
4.1 Reduction to Hessenberg-triangular form	146
4.2 The doubly shifted QZ step	151
Chapter 3. The Symmetric Eigenvalue Problem	
1.1 Reduction to tridiagonal form	161
1.2 Hermitian tridiagonal to real tridiagonal form	164
1.3 The symmetric tridiagonal QR step	166
2.1 Rank-one update of the spectral decomposition	172
2.2 Eigenvectors of a rank-one update	180
2.3 Divide-and-conquer eigensystem of a symmetric tridiagonal matrix . .	184
2.4 Counting left eigenvalues	192
2.5 Finding a specified eigenvalue	194
2.6 Computation of selected eigenvectors	199
3.1 The cross-product algorithm	211
3.2 Reduction to bidiagonal form	216

3.3	Complex bidiagonal to real bidiagonal form	218
3.4	Bidiagonal QR step	221
3.5	Back searching a bidiagonal matrix	225
3.6	Hybrid QR-SVD algorithm	226
4.1	Solution of the S/PD generalized eigenvalue problem	232
4.2	The computation of $C = R^{-T} A R^{-1}$	234
Chapter 5. Krylov Sequence Methods		
1.1	The Arnoldi step	304
1.2	The Lanczos recurrence	308
1.3	Krylov decomposition to Arnoldi decomposition	313
2.1	Filtering an Arnoldi decomposition	321
2.2	The restarted Arnoldi cycle	322
2.3	The Krylov–Schur cycle	330
2.4	Deflating in a Krylov–Schur decomposition	341
2.5	The rational Krylov transformation	344
3.1	Lanczos procedure with orthogonalization	349
3.2	Estimation of $u_{k+1}^T u_j$	357
3.3	Periodic orthogonalization I: Outer loop	360
3.4	Periodic reorthogonalization II: Orthogonalization step	361
Chapter 6. Alternatives		
1.1	Subspace iteration	387
2.1	The basic Jacobi–Davidson algorithm	405
2.2	Extending a partial Schur decomposition	410
2.3	Harmonic extension of a partial Schur decomposition	413
2.4	Solution of projected equations	415
2.5	Davidson’s method	419

PREFACE

This book, *Eigensystems*, is the second volume in a projected five-volume series entitled *Matrix Algorithms*. The first volume treated basic decompositions. The three following this volume will treat iterative methods for linear systems, sparse direct methods, and special topics, including fast algorithms for structured matrices.

My intended audience is the nonspecialist whose needs cannot be satisfied by black boxes. It seems to me that these people will be chiefly interested in the methods themselves — how they are derived and how they can be adapted to particular problems. Consequently, the focus of the series is on algorithms, with such topics as rounding-error analysis and perturbation theory introduced impromptu as needed. My aim is to bring the reader to the point where he or she can go to the research literature to augment what is in the series.

The series is self-contained. The reader is assumed to have a knowledge of elementary analysis and linear algebra and a reasonable amount of programming experience — about what you would expect from a beginning graduate engineer or an advanced undergraduate in an honors program. Although strictly speaking the individual volumes are not textbooks, they are intended to teach, and my guiding principle has been that if something is worth explaining it is worth explaining fully. This has necessarily restricted the scope of the series, but I hope the selection of topics will give the reader a sound basis for further study.

The subject of this volume is computations involving the eigenvalues and eigenvectors of a matrix. The first chapter is devoted to an exposition of the underlying mathematical theory. The second chapter treats the QR algorithm, which in its various manifestations has become the universal workhorse in this field. The third chapter deals with symmetric matrices and the singular value decomposition. The second and third chapters also treat the generalized eigenvalue problem.

Up to this point the present volume shares the concern of the first volume with dense matrices and their decompositions. In the fourth chapter the focus shifts to large matrices for which the computation of a complete eigensystem is not possible. The general approach to such problems is to calculate approximations to subspaces corresponding to groups of eigenvalues — eigenspaces or invariant subspaces they are called. Accordingly, in the fourth chapter we will present the algebraic and analytic theory of eigenspaces. We then return to algorithms in the fifth chapter, which treats Krylov sequence methods — in particular the Arnoldi and the Lanczos methods. In

the last chapter we consider alternative methods such as subspace iteration and the Jacobi–Davidson method.

With this second volume, I have had to come to grips with how the volumes of this work hang together. Initially, I conceived the series being a continuous exposition of the art of matrix computations, with heavy cross referencing between the volumes. On reflection, I have come to feel that the reader will be better served by semi-independent volumes. However, it is impossible to redevelop the necessary theoretical and computational material in each volume. I have therefore been forced to assume that the reader is familiar, in a general way, with the material in previous volumes. To make life easier, much of the necessary material is slipped unobtrusively into the present volume and more background can be found in the appendix. However, I have not hesitated to reference the first volume — especially its algorithms — where I thought it appropriate.

Many methods treated in this volume are summarized by displays of pseudocode (see the list of algorithms following the table of contents). These summaries are for purposes of illustration and should not be regarded as finished implementations. In the first place, they often leave out error checks that would clutter the presentation. Moreover, it is difficult to verify the correctness of algorithms written in pseudocode. Wherever possible, I have checked the algorithms against MATLAB implementations. However, such a procedure is not proof against transcription errors. Be on guard!

A word on organization. The book is divided into numbered chapters, sections, and subsections, followed by unnumbered subsubsections. Numbering is by section, so that (3.5) refers to the fifth equations in section three of the current chapter. References to items outside the current chapter are made explicitly — e.g., Theorem 2.7, Chapter 1. References to Volume I of this series [265] are preceded by the Roman numeral I — e.g., Algorithm I:3.2.1.

While I was writing this volume, I posted it on the web and benefited greatly from the many comments and suggestions I received. I should like to thank K. M. Briggs, Austin Dubrulle, David Dureisseix, Mei Kobayashi, Daniel Kressner, Andrzej Mackiewicz, Nicola Mastronardi, Arnold Neumaier, John Pryce, Thomas Strohmer, Henk van der Vorst, and David Watkins for their comments and suggestions. I am indebted to the Mathematical and Computational Sciences Division of NIST for the use of their research facilities. Special thanks go to the editors of *Templates for the Solution of Algebraic Eigenvalue Problems*, who allowed me access to their excellent compendium while it was in preparation.

SIAM publications and its people, past and present, have aided the writing and production of this book. Vickie Kearn first encouraged me to start the project and has helped it along too many ways to list. Kelly Thomas and Sam Young have seen the present volume through production. Special mention should be made of my long-time copy editor and email friend, Jean Anderson. As usual, she has turned an onerous job into a pleasant collaboration.

In 1997, as I was preparing to write this volume, Richard Lehoucq invited me to a

workshop on large-scale eigenvalue problems held at Argonne National Laboratory. I was stunned by the progress made in the area since the late sixties and early seventies, when I had worked on the problem. The details of these new and improved algorithms are the subject of the second half of this volume. But any unified exposition makes it easy to forget that the foundations for these advances had been laid by people who were willing to work in an unexplored field, where confusion and uncertainty were the norm. Naming names carries the risk of omitting someone, but any list would surely include Vel Kahan, Shmuel Kaniel, Chris Paige, Beresford Parlett, and Yousef Saad. This book is dedicated to these pioneers and their coworkers.

G. W. Stewart
College Park, MD

This page intentionally left blank

1

EIGENSYSTEMS

In 1965 James Hardy Wilkinson published his classic work on numerical linear algebra, *The Algebraic Eigenvalue Problem*. His purpose was to survey the computational state of the art, but a significant part of the book was devoted to the mathematical foundations, some of which had been developed by Wilkinson himself. This happy mixture of the theoretical and computational has continued to characterize the area — computational advances go hand in hand with deeper mathematical understanding.

The purpose of this chapter is to provide the mathematical background for the first half of this volume. The first section consists of an exposition of the classic algebraic theory of eigensystems. The predominant theme of this section is the systematic reduction of matrices to simpler forms by similarity transformations. One of these forms — the Schur decomposition — is the goal of the QR algorithm, the workhorse of dense eigenvalue computations.

The second and third sections are devoted to the analysis of eigensystems. The second section concerns norms and their relation to the spectrum of a matrix. Here we will pay special attention to the asymptotic behavior of powers of matrices, which are important in the analysis of many matrix algorithms. The third section is devoted to perturbation theory — how eigensystems behave when the original matrix is perturbed. This is a large subject, and we will only be able to survey the basics.

The majority of eigenvalue problems involve real matrices, and their reality often results in computational savings. However, the theory of eigensystems is best treated in terms of complex matrices. Consequently, throughout this chapter:

Unless otherwise stated A is a complex matrix of order n .

1. THE ALGEBRA OF EIGENSYSTEMS

In this section we will develop the classical theory of eigenvalues, eigenvectors, and reduction by similarity transformations, with an emphasis on the reduction of a matrix by similarity transformations to a simpler form. We begin with basic definitions and their consequences. We then present Schur's reduction of a triangular matrix to triangular form by unitary similarity transformations — a decomposition that serves as

a springboard for computing most others. Further reduction to block diagonal form is treated in §1.5. The reduction depends on our ability to solve a matrix equation called Sylvester's equation, which we investigate in some detail. Finally in §1.6 we present the Jordan canonical form.

1.1. EIGENVALUES AND EIGENVECTORS

Definitions

This volume has two heroes: the eigenvalue and its companion the eigenvector. Since the two are inseparable, we will formalize their pairing in the following definition.

Definition 1.1. Let A be of order n . The pair (λ, x) is called an **EIGENPAIR** or a **RIGHT EIGENPAIR** of A if

1. $x \neq 0$,
2. $Ax = \lambda x$.

The scalar λ is called an **EIGENVALUE** of A and the vector x is called an **EIGENVECTOR**. The pair (λ, y) is called a **LEFT EIGENPAIR** if y is nonzero and $y^H A = \lambda y^H$. The multiset of eigenvalues of A is called the **SPECTRUM** of A and is denoted by $\Lambda(A)$.

Here are some comments on this definition.

- The unqualified terms “eigenpair” and “eigenvector” refer to right eigenpairs and right eigenvectors. Left eigenpairs and eigenvectors will always be qualified.
- If (λ, x) is an eigenpair and $\tau \neq 0$, then $(\lambda, \tau x)$ is also an eigenpair. To remove this trivial nonuniqueness, one often specifies a *normalization* of x . For example, one might require that $x^H x = 1$ or that some component of x be equal to one. In any event, when the eigenvector corresponding to an eigenvalue is unique up to a scalar multiple, we will refer to the one we are currently interested in as *the* eigenvector.
- Eigenvalues can be repeated. For example, it is natural to regard the identity matrix of order n as having n eigenvalues, all equal to one (see Theorem 1.2). To keep track of multiplicities we define the spectrum of a matrix to be a *multiset*, i.e., a set in which a member may be repeated.

Perhaps the best way to think of an eigenvector x of a matrix A is that it represents a direction which remains invariant under multiplication by A . The corresponding eigenvalue λ is then the representation of A in the subspace spanned by the eigenvector x . The advantage of having this representation is that multiplication by A is reduced to a scalar operation along the eigenvector. For example, it is easy to show that if $Ax = \lambda x$ then

$$A^k x = \lambda^k x. \tag{1.1}$$

Thus, the effect of a power of A along x can be determined by taking the corresponding power of λ .

Existence of eigenpairs: Special cases

In a moment we will show that any matrix has an eigenpair. However, before we formally establish the existence of eigenpairs, it will be convenient to examine three important cases in which a matrix must by its nature have eigenpairs.

- **Singular matrices.** If A is singular, then A has a null vector, i.e., a vector $x \neq 0$ for which $Ax = 0$. It follows that $(0, x)$ is an eigenpair of A .
- **Diagonal matrices.** If A is diagonal and \mathbf{e}_i denotes the i th unit vector (i.e., the vector whose i th component is one and whose other components are zero), then

$$A\mathbf{e}_i = \alpha_{ii}\mathbf{e}_i.$$

Thus $(\alpha_{ii}, \mathbf{e}_i)$ is an eigenpair of A .

- **Triangular matrices.** Let A be a triangular matrix partitioned in the form

$$A = \begin{pmatrix} A_{11} & a_{1k} & A_{1,k+1} \\ 0 & \alpha_{kk} & a_{k,k+1}^H \\ 0 & 0 & A_{k+1,k+1} \end{pmatrix},$$

where A_{11} is of order k . If no diagonal element of A_{11} is equal to α_{kk} , then $\alpha_{kk}I - A_{11}$ is nonsingular, and it is easily verified that

$$\begin{pmatrix} (\alpha_{kk}I - A_{11})^{-1}a_{1k} \\ 1 \\ 0 \end{pmatrix}$$

is an eigenvector of A corresponding to the eigenvalue α_{kk} . Since any diagonal element of A must have a first appearance on the diagonal, it follows that the diagonal elements of A are eigenvalues of A . Conversely, if (λ, x) is an eigenvector of A it must have a last nonzero component, say ξ_k . It then follows from the equation

$$\begin{pmatrix} A_{11} & a_{1k} & A_{1,k+1} \\ 0 & \alpha_{kk} & a_{k,k+1} \\ 0 & 0 & A_{k+1,k+1} \end{pmatrix} \begin{pmatrix} x_1 \\ \xi_k \\ 0 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ \xi_k \\ 0 \end{pmatrix} \quad (1.2)$$

that $\alpha_{kk}\xi_k = \lambda\xi_k$ or $\lambda = \alpha_{kk}$. Thus the eigenvalues of a triangular matrix are precisely its diagonal elements.

The characteristic equation and the existence of eigenpairs

We now turn to the existence of eigenpairs. If (λ, x) is an eigenpair of A , then $\lambda x - Ax = 0$, or equivalently

$$(\lambda I - A)x = 0.$$

Since x is nonzero, $\lambda I - A$ must be singular. Conversely, if $\lambda I - A$ is singular, then it has a null vector x , and (λ, x) is an eigenpair of A . Thus the eigenvalues of A are

the scalars for which $\lambda I - A$ is singular. Since a matrix is singular if and only if its determinant is zero, the eigenvalues of A are the scalars for which

$$p_A(\lambda) \equiv \det(\lambda I - A) = 0. \quad (1.3)$$

The function $p_A(\lambda)$ is a monic polynomial of degree n called the *characteristic polynomial* of A . We have thus shown that the eigenvalues of A are the roots of the *characteristic equation* (1.3). Since a polynomial of degree n has exactly n roots, a matrix of order n has n eigenvalues. More precisely, from the fundamental theorem of algebra we have the following theorem.

Theorem 1.2. *Let A be of order n , and let $p_A(\lambda) = \det(\lambda I - A)$ be its characteristic polynomial. Then $p_A(\lambda)$ can be factored uniquely in the form*

$$p_A(\lambda) = (\lambda - \lambda_1)^{m_1}(\lambda - \lambda_2)^{m_2} \cdots (\lambda - \lambda_k)^{m_k},$$

where the numbers λ_i are distinct and

$$m_1 + m_2 + \cdots + m_k = n.$$

The eigenvalues of A are the scalars λ_i , which are said to have ALGEBRAIC MULTIPLICITY m_i . Thus, counting algebraic multiplicities every matrix of order n has n eigenvalues.

This theorem and the accompanying definition has two consequences.

- It is important to keep track of the algebraic multiplicity of the eigenvalues. To do so, we have already defined $\Lambda(A)$ to be a multiset. We will also use the phrase “counting multiplicities” as a reminder that multiplicities count.
- It follows from the theorem that we do not need to distinguish between “right” and “left” eigenvalues. For if $\lambda I - A$ is singular, then, in addition to a right null vector, it has a left null vector y^H . Equivalently, $y^H A = \lambda y^H$; i.e., y^H is a left eigenvector corresponding to λ . Thus every eigenvalue has both a left and right eigenvector.

Matrices of order two

In many applications it is useful to have the characteristic polynomial of a 2×2 matrix, say,

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

For this matrix

$$p_A(\lambda) = \lambda^2 - (a + d)\lambda + (ad - bc).$$

An easily memorized equivalent is

$$p_A(\lambda) = \lambda^2 - \text{trace}(A)\lambda + \det(A).$$

In fact for any square matrix A of order n ,

$$p_A(\lambda) = \lambda^n - \text{trace}(A)\lambda^{n-1} + \cdots + (-1)^n \det(A)$$

(see the proof of Theorem 1.11).

Block triangular matrices

Another important consequence of the characterization (1.3) of the eigenvalues of a matrix concerns block triangular matrices. A matrix A is *block upper triangular* if it can be partitioned in the form

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ 0 & A_{22} & \cdots & A_{2k} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & A_{kk} \end{pmatrix}, \quad (1.4)$$

where the diagonal blocks in the partitions are square. A block lower triangular matrix is defined analogously.

Now if A is block upper triangular with partition (1.4), then it is easily verified that

$$\det(\lambda I - A) = \det(\lambda I - A_{11}) \det(\lambda I - A_{22}) \cdots \det(\lambda I - A_{kk})$$

or

$$p_A(\lambda) = p_{A_{11}}(\lambda) p_{A_{22}}(\lambda) \cdots p_{A_{kk}}(\lambda).$$

Since the zeros of a product of polynomials, counting multiplicities, are the union of the zeros of the individual polynomials, counting multiplicities, we see that:

The eigenvalues of a block triangular matrix, counting multiplicities, are the union of eigenvalues of its diagonal blocks, counting multiplicities. (1.5)

In particular, the eigenvalues of a triangular matrix are the diagonal elements of the matrix — a fact we have already established.

Real matrices

It is easy to construct real matrices that have complex eigenvalues. For example, the matrix

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

whose characteristic polynomial is $\lambda^2 + 1$, has eigenvalues $\pm i$. Because complex arithmetic is more expensive than real arithmetic, the existence of real matrices with complex eigenvalues poses computational problems. Fortunately, the complex eigenpairs

of a real matrix have a special structure that we can exploit to eliminate the use of complex arithmetic.

Specifically, if A is real, then its characteristic polynomial is real. Now the complex zeros of a real polynomial occur in conjugate pairs. It follows that the complex eigenvalues of a real matrix occur in conjugate pairs, and such pairs have the same algebraic multiplicity. Actually, we can say more. For if $Ax = \lambda x$, where λ is complex, then on taking conjugates we find that $A\bar{x} = \bar{\lambda}\bar{x}$. It follows that \bar{x} is an eigenvector of A corresponding to $\bar{\lambda}$. Thus, if we have computed a complex eigenpair, we have effectively computed the conjugate eigenpair.

Moreover, the real and imaginary parts of the eigenvector x corresponding to a complex eigenvalue λ are linearly independent. To see this write $x = y + iz$ and assume that, say, y is nonzero. Then for y and z to be linearly dependent, we must have $z = \tau y$ for some real scalar τ . It follows that the vector

$$\hat{x} \equiv (1 - i\tau)x = (1 - i\tau)(y + iz) = (1 + \tau^2)y$$

is real and nonzero. Hence on multiplying the relation $Ax = \lambda x$ by $1 - i\tau$, we get $A\hat{x} = \lambda\hat{x}$. Since A and \hat{x} are real and $\hat{x} \neq 0$, λ is also real—a contradiction.

We may summarize these results in the following theorem.

Theorem 1.3. *The complex eigenvalues of a real matrix A occur in conjugate pairs of equal algebraic multiplicity. If (λ, x) is a complex eigenpair of A , then so is $(\bar{\lambda}, \bar{x})$. Moreover, the real and imaginary parts of x are linearly independent.*

1.2. GEOMETRIC MULTIPLICITY AND DEFECTIVE MATRICES

Geometric multiplicity

The fact that we refer to the multiplicity of an eigenvalue as a root of the characteristic equation as its *algebraic* multiplicity suggests that there is another kind of multiplicity. The following theorem sets the stage for defining it. Its proof is left as an exercise.

Theorem 1.4. *Let λ be an eigenvalue of A . Then the set of all eigenvectors corresponding to λ together with the zero vector is the null space of $\lambda I - A$.*

The theorem shows that the set of eigenvectors corresponding to an eigenvalue along with the zero vector forms a subspace. The dimension of that subspace is the maximal number of linearly independent eigenvectors associated with the eigenvalue. This suggests the following definition.

Definition 1.5. *The GEOMETRIC MULTIPLICITY of an eigenvalue λ is the dimension of the subspace spanned by its eigenvectors or equivalently $\text{null}(\lambda I - A)$.*

It is natural to conjecture that the algebraic and geometric multiplicities of an eigenvalue must be the same. Unfortunately, that need not be true. The geometric multiplicity is never greater than the algebraic multiplicity (see the notes following Theorem 1.12), but it can be less, as the following example shows.

Example 1.6. Let

$$J_2(0) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

The characteristic equation of $J_2(0)$ is $\lambda^2 = 0$, from which it follows that 0 is an eigenvalue of algebraic multiplicity two. On the other hand, the null space of $J_2(0)$ is spanned by e_1 . Hence the geometric multiplicity of the eigenvalue 0 is one.

Defectiveness

The matrix $J_2(0)$ is an example of a *Jordan block*. It is also an example of a defective matrix. Specifically, we say that an eigenvalue whose algebraic multiplicity is less than its geometric multiplicity is *defective*. A matrix with one or more defective eigenvalues is also said to be defective.

Defectiveness introduces two practical difficulties. First, a defective matrix has too few eigenvectors. To see why this is a problem, let us agree to say that the matrix A has a *complete system of eigenvectors* if it has n linearly independent eigenvectors x_i corresponding to eigenvalues λ_i . (We also say that the eigenpairs (λ_i, x_i) form a complete system.) Then any vector y can be expressed in the form

$$y = \gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_n x_n. \quad (1.6)$$

An expansion like this is particularly useful in numerical computations. For example, since $A^k x_i = \lambda_i^k x_i$ ($i = 1, \dots, n$), it follows that

$$A^k y = \gamma_1^k x_1 + \gamma_2^k x_2 + \cdots + \gamma_n^k x_n. \quad (1.7)$$

Thus we can replace the process of computing $A^k y$ by multiplication of y by A with the less expensive process of taking the powers of the eigenvalues and summing their products with their eigenvectors. Unfortunately, Example 1.6 shows that not all matrices have complete systems of eigenvectors.

The second difficulty with defective matrices is that defective eigenvalues are very sensitive to perturbations in the matrix elements, as the following example shows.

Example 1.7. The matrix

$$\tilde{J}_2(0) = \begin{pmatrix} 0 & 1 \\ \epsilon & 0 \end{pmatrix}$$

differs from $J_2(0)$ by a perturbation of ϵ in its $(2, 1)$ -element. But its eigenvalues are $\pm\sqrt{\epsilon}$. This represents a dramatic perturbation of the eigenvalue. If, for example, $\epsilon = 10^{-16}$, the perturbed eigenvalues are $\pm 10^{-8}$ — eight orders of magnitude greater than the perturbation in the matrix.

Because rounding a defective matrix will generally cause its eigenvalues to become simple, it is sometimes argued that defective matrices do not exist in practice. However, a matrix that is near a defective matrix is not much better than a defective matrix. It may have a complete system of eigenvectors, but those eigenvectors will be nearly linearly dependent, which makes the computation of expansions like (1.6) problematic. Moreover, the perturbed eigenvalues, though distinct, are just as sensitive as the original multiple eigenvalues.

Thus defectiveness is a phenomenon we will have to live with. In some cases, where defectiveness is only incidental to the problem, we can work around it—for example, by using a Schur decomposition (Theorem 1.12). In other cases we must come to grips with it. We could wish our tools were sharper.

Simple eigenvalues

We have just seen that multiple eigenvalues may be accompanied by theoretical and practical difficulties that do not affect their nonmultiple brethren. To distinguish this latter class of eigenvalues, we will say that an eigenvalue is *simple* if it has algebraic multiplicity one. We will also say the the corresponding eigenvector and eigenpair are simple.

1.3. SIMILARITY TRANSFORMATIONS

A key strategy in matrix computations is to transform the matrix in question to a form that makes the problem at hand easy to solve. The class of transformations and the final form will depend on the problem. For example, in transforming the least squares problem of minimizing $\|y - Xb\|_2$ one uses orthogonal matrices because they preserve the 2-norm. In fact, one of the most successful algorithms for solving least squares problems is based on the orthogonal reduction of X to triangular form.

For the eigenvalue problem, the transformations of choice are called similarity transformations.

Definition 1.8. Let A be of order n and let U be nonsingular. Then the matrices A and $B = U^{-1}AU$ are said to be **SIMILAR**. We also say that B is obtained from A by a **SIMILARITY TRANSFORMATION**.

The following is an important example of a similarity transformation.

Example 1.9. Let the matrix A have a complete system of eigenpairs (λ_i, x_i) ($i = 1, \dots, n$). Let

$$X = (x_1 \ \cdots \ x_n) \quad \text{and} \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n).$$

The the individual relations $Ax_i = \lambda_i x_i$ can be combined in the matrix equation

$$AX = X\Lambda.$$

Because the eigenvectors x_i are linearly independent, the matrix X is nonsingular. Hence we may write

$$X^{-1}AX = \Lambda.$$

Thus we have shown that a matrix with a complete system of eigenpairs can be reduced to diagonal form by a similarity transformation, whose columns are eigenvectors of A .

Conversely, by reversing the above argument, we see that if A can be diagonalized by a similarity transformation $X^{-1}AX$, then the columns of X are eigenvectors of A , which form a complete system.

A nice feature of similarity transformations is that they affect the eigensystem of a matrix in a systematic way, as the following theorem shows.

Theorem 1.10. Let A be of order n and let $B = U^{-1}AU$ be similar to A . Then the eigenvalues of A and B are the same and have the same multiplicities. If (λ, x) is an eigenpair of A , then $(\lambda, U^{-1}x)$ is an eigenpair of B .

Proof. Since $\det(U^{-1})\det(U) = \det(U^{-1}U) = \det(I) = 1$,

$$\begin{aligned}\det(\lambda I - A) &= \det(U^{-1})\det(\lambda I - A)\det(U) \\ &= \det(\lambda I - U^{-1}AU) \\ &= \det(\lambda I - B).\end{aligned}$$

Thus A and B have the same characteristic polynomial and hence the same eigenvalues with the same multiplicities.

If (λ, x) is an eigenpair of A , then

$$B(U^{-1}x) = U^{-1}AUU^{-1}x = U^{-1}Ax = \lambda(U^{-1}x),$$

so that $(\lambda, U^{-1}x)$ is an eigenpair of B . ■

In addition to preserving the eigenvalues of a matrix (and transforming eigenvectors in a predictable manner), similarity transformations preserve functions of the eigenvalues. The determinant of a matrix, which is the product of its eigenvalues, is clearly unchanged by similarity transformations. The following theorem shows that the trace is also invariant.

Theorem 1.11. Let A be of order n . The trace of A , i.e.,

$$\text{trace}(A) \stackrel{\text{def}}{=} \sum_{i=1}^n a_{ii},$$

is the sum of the eigenvalues of A counting multiplicities and hence is unchanged by similarity transformations.

Proof. Let the eigenvalues of A be $\lambda_1, \dots, \lambda_n$. The characteristic polynomial of A can be written in two ways:

$$\det(\lambda I - A) = (\lambda - \lambda_1) \cdots (\lambda - \lambda_n).$$

By expanding the determinant on the right, we find that the coefficient of λ^{n-1} is the negative of the sum of the diagonal elements of A . From the expression on the left, we find that the same coefficient is the negative of the sum of the eigenvalues of A . ■

This result is invaluable in debugging matrix algorithms that proceed by similarity transformations. Just print out the trace after each transformation. If it changes significantly, there is a bug in the code.

We have seen (Example 1.9) that a matrix with a complete system of eigenpairs can be reduced to diagonal form by a similarity transformation. Unfortunately, not all matrices can be so reduced; and even when one can, its matrix of eigenvectors may be nearly singular. Consequently, we will now investigate the existence of intermediate forms that stop before full diagonality.

1.4. SCHUR DECOMPOSITIONS

Unitary similarities

This subsection is devoted to the reduction of matrices by unitary and orthogonal similarities. Unitary transformations play a leading role in matrix computations. They are trivial to invert, so that a unitary transformation can easily be undone. They have spectral norm one, so that they do not magnify errors. More important, if, say,

$$B = U^H A U,$$

where U is unitary and we perturb B by an error to get $\tilde{B} = B + F$, we can write

$$\tilde{B} = U^H (A + E) U, \quad \text{where } E = UFU^H.$$

Moreover, $\|E\|_2 = \|F\|_2$ [see (2.1.7)]. Thus we can throw an error made in a computed transformation back on the original matrix without magnification. For example, if we set a small element of B to zero — say when an iteration is deemed to have converged — it is equivalent to making a perturbation of the same size in A . For an example, see (2.26), Chapter 2.

Schur form

In asking how far one can reduce a matrix by a class of transformations, it is useful to ask how many degrees of freedom the transforming matrices possesses. A unitary matrix has n^2 elements. But these elements are constrained by $n(n+1)/2$ conditions: $n(n-1)/2$ orthogonality conditions and n normality conditions. This leaves $n(n-1)/2$ degrees of freedom. Since a triangular matrix has $n(n-1)/2$ zero elements, we can hope to reduce a matrix A to triangular form by a unitary similarity. The following theorem shows that this hope is realized.

Theorem 1.12 (Schur). *Let A be of order n . Then there is a unitary matrix U such that*

$$U^H A U = T,$$

where T is upper triangular. By appropriate choice of U , the eigenvalues of A , which are the diagonal elements of T , may be made to appear in any order.

Proof. Let $\lambda_1, \dots, \lambda_n$ be an ordering of the eigenvalues of A . Let x_1 be an eigenvector corresponding to λ_1 and assume without loss of generality that $x^H x = 1$. Let the matrix $X = (x_1 \ X_2)$ be unitary. Then

$$\begin{aligned} X^H A X &= \begin{pmatrix} x_1^H \\ X_2^H \end{pmatrix} A \begin{pmatrix} x_1 & X_2 \end{pmatrix} \\ &= \begin{pmatrix} x_1^H A x_1 & x_1^H A X_2 \\ X_2^H A x_1 & X_2^H A X_2 \end{pmatrix} \\ &= \begin{pmatrix} \lambda_1 x_1^H x_1 & x_1^H A X_2 \\ \lambda_1 X_2^H x_1 & X_2^H A X_2 \end{pmatrix}, \end{aligned}$$

the last equality following from the relation $Ax_1 = \lambda_1 x_1$. Because X is unitary we have $x_1^H x_1 = 1$ and $x_1^H X_2 = 0$. Hence

$$X^H A X = \begin{pmatrix} \lambda_1 & x_1^H A X_2 \\ 0 & X_2^H A X_2 \end{pmatrix} \equiv \begin{pmatrix} \lambda_1 & t_{12}^H \\ 0 & M \end{pmatrix}. \quad (1.8)$$

Now by (1.5), the eigenvalues of M are $\lambda_2, \dots, \lambda_n$. By an obvious induction, we may assume that there is a unitary matrix V such that $V^H M V = T_{22}$, where T_{22} is an upper triangular matrix with the eigenvalues $\lambda_2, \dots, \lambda_n$ appearing on its diagonal in the prescribed order. If we set

$$U = (x_1 \ X_2) \begin{pmatrix} 1 & 0 \\ 0 & V \end{pmatrix},$$

then it is easily verified that

$$U^H A U = \begin{pmatrix} \lambda_1 & t_{12} \\ 0 & T_{22} \end{pmatrix},$$

which has the required form. ■

Here are some comments on this theorem.

- The decomposition $A = UTU^H$ is called a *Schur decomposition* of A . It is not unique because it depends on the order of the eigenvalues in T . In addition multiple eigenvalues can cause the decomposition not to be unique—e.g., there may be more

than one choice of the eigenvector x_1 in the proof of Theorem 1.12. Unique or not, the columns of U are called *Schur vectors*.

- Given the eigenpair (λ_1, x_1) , the unitary matrix $X = (x_1 \ X_2)$ in the proof can be constructed as a Householder transformation [see (2.19), Chapter 2].
- The proof of Theorem 1.12 is not constructive, since it requires a ready supply of eigenpairs. However, if one has an eigenpair at hand, it provides a constructive method for reducing the eigenvalue problem to a smaller one in which the eigenpair has been removed [see (1.8)]. This process is known as *deflation* and is an important technique in eigencalculations.
- When A has a complete system of eigenvectors, the Schur vectors bear an interesting relation to the eigenvectors. Let X be the matrix of eigenvectors of A , so that

$$X^{-1}AX = \Lambda, \quad (1.9)$$

where Λ is the diagonal matrix of eigenvalues. Let

$$X = UR$$

be the QR decomposition of X . Then (1.9) is equivalent to

$$U^H AU = R\Lambda R^{-1}.$$

The right-hand side of this relation is upper triangular with the eigenvalues of A on its diagonal. Hence $U^H AU$ is a Schur decomposition of A . In other words, the Schur vectors are the columns of the Q-factor of the matrix of eigenvectors arranged in the prescribed order.

- Schur decompositions can often replace other, more complicated forms in theoretical and computational applications. For example, the Jordan canonical form, to be treated later, shows that the geometric multiplicity of an eigenvalue cannot exceed its algebraic multiplicity. To prove the same fact via the Schur decomposition, let λ have algebraic multiplicity k . Let the first k diagonals of $T = U^H AU$ be λ , so that all the others are distinct from λ . Then it is easy to show that the last $n-k$ components of any eigenvector corresponding to λ must be zero [see (1.2)]. It follows that the eigenvectors corresponding to λ are confined to a subspace of dimension k and cannot span a subspace of dimension greater than k .

Nilpotent matrices

We now turn to applications of the Schur decomposition, beginning with an analysis of nilpotent matrices. A matrix A is *nilpotent* if $A^k = 0$ for some $k \geq 1$. The zero matrix is a trivial example of a nilpotent matrix. A nontrivial example is the matrix $J_2(0)$ of Example 1.7, whose square is zero. In general, if T is an upper triangular matrix of order n with zeros on its diagonal, then the diagonal and first $k-1$ superdiagonals of T^k are zero, as can easily be verified. Hence $T^n = 0$.

The following theorem generalizes the above observations.

Theorem 1.13. A matrix A of order n is nilpotent if and only if its eigenvalues are all zero. In this case,

$$A^n = 0.$$

Proof. Let $A^k = 0$ and let (λ, x) be an eigenpair of A . Then from (1.1),

$$0 = A^k x = \lambda^k x,$$

and it follows that $\lambda = 0$.

Conversely, suppose that the eigenvalues of A are all zero. Then any Schur decomposition of A has the form

$$A = UTU^H,$$

where T has zeros on its diagonal. It follows that

$$A^n = UT^nU^H = 0. \quad \blacksquare$$

Hermitian matrices

The elementary facts about Hermitian matrices can be obtained almost effortlessly from the Schur form. Specifically, if A is Hermitian and $T = U^H A U$ is a Schur form of A , then

$$T^H = (U^H A U)^H = U^H A^H U = U^H A U = T,$$

so that T itself is Hermitian. Thus T is both upper and lower triangular and hence is diagonal, say,

$$T = U^H A U = \Lambda \equiv \text{diag}(\lambda_1, \dots, \lambda_n).$$

Since the diagonal elements of a Hermitian matrix are real, the λ_i are real. Moreover, from the relation

$$AU = U\Lambda,$$

we see that if u_i denotes the i th column of U then (λ_i, u_i) is an eigenpair of A .

To summarize:

If A is Hermitian, its eigenvalues are real, and it has a complete orthonormal system of eigenvectors. We call the decomposition

$$A = U\Lambda U^H \tag{1.10}$$

the SPECTRAL DECOMPOSITION OF A .

Normal matrices

The key observation in establishing (1.10) is that the triangular factor in any Schur decomposition of a Hermitian matrix is diagonal. This property is true of a more general class of matrices.

Definition 1.14. A matrix A is normal if $A^H A = AA^H$.

The three naturally occurring examples of normal matrices are Hermitian matrices, skew Hermitian matrices (matrices for which $A^H = -A$), and unitary matrices.

A key fact about normal matrices is contained in the following theorem, which uses the Frobenius norm $\|\cdot\|_F$ to be introduced in §2.1.

Theorem 1.15. Let the normal matrix A be partitioned in the form

$$A = \begin{pmatrix} L & H \\ G & M \end{pmatrix}.$$

Then $\|G\|_F = \|H\|_F$.

Proof. By computing the (1,1)-blocks of each side of the relation $A^H A = AA^H$ we find that

$$L^H L + G^H G = LL^H + HH^H.$$

Since for any matrix B

$$\|B\|_F^2 = \text{trace}(B^H B) = \text{trace}(BB^H),$$

it follows that

$$\|L\|_F^2 + \|G\|_F^2 = \|L\|_F^2 + \|H\|_F^2.$$

Hence $\|G\|_F = \|H\|_F$. ■

If A is a normal matrix and U is unitary, then $U^H AU$ is also normal. Consequently, if T is the triangular factor in a Schur decomposition of A , the matrix T is normal. We claim that T is also diagonal. To see this, partition

$$T = \begin{pmatrix} t_{11} & t_{12}^H \\ 0 & T_{22} \end{pmatrix}.$$

By Theorem 1.15, $t_{12} = 0$. It then follows from the relation $T^H T = TT^H$ that $T_{22}^H T_{22} = T_{22} T_{22}^H$, so that T_{22} is normal. An obvious induction now establishes the claim.

It follows that:

A normal matrix has a complete orthonormal system of eigenvectors. (1.11)

We add for completeness that the eigenvalues of a unitary matrix have absolute value one. The eigenvalues of a skew Hermitian matrix are zero or purely imaginary.

1.5. BLOCK DIAGONALIZATION AND SYLVESTER'S EQUATION

A Schur decomposition of a matrix A represents a solution of the eigenvalue problem in the sense that the eigenvalues of the matrix in question can be obtained by inspecting the diagonals of the triangular factor. But only the first Schur vector—the first column of the orthogonal factor—is an eigenvector. We have seen (Example 1.9) that if we can diagonalize A , then the diagonalizing transformation provides a complete set of eigenvectors. Unfortunately, a defective matrix, which has no complete system of eigenvectors, cannot be diagonalized. Thus to proceed beyond Schur form we must look to coarser decompositions, such as a block diagonal form.

From block triangular to block diagonal matrices

Let A have the block triangular form

$$A = \begin{pmatrix} L & H \\ 0 & M \end{pmatrix},$$

where L and M are square. Such a matrix could, for example, be obtained by partitioning a Schur decomposition, in which case L and M would be upper triangular. But we make no such assumption.

Suppose that we wish to reduce A to block diagonal form by a similarity transformation—preferably a simple one. We must, of course, preserve the block upper triangularity of A , which can be accomplished by making the transformation itself block upper triangular. We can make the transformation even simpler by requiring its diagonal blocks to be identity matrices. Thus we are led to a transformation of the form

$$\begin{pmatrix} I & Q \\ 0 & I \end{pmatrix},$$

whose inverse is

$$\begin{pmatrix} I & -Q \\ 0 & I \end{pmatrix}.$$

Applying this transformation to A we get

$$\begin{pmatrix} I & -Q \\ 0 & I \end{pmatrix} \begin{pmatrix} L & H \\ 0 & M \end{pmatrix} \begin{pmatrix} I & Q \\ 0 & I \end{pmatrix} = \begin{pmatrix} L & LQ - QM + H \\ 0 & M \end{pmatrix}.$$

Thus to achieve block diagonality we must have

$$LQ - QM = -H.$$

This equation is linear in the elements of Q . If we can solve it, we can block diagonalize A . However, we must first investigate conditions under which the equation has a solution.

Sylvester's equation

Let us consider the general problem of solving an equation of the form

$$AX - XB = C,$$

where A and B are square, say of orders n and m . This equation is called a *Sylvester equation*, and it occurs in many important applications. Here we will be concerned with sufficient conditions for the existence of a solution.

If we introduce the *Sylvester operator* $\mathbf{S}: \mathbb{C}^{n \times m} \rightarrow \mathbb{C}^{n \times m}$ defined by

$$\mathbf{S}X = AX - XB,$$

then we may write Sylvester's equation in the form

$$\mathbf{S}X = C.$$

The operator \mathbf{S} is linear in X . Hence, our problem reduces to that of determining when \mathbf{S} is nonsingular. It turns out that the nonsingularity depends on the spectra of A and B . Specifically, we have the following theorem.

Theorem 1.16. *The Sylvester operator \mathbf{S} is nonsingular if and only if the spectra of A and B are disjoint, i.e., if and only if*

$$\Lambda(A) \cap \Lambda(B) = \emptyset.$$

Proof. To prove the necessity of the condition, let (λ, x) be a right eigenpair of A and let (μ, y) be a left eigenpair of B . Let $X = xy^H$. Then

$$\mathbf{S}X = Axy^H - xy^H B = \lambda xy^H - \mu xy^H = (\lambda - \mu)X. \quad (1.12)$$

Hence if we can choose $\lambda = \mu$ — that is, if A and B have a common eigenvalue — then X is a “null vector” of \mathbf{S} , and \mathbf{S} is singular.

To prove the converse, we use the fact that an operator is nonsingular if every linear system in the operator has a solution.

Consider the system $\mathbf{S}X = C$. The first step is to transform this system into a more convenient form. Let $T = V^H B V$ be a Schur decomposition of B . Then Sylvester's equation can be written in the form

$$A(XV) - (XV)(V^H B V) = (CV).$$

If we set

$$Y = XV \quad \text{and} \quad D = CV,$$

we may write the transformed equation in the form

$$AY - YT = D.$$

Let us partition this system in the form

$$A(y_1 \ y_2 \ y_3 \ \dots) - (y_1 \ y_2 \ y_3 \ \dots) \begin{pmatrix} t_{11} & t_{12} & t_{13} & \dots \\ 0 & t_{22} & t_{23} & \dots \\ 0 & 0 & t_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} = (d_1 \ d_2 \ d_3 \ \dots). \quad (1.13)$$

The first column of this system has the form

$$Ay_1 - t_{11}y_1 = (A - t_{11}I)y_1 = d_1.$$

Now t_{11} is an eigenvalue of B and hence is not an eigenvalue of A . Thus $A - t_{11}I$ is nonsingular, and the first column of Y is well defined as the solution of the equation $(A - t_{11}I)y_1 = d_1$.

From the second column of the system (1.13) we get the equation

$$(A - t_{22}I)y_2 = d_2 + t_{12}y_1.$$

Because y_1 is well defined the right-hand side of this system is well defined. Moreover, as argued above, $A - t_{22}I$ is nonsingular. Hence y_2 is well defined.

In general, suppose that we have found y_1, \dots, y_{k-1} . From the k th column of (1.13) we get

$$(A - t_{kk}I)y_k = d_k + \sum_{i=1}^{k-1} t_{ik}y_i.$$

The right-hand side of this equation is well defined and the matrix on the left is nonsingular. Hence y_k is well defined.

We have found a solution Y of the equation $AY - YT = D$. Hence $X = YV^H$ is a solution of $\mathbf{S}X = C$. ■

We may restate this theorem in a way that will prove useful in the sequel. Let $\mathbf{S}X = \sigma X$, so that (σ, X) is an eigenpair of \mathbf{S} . Equivalently, $(A - \sigma I)X - XB = 0$ so that the Sylvester operator $X \mapsto (A - \sigma I)X - XB$ is singular. It follows from Theorem 1.16 that this can be true if and only if there is a $\lambda \in \Lambda(A)$ and $\mu \in \Lambda(B)$ such that $\lambda - \sigma = \mu$, or equivalently $\sigma = \lambda - \mu$. Conversely, if $\lambda \in \Lambda(A)$ and $\mu \in \Lambda(B)$, then (1.12) shows that $(\lambda - \mu) \in \Lambda(\mathbf{S})$. Thus we have the following corollary.

Corollary 1.17. *The eigenvalues of \mathbf{S} are the differences of the eigenvalues of A and B .*

Given matrices $A \in \mathbb{C}^{n \times n}$ and $B \in \mathbb{C}^{m \times m}$, this algorithm solves Sylvester's equation

$$AX - XB = C.$$

In the code below all partitionings are by columns.

1. Compute a Schur decomposition $A = USU^H$
2. Compute a Schur decomposition $B = VTV^H$
3. $D = U^HCV$
4. **for** $k = 1$ to m
5. Solve the triangular system $(S - t_{kk}I)y_k = d_k + \sum_{i=1}^{k-1} t_{ik}y_i$
6. **end for** k
7. $X = UYV^H$

Algorithm 1.1: Solution of Sylvester's Equation

An algorithm

Theorem 1.16 and its corollary are sufficient for our immediate purposes. But it is instructive to reflect on the the second part of the proof.

If we have a method for computing Schur decompositions, the procedure of transforming the Sylvester equation into the form $AY - YT = D$ and solving columnwise for Y represents an algorithm for solving Sylvester's equation. As it stands, it involves solving m systems whose matrices are $(A - r_{kk}I)$. For a general matrix A these systems would require $O(mn^3)$ operations to solve. Unless m is quite small, this operation count is prohibitive. However, by a further transformation of the problem we can reduce the count for solving the systems to $O(mn^2)$.

Specifically, let $S = U^H AU$ be a Schur decomposition of A . Then we can write Sylvester's equation in the form

$$(U^H AU)(U^H XV) - (U^H XV)(V^H BV^H) = (U^H CU),$$

or equivalently

$$SY - YT = D,$$

where $Y = U^H XV$ and $D = U^H CU$. If we now solve columnwise for Y , as in the proof of Theorem 1.16, the systems have upper triangular matrices of the form $S - t_{kk}I$, which can be solved in $O(n^2)$ operations.

These observations form the basis for one of the best algorithms for solving dense Sylvester equations. It is summarized in Algorithm 1.1. More details are given in the notes and references.

Block diagonalization redux

Let us now return to our original goal of block diagonalizing the matrix

$$A = \begin{pmatrix} L & H \\ 0 & M \end{pmatrix}.$$

We have seen that a sufficient condition for a diagonalizing similarity to exist is that the equation $LQ - QM = -H$ have a solution. Moreover, a sufficient condition for this Sylvester equation to have a solution is that L and M have disjoint spectra. Hence we have the following theorem.

Theorem 1.18. *Let*

$$A = \begin{pmatrix} L & H \\ 0 & M \end{pmatrix}$$

and suppose that

$$\Lambda(L) \cap \Lambda(M) = \emptyset.$$

Then there is a unique matrix Q satisfying

$$LQ - QM = -H$$

such that

$$\begin{pmatrix} I & -Q \\ 0 & I \end{pmatrix} \begin{pmatrix} L & H \\ 0 & M \end{pmatrix} \begin{pmatrix} I & Q \\ 0 & I \end{pmatrix} = \begin{pmatrix} L & 0 \\ 0 & M \end{pmatrix}.$$

The importance of this theorem is that it can be applied recursively. Suppose, for example, that $\Lambda(A)$ can be partitioned into the union $\mathcal{L}_1 \cup \dots \cup \mathcal{L}_k$ of disjoint multisets. By the Schur decomposition theorem (Theorem 1.12) we can find a unitary matrix X such that $T = X^H A X$ is upper triangular with the eigenvalues of A appearing in the order of their appearance in the multisets $\mathcal{L}_1, \dots, \mathcal{L}_k$. Partition

$$T = \begin{pmatrix} L_1 & H \\ 0 & M \end{pmatrix},$$

where $\Lambda(L_1) = \mathcal{L}_1$. Then the eigenvalues of L_1 are disjoint from those of M , and by Theorem 1.18 we can reduce T to the form

$$\begin{pmatrix} L_1 & 0 \\ 0 & M \end{pmatrix}.$$

In the same way we may apply Theorem 1.18 to M to cut out a block L_2 corresponding to \mathcal{L}_2 . And so on, to give the following theorem.

Theorem 1.19. Let $\Lambda(A)$ be the union $\mathcal{L}_1 \cup \dots \cup \mathcal{L}_k$ of disjoint multisets. Then there is a nonsingular matrix X such that

$$X^{-1}AX = \text{diag}(L_1, \dots, L_k),$$

where $\Lambda(L_i) = \mathcal{L}_i$.

If the sets \mathcal{L}_i correspond to the distinct eigenvalues of A , we obtain the following corollary.

Corollary 1.20. Let the matrix A of order n have distinct eigenvalues $\lambda_1, \dots, \lambda_k$ with multiplicities m_1, \dots, m_k . Then there is a nonsingular matrix X such that

$$X^{-1}AX = \text{diag}(L_1, \dots, L_k),$$

where L_i is of order m_i and has only the eigenvalue λ_i .

Finally, if all the eigenvalues of A are distinct, then the blocks L_i in this corollary are scalars. Hence:

If A has distinct eigenvalues, A is diagonalizable and has a complete system of eigenvectors.

The decompositions in Theorem 1.19 and its corollaries enjoy a certain uniqueness. Let $X = (X_1 \ \dots \ X_k)$ be partitioned conformally with the partition of $\Lambda(A)$ and let $X^{-1} = (Y_1 \ \dots \ Y_k)^H$ also be partitioned conformally. Then the spaces spanned by the X_i and Y_i are unique (this fact is nontrivial to prove; see p. 246). However, if S_i is nonsingular, we may replace X_i , Y_i , and L_i in Theorem 1.19 by $X_i S_i$, $Y_i S_i^{-H}$, and $S_i^{-1} L_i S_i$. This lack of uniqueness can be useful. For example, it allows us to assume, say, that Y_i is orthonormal.

1.6. JORDAN FORM

*The reports of my death are
greatly exaggerated.*

Mark Twain

We have seen that any matrix can be reduced to triangular form by a unitary similarity transformation. It is natural to ask how much further we can go by relaxing the requirement that the transformation be unitary. The answer is the classical Jordan canonical form, which is the topic of this subsection.

We begin with a definition.

Definition 1.21. A JORDAN BLOCK $J_m(\lambda)$ is a matrix of order m of the form

$$J_m(\lambda) = \begin{pmatrix} \lambda & 1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda & 1 & \cdots & 0 & 0 \\ 0 & 0 & \lambda & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & 0 \\ 0 & 0 & 0 & \cdots & \lambda & 1 \\ 0 & 0 & 0 & \cdots & 0 & \lambda \end{pmatrix}.$$

From an algebraic point of view a Jordan block is as defective as a matrix can be. The eigenvalue λ of the matrix $J_m(\lambda)$ has algebraic multiplicity m . But $\text{rank}[\lambda I - J_m(\lambda)]$ is $m-1$, so λ has geometric multiplicity one.

The Jordan canonical form — a decomposition of a matrix into Jordan blocks — is described in the following theorem.

Theorem 1.22. Let $A \in C^{n \times n}$ have k distinct eigenvalues $\lambda_1, \dots, \lambda_k$ of algebraic multiplicities m_1, \dots, m_k . Then there are unique integers m_{ij} ($i = 1, \dots, k$, $j = 1, \dots, l_i$) satisfying

$$m_i = \sum_{j=1}^{l_i} m_{ij}$$

and a nonsingular matrix X such that

$$X^{-1}AX = \text{diag}(J_1, \dots, J_k), \quad (1.14)$$

where

$$J_i = \text{diag}(J_{m_{i1}}(\lambda_i), \dots, J_{m_{il_i}}(\lambda_i)). \quad (1.15)$$

Here are some comments about this theorem.

- The proof of the theorem is quite involved. It begins with a reduction to block diagonal form, each block corresponding to a distinct eigenvalue of A . The hard part is reducing the individual blocks to the J_i in (1.14) and (1.15). For more see the notes and references.
- The Jordan canonical form has a limited uniqueness. The multiplicities m_i and the numbers m_{ij} along with their associated eigenvalues are characteristic of the matrix A . Any reduction of A into Jordan blocks will result in the same number of blocks of the same order having the same eigenvalues. The spaces spanned by the X_i are unique. However, the columns of the X_i are not. Principal vectors may be chosen in different ways. But in spite of this nonuniqueness, we still speak of *the* Jordan canonical form of a matrix.

- We can partition X at several levels corresponding to levels in the decomposition.
- First partition

$$X = (X_1 \cdots X_k)$$

in conformity with (1.14). It then follows from the relation $AX = XJ$ that

$$AX_i = X_i J_i;$$

i.e., the action of A on the matrix X_i is entirely specified by the matrix J_i . The spaces spanned by the columns of the X_i are examples of eigenspaces (to be treated in Chapter 4).

If we now partition X_i in the form

$$X_i = (X_{i1} \cdots X_{il_i})$$

in conformity with (1.15), it can be shown that

$$AX_{ij} = X_{ij} J_{m_{ij}}(\lambda_j). \quad (1.16)$$

Thus the actions of A on the matrices X_{ij} are specified by the individual Jordan blocks in the decomposition.

Finally, if we write

$$X_{ij} = (x_1^{(ij)}, \dots, x_{m_{ij}}^{(ij)}),$$

then the vectors $x_p^{(ij)}$ satisfy

$$\begin{aligned} Ax_1^{(ij)} &= \lambda_i x_1^{(ij)}, \\ Ax_p^{(ij)} &= \lambda_i x_p^{(ij)} + x_{p-1}^{(ij)}, \quad p = 2, \dots, m_{ij}. \end{aligned} \quad (1.17)$$

A set of vectors satisfying (1.17) is called a sequence of *principal vectors* of A . Note that the first and only the first principal vector associated with a Jordan block is an eigenvector. Nonetheless the simple relations (1.17) make calculations involving principal vectors quite easy.

The Jordan canonical form is now less frequently used than previously. The simplicity of its blocks make it a keen tool for conjecturing new theorems and proving them. Against this must be set the fact that the form is virtually uncomputable. Perturbations in the matrix send the eigenvalues flying, so that, in principle, the matrix becomes diagonalizable (Example 1.7). If we know the nature of the defectiveness in the original matrix, then it may be possible to recover the original form. But otherwise attempts to compute the Jordan canonical form of a matrix have not been very successful. Under the circumstances there is an increasing tendency to turn whenever possible to friendlier decompositions—for example, to the Schur decomposition. However, as a mathematical probe the Jordan canonical form is still useful, and reports of its death are greatly exaggerated.

1.7. NOTES AND REFERENCES

General references

The algebra and analysis of eigensystems is treated in varying degrees by most books on linear and applied linear algebra. There are too many of these books to survey here. Special mention, however, should be made of Bellman's classic *Introduction to Matrix Analysis* [18] and Horn and Johnson's two books on introductory and advanced matrix analysis [118, 119]. They contain a wealth of useful information.

More or less systematic treatments of eigensystems are found in books on matrix computations. The classics of the field are Householder's *Theory of Matrices in Numerical Analysis* [121] and Wilkinson's *Algebraic Eigenvalue Problem* [300]. Golub and Van Loan's *Matrix Computations* [95] is the currently definitive survey of the field. Trefethen and Bau's *Numerical Linear Algebra* [278] is a high-level, insightful treatment with a stress on geometry. Datta's *Numerical Linear Algebra and Applications* [52] is an excellent text for the classroom, as is Watkins's *Fundamentals of Matrix Computations* [288]. Chatelin's *Eigenvalues of Matrices* [39] and Saad's *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms* [233] are advanced treatises that cover both theory and computation.

Special mention should be made of *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide* edited by Bai, Demmel, Dongarra, Ruhe, and van der Vorst [10]. It consists of sketches of all the important algorithms for large eigenproblems with brief treatments of the theoretical and computational background and an invaluable bibliography. It is a perfect complement to this volume and should be in the hands of anyone seriously interested in large eigenproblems.

History and Nomenclature

A nice treatment of the history of determinants and matrices, including eigenvalues, is given by Kline [147, Ch. 33]. He makes the interesting point that many of the results we think of as belonging to matrix theory were derived as theorems on determinants and bilinear and quadratic forms. (Matrices were introduced by Cayley in 1858 [32], but they only gained wide acceptance in the twentieth century.)

The nontrivial solutions of the equations $Ax = \lambda x$ were introduced by Lagrange [155, 1762] to solve systems of differential equations with constant coefficients. In 1829 Cauchy [31] took up the problem and introduced the term "characteristic equation," not for the polynomial equation but for the system $Ax = \lambda x$.

Our term "eigenvalue" derives from the German *Eigenwert*, which was introduced by Hilbert [116, 1904] to denote for integral equations the reciprocal of our matrix eigenvalue. At some point Hilbert's *Eigenwerte* inverted themselves and became attached to matrices. The nomenclature for integral equations remains ambiguous to this day; for example, see [213, p. 96].

Eigenvalues have been called many things in their day. The term "characteristic value" is a reasonable translation of *Eigenwert*. However, "characteristic" has an inconveniently large number of syllables and survives only in the terms "characteristic

equation” and “characteristic polynomial.” (For symmetric matrices the characteristic equation and its equivalents are also called the *secular equation* owing to its connection with the secular perturbations in the orbits of planets.) Other terms are “latent value” and “proper value” from the French *valeur propre*.

The day when purists and pedants could legitimately object to “eigenvalue” as a hybrid of German and English has long since passed. The German *eigen* has become a thoroughly naturalized English prefix meaning having to do with eigenvalues and eigenvectors. Thus we can use “eigensystem,” “eigenspace,” or “eigenexpansion” without fear of being misunderstood.

The term “eigenpair” used to denote an eigenvalue and eigenvector is a recent innovation. The earliest use I can find is in Parlett’s *Symmetric Eigenvalue Problem* [206].

Schur form

The proof of the existence of Schur decompositions is essentially the one given by Schur in 1909 [234]—an early example of the use of partitioned matrices. The decomposition can be computed stably by the QR algorithm (see §§2–3, Chapter 2), and it can often replace more sophisticated decompositions—the Jordan canonical form, for example—in matrix algorithms.

Sylvester’s equation

Sylvester’s equation is so called because J. J. Sylvester considered the homogeneous case $AX - XB = 0$ [273, 1884]. For A and B of order two he showed that the equation has a nontrivial solution if and only if A and B have a common eigenvalue (essentially Theorem 1.16) and then stated that it easily generalizes.

The linearity of the Sylvester operator means that it has a matrix representation once a convention is chosen for representing X as a vector. The customary representation is $\text{vec}(X)$ in which the columns of X are stacked atop one another in their natural order. In this representation the Sylvester equation becomes

$$(I_m \otimes A - B^T \otimes I_n) \text{vec}(X) = \text{vec}(C),$$

where \otimes is the *Kronecker product* defined by

$$R \otimes S = \begin{pmatrix} r_{11}S & r_{12}S & \cdots \\ r_{21}S & r_{22}S & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}.$$

Algorithm 1.1 is a stripped down version of one due to Bartels and Stewart [14]. The difference is that a real Schur form is used to keep the arithmetic real. Golub, Nash, and Van Loan [90] have proposed a Hessenberg–Schur variant that saves some operations.

Higham [114, Ch. 15] devotes a chapter to Sylvester’s equation, which contains error and perturbation analyses, historical comments, and an extensive bibliography.

Block diagonalization

Although rounding error prevents matrices that are exactly defective from arising in practice, one frequently encounters matrices with very ill-conditioned systems of eigenvectors. Block diagonalization would seem to be an attractive way out of this problem. The idea is to generate clusters of eigenvalues associated with nearly dependent eigenvectors, find bases for their eigenspaces, and reduce the matrix to block diagonal form. If the blocks are small, one can afford to lavish computation on their subsequent analysis.

Unfortunately, this sensible-sounding procedure is difficult to implement. Leaving aside the possibility that the smallest cluster may consist of all the eigenvalues, the problem is to find the clusters. Example 1.7 shows that eigenvalues that should be clustered do not have to lie very close to one another. Consequently, schemes for automatically block diagonalizing a matrix by a well-conditioned similarity transformation have many ad hoc features. For more see [17, 57, 60, 61, 99].

Jordan canonical form

The Jordan canonical form was established in 1874 by Camile Jordan [136]. The characteristic polynomials of the Jordan blocks of a matrix A are called the *elementary divisors* of A . Thus to say that A has linear elementary divisors is to say that A is nondefective or that A is diagonalizable.

All the above comments about computing block diagonal forms of a matrix apply to the Jordan form — and then some. See [96, 139, 140, 218] for the issues.

2. NORMS, SPECTRAL RADII, AND MATRIX POWERS

Eigenpairs are defined by a nonlinear equation and hence are creatures of analysis. But once the existence of eigenpairs is established, an algebra of eigensystems can be erected on the definition. The first section of this chapter was devoted to that algebra. We now turn to the analytic properties of eigensystems.

We begin with a brief review of matrix and vector norms, which measure the size of matrices and vectors. In the next subsection we study the relation of norms to the magnitude of the largest eigenvalue of a matrix — the *spectral radius* of the matrix. The results are then applied to determine the behavior of powers of a matrix, a topic which will often recur in the course of this volume. For more on norms see §I:4.1.

2.1. MATRIX AND VECTOR NORMS

Definition

Analysis happens when a metric meets an algebraic structure. For matrices the metric is generated by a norm, a function that in some sense measures the size of a matrix. It is a generalization of the absolute value of a scalar, as the following definition shows.

Definition 2.1. A MATRIX NORM ON $\mathbb{C}^{m \times n}$ is a function $\|\cdot\|: \mathbb{C}^{m \times n} \rightarrow \mathbb{R}$ satisfying the following three conditions:

1. $A \neq 0 \implies \|A\| > 0$,
2. $\|\mu A\| = |\mu| \|A\|$,
3. $\|A + B\| \leq \|A\| + \|B\|$.

The first two conditions are natural properties of any measure of size. The third condition—called the *triangle inequality*—specifies the interaction of the norm with the matrix sum. It is frequently used in the equivalent form

$$\|A - B\| \geq \|\|A\| - \|B\|\|.$$

In this work we will always identify $\mathbb{C}^{n \times 1}$ with the space of n -vectors \mathbb{C}^n , and we call a norm on either space a *vector norm*.

Examples of norms

The norms most widely used in matrix computations are the 1-norm, the Frobenius norm, and the ∞ -norm defined by

1. $\|A\|_1 = \max_j \sum_i |a_{ij}|$,
2. $\|A\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$,
3. $\|A\|_\infty = \max_i \sum_j |a_{ij}|$.

Each of these norms is actually a *family of matrix norms* in the sense that each is defined for matrices of all orders. In particular, these norms are defined for vectors, in which case their definitions take the following forms:

1. $\|x\|_1 = \sum_i |x_i|$,
2. $\|x\|_F = \sqrt{\sum_i |x_i|^2}$,
3. $\|x\|_\infty = \max_i |x_i|$.

The Frobenius norm has the useful characterization

$$\|A\|_F^2 = \text{trace}(A^H A) = \text{trace}(AA^H),$$

which in the case of a vector reduces to

$$\|x\|_F^2 = x^H x.$$

The Frobenius vector norm is also called the Euclidean norm because in \mathbb{R}^2 and \mathbb{R}^3 the quantity $\|x\|_F$ is the ordinary Euclidean length x . The Frobenius norm of a vector x is usually written $\|x\|_2$, for reasons that will become evident shortly.

Operator norms

Let μ be a vector norm on \mathbb{C}^m and ν be a vector norm on \mathbb{C}^n . Then it is easily verified that the function $\|\cdot\|_{\mu,\nu}$ defined on $\mathbb{C}^{m \times n}$ by

$$\|A\|_{\mu,\nu} = \max_{\nu(x)=1} \mu(Ax)$$

is a matrix norm. It is called the *operator norm generated by μ and ν* or the *matrix norm subordinate to μ and ν* .

We have already seen examples of operator norms. The vector 1-norm generates the matrix 1-norm and the vector ∞ -norm generates the matrix ∞ -norm.

The vector 2-norm generates a new norm called the *matrix 2-norm* or the *spectral norm*. It is written $\|\cdot\|_2$. Unlike the 1-norm, the ∞ -norm, and the Frobenius norm, it is difficult to calculate. Nonetheless it is of great importance in the analysis of matrix algorithms. Here is a list of some of its properties.

1. $\|A\|_2$ is the largest singular value of A .
 2. $\|A\|_2^2$ is the largest eigenvalue of $A^H A$ or AA^H .
 3. $\|A\|_2 = \max_{\|x\|_2=\|y\|_2=1} |y^H Ax|$.
 4. $\|A\|_2 \leq \|A\|_F$.
 5. $\|A\|_2^2 \leq \|A\|_1 \|A\|_\infty$.
 6. If U is orthonormal, then $\|U\|_2 = 1$.
 7. If U and V are orthonormal, then $\|A\|_2 = \|UA\|_2 = \|AV^H\|_2$.
- (2.1)

Because of property 7, we say that the 2-norm is *unitarily invariant*. The Frobenius norm is also unitarily invariant.

Norms and rounding error

As an example of the use of norms, let us examine what happens when a matrix A is rounded. Specifically, suppose that the machine in question has *rounding unit* ϵ_M —roughly the largest number for which $\text{fl}(1 + \epsilon) = 1$, where fl denotes floating point evaluation. Denote by $\text{fl}(A)$ the result of rounding the elements of A . Then element-wise

$$\text{fl}(a_{ij}) = a_{ij}(1 + \epsilon_{ij}),$$

where

$$|\epsilon_{ij}| \leq \epsilon_M.$$

In terms of matrices, we have $\text{fl}(A) = A + E$, where the elements of E are $a_{ij}\epsilon_{ij}$. It follows that

$$|E| \leq |A|\epsilon_M.$$

Thus in any norm that satisfies $\| |A| \| = \| A \|$ (such a norm is called an *absolute norm*), we have $\| E \| \leq \| A \| \epsilon_M$ or

$$\frac{\| E \|}{\| A \|} \leq \epsilon_M. \quad (2.2)$$

The left-hand side of (2.2) is called the *normwise relative error* in $\text{fl}(A)$. What we have shown is that:

In any absolute norm, rounding a matrix produces a normwise relative error that is bounded by the rounding unit. (2.3)

Incidentally, the commonly used one, Frobenius, and infinity norms are absolute norms. Unfortunately, the 2-norm is not; although (2.2) is still a good approximation.

Limitations of norms

We have just shown that if the elements of a matrix have small relative error, then the matrix as a whole has a small normwise relative error. Unfortunately the converse is not true, as the following example shows.

Example 2.2. Let

$$x = \begin{pmatrix} 1 \\ 10^{-3} \\ 10^{-5} \end{pmatrix} \quad \text{and} \quad \tilde{x} = \begin{pmatrix} 1.00001 \\ 1.01 \cdot 10^{-3} \\ 2 \cdot 10^{-5} \end{pmatrix}.$$

Then it is easily verified that

$$\frac{\| \tilde{x} - x \|_\infty}{\| x \|_\infty} = 10^{-5},$$

so that \tilde{x} has low normwise relative error. On the other hand, the relative errors in the components vary. It is 10^{-5} for the first component, 10^{-2} for the second, and 1 for the third, which has no accuracy at all.

In general, with balanced norms like the ∞ -norm, a small normwise relative error guarantees a similar relative error only in the largest elements of the matrix; the smaller elements can be quite inaccurate. Now in many applications the accuracy of small elements is unimportant. But in others it is, and in such applications one must either scale the problem to fit the norm or use a norm that gives greater weight to small elements.

Consistency

The triangle inequality provides a useful way of bounding the norm of the sum of two matrices. The property of consistency does the same thing for the product of two matrices.

Definition 2.3. Let $\|\cdot\|_{l,m}$, $\|\cdot\|_{m,n}$, and $\|\cdot\|_{l,n}$ be matrix norms defined on $\mathbb{C}^{l \times m}$, $\mathbb{C}^{m \times n}$, and $\mathbb{C}^{l \times n}$. These norms are CONSISTENT if for all $A \in \mathbb{C}^{l \times m}$ and $B \in \mathbb{C}^{m \times n}$ we have

$$\|AB\|_{l,n} \leq \|A\|_{l,m} \|B\|_{m,n}.$$

Four comments on this definition.

- An important special case is a matrix norm $\|\cdot\|$ defined on $\mathbb{R}^{n \times n}$. In this case consistency reduces to

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

- Matrix norms do not have to be consistent. For example, the function defined by

$$\|A\| = \max_{i,j} |\alpha_{ij}|$$

is a norm. But it is not a consistent norm. For example, if

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix},$$

then $2 = \|A^2\| > \|A\| \|A\| = 1$.

- The 1-, 2-, and ∞ -norms as well as the Frobenius norm are consistent in the sense that for $p = 1, 2, \infty, F$,

$$\|AB\|_p \leq \|A\|_p \|B\|_p$$

whenever the product AB is defined.

- The Frobenius and 2-norms are consistent in the sense that

$$\|AB\|_F \leq \|A\|_F \|B\|_2, \|A\|_2 \|B\|_F.$$

—

Given a consistent matrix norm $\|\cdot\|$ on $\mathbb{C}^{n \times n}$ it is natural to ask if there is a vector norm ν on \mathbb{C}^n that is consistent with $\|\cdot\|$. In fact there are many such norms. For let $y \in \mathbb{C}^n$ be nonzero. Then it is easily verified that the function ν defined by

$$\nu(x) = \|xy^H\|$$

is a vector norm. But

$$\nu(Ax) = \|Axy^H\| \leq \|A\| \|xy^H\| = \|A\| \nu(x).$$

Thus we have shown that:

$$\text{Any consistent matrix norm on } \mathbb{C}^{n \times n} \text{ has a consistent vector norm.} \quad (2.4)$$

Another useful construction is to define a norm by multiplying by a matrix. The proof of the following theorem is left as an exercise.

Theorem 2.4. *Let $\|\cdot\|$ be a norm on $\mathbb{C}^{n \times n}$. If U is nonsingular, then the functions*

$$\nu_{UA}(A) \equiv \|UA\|, \quad \nu_{AU}(A) \equiv \|AU\|, \quad \text{and} \quad \nu_{\text{sim}}(A) \equiv \|U^{-1}AU\|$$

are norms. If $\|\cdot\|$ is consistent, then so is $\nu_{\text{sim}}(A)$.

For a proof see Theorem I:4.4.6.

Norms and convergence

Given a matrix norm $\|\cdot\|$ on $\mathbb{C}^{m \times n}$, we may define the distance between two matrices A and B by $\|A - B\|$. This distance function can in turn be used to define a notion of convergence for matrices. Specifically, if $\{A_k\}$ is a sequence of matrices, we write

$$\lim_{k \rightarrow \infty} A_k = B \quad \text{if} \quad \lim_{k \rightarrow \infty} \|A_k - B\| = 0. \quad (2.5)$$

At first glance, this definition of convergence appears to depend on the choice of the norm $\|\cdot\|$. But in fact, if a sequence converges in one norm, then it converges in any norm. This is an immediate consequence of the following theorem on the equivalence of norms.

Theorem 2.5. *Let μ and ν be norms on $\mathbb{C}^{m \times n}$. Then there are positive constants σ and τ such that for all $A \in \mathbb{C}^{m \times n}$*

$$\sigma\mu(A) \leq \nu(A) \leq \tau\mu(A).$$

Thus it does not matter which norm we choose to define convergence. In particular, it is easy to see that $\|A_k - B\|_F \rightarrow 0$ if and only if $|a_{ij}^{(k)} - b_{ij}| \rightarrow 0$ ($i = 1, \dots, m$, $j = 1, \dots, n$). Thus convergence in any norm is equivalent to the componentwise convergence of the elements of a matrix.

Although all norms are mathematically equivalent, as a practical matter they may say very different things about a converging sequence, at least in the initial stages. For an example, see the discussion following Example 2.11.

2.2. THE SPECTRAL RADIUS

Definition

In this subsection we will explore the relation of the largest eigenvalue of a matrix to its norms. We begin with a simple but far-reaching definition.

Definition 2.6. Let A be of order n . Then the **SPECTRAL RADIUS** of A is the number

$$\rho(A) \stackrel{\text{def}}{=} \max_{\lambda \in \Lambda(A)} |\lambda|.$$

We call any eigenpair (λ, x) for which $|\lambda| = \rho(A)$ a **DOMINANT EIGENPAIR**. The eigenvalue λ is called a **DOMINANT EIGENVALUE** and its eigenvector is called a **DOMINANT EIGENVECTOR**.

Thus the spectral radius is the magnitude of the largest eigenvalue of A . It gets its name from the fact that it is the radius of the smallest circle centered at the origin that contains the eigenvalues of A .

Norms and the spectral radius

The spectral radius has a normlike flavor. If it is large, the matrix must be large. It also tracks the powers of a matrix with greater fidelity than a norm. Specifically, $\rho(A^k) = \rho(A)^k$, whereas the best we can say for a consistent norm is that $\|A^k\| \leq \|A\|^k$. We begin our investigation of norms and spectral radii with a trivial but important theorem.

Theorem 2.7. Let A be of order n , and let $\|\cdot\|$ be a consistent norm on $\mathbb{C}^{n \times n}$. Then

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

Proof. Let ν be a vector norm consistent with the matrix norm $\|\cdot\|$ [see (2.4)]. If (λ, x) is an eigenpair of A with $\nu(x) = 1$, then

$$\|A\| = \|A\|\nu(x) \geq \nu(Ax) = \nu(\lambda x) = |\lambda|\nu(x) = |\lambda|.$$

Since λ is an arbitrary eigenvalue of A ,

$$\|A\| \geq \max_{\lambda \in \Lambda(A)} |\lambda| = \rho(A). \blacksquare$$

In general, a consistent norm will exceed the spectral radius of A . For example, the Jordan block

$$J_2(0) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

has spectral radius zero, but $\|J_2(0)\|_\infty = 1$. In fact, something worse is true. Since $J_2(0)$ is nonzero, any norm of $J_2(0)$ must be positive. In other words, there is no norm that reproduces the spectral radius of $J_2(0)$. However, there are norms that approximate it, as the following theorem shows.

Theorem 2.8. Let A be of order n . Then for any $\epsilon > 0$ there is a consistent norm $\|\cdot\|_{A,\epsilon}$ (depending on A and ϵ) such that

$$\|A\|_{A,\epsilon} \leq \rho(A) + \epsilon. \quad (2.6)$$

If the dominant eigenvalues of A are nondefective, then there is a consistent norm $\|\cdot\|_A$ (depending on A) such that

$$\|A\|_A = \rho(A).$$

Proof. Let

$$U^H A U = \Lambda + N$$

be a Schur form of A in which Λ is diagonal and N is strictly upper triangular. For $\eta > 0$, let

$$D_\eta = \text{diag}(1, \eta, \dots, \eta^{n-1}).$$

Then for $i < j$ the (i, j) -element of $D^{-1}ND$ is $\nu_{ij}\eta^{j-i}$. Since for $i < j$ we have $\eta^{j-i} \rightarrow 0$ as $\eta \rightarrow 0$, there is an η such that $\|D^{-1}ND\|_\infty < \epsilon$. It follows that

$$\begin{aligned} \|D^{-1}U^H AUD\|_\infty &= \|\Lambda + D^{-1}ND\|_\infty \\ &\leq \|\Lambda\|_\infty + \|D^{-1}ND\|_\infty \\ &\leq \rho(A) + \epsilon. \end{aligned} \tag{2.7}$$

If we now define $\|\cdot\|_{A,\epsilon}$ by

$$\|B\|_{A,\epsilon} = \|D^{-1}U^H BUD\|_\infty,$$

then by Theorem 2.4 the norm $\|\cdot\|_{A,\epsilon}$ is consistent, and by (2.7) it satisfies (2.6).

If the dominant eigenvalues of A are nondefective, we can reduce A to the form

$$U^{-1}AU = \text{diag}(\Lambda_1, \Lambda_2 + N),$$

where Λ_1 is a diagonal matrix containing the dominant eigenvalues of A , Λ_2 is a diagonal matrix containing the remain eigenvalues of A , and N is strictly upper triangular. Specifically, by Theorem 1.19 we can reduce A to the form $\text{diag}(B_1, B_2)$, where B_1 contains the dominant eigenvalues of A and B_2 contains the others. Since B_1 is not defective, we can diagonalize it to get the matrix Λ_1 . We can then reduce B_2 to Schur form to get the matrix $\Lambda_2 + N$.

Since the diagonals of Λ_2 are strictly less than $\rho(A)$, we can choose a diagonal matrix $D_\eta = \text{diag}(I, \hat{D}_\eta)$ as above so that

$$\begin{aligned} \|D_\eta^{-1}\text{diag}(\Lambda_1, \Lambda_2 + N)D_\eta\|_\infty &= \max\{\|\Lambda_1\|_\infty, \|\hat{D}_\eta^{-1}(\Lambda_2 + N)\hat{D}_\eta\|_\infty\} \\ &= \|\Lambda_1\|_\infty \\ &= \rho(A). \end{aligned}$$

The norm $\|\cdot\|_A$ defined by

$$\|B\|_A = \|D^{-1}U^{-1}BUD\|_\infty$$

now does the job. ■

Theorem 2.8 shows that the spectral radius of a matrix can be approximated to arbitrary accuracy by some consistent norm. We will see that this norm may be highly skewed with respect to the usual coordinate system — see the discussion following Example (2.11). Nonetheless, it can be used to establish valuable theoretical results, as we shall show in the next subsection.

2.3. MATRIX POWERS

Many matrix processes and algorithms are driven by powers of matrices. For example, the errors in certain iterative algorithms for solving linear systems increase decrease as the powers of a matrix called the iteration matrix. It is therefore important to have criteria for when we observe a decrease. Equally important, we need to know the rate of decrease. For this reason, we now turn to the asymptotic behavior of the powers of a matrix.

Asymptotic bounds

The basic result is contained in the following theorem.

Theorem 2.9. *Let A be of order n and let $\epsilon > 0$ be given. Then for any norm $\|\cdot\|$ there are positive constants σ (depending on the norm) and $\tau_{A,\epsilon}$ (depending on A , ϵ , and the norm) such that*

$$\sigma\rho(A)^k \leq \|A^k\| \leq \tau_{A,\epsilon}(\rho(A) + \epsilon)^k. \quad (2.8)$$

If the dominant eigenvalues of A are nondefective, we may take $\epsilon = 0$.

Proof. We will first establish the lower bound. Let (λ, x) be a dominant eigenpair of A with $\|x\|_2 = 1$. Then

$$\|A^k x\|_2 = \|\lambda^k x\|_2 = \rho(A)^k.$$

It follows that $\|A^k\|_2 \geq \rho(A)^k$. By the equivalence of norms, there is a constant $\sigma > 0$ such that $\|B\| \geq \sigma\|B\|_2$ for all B . Hence

$$\|A^k\| \geq \sigma\|A^k\|_2 \geq \sigma\rho(A)^k.$$

For the upper bound, let $\|\cdot\|_{A,\epsilon}$ be a consistent norm such that $\|A\|_{A,\epsilon} \leq \rho(A) + \epsilon$. Then

$$\|A^k\|_{A,\epsilon} \leq \|A\|_{A,\epsilon}^k \leq (\rho(A) + \epsilon)^k.$$

Again by the equivalence of norms, there is a constant $\tau_{A,\epsilon} > 0$ such that $\|B\| \leq \tau_{A,\epsilon}\|B\|_{A,\epsilon}$ for all B . Then

$$\|A^k\| \leq \tau_{A,\epsilon}\|A^k\|_{A,\epsilon} \leq \tau_{A,\epsilon}(\rho(A) + \epsilon)^k.$$

If the dominant eigenvalues of A are nondefective, we can repeat the proof of the upper bound with a consistent norm $\|\cdot\|_A$ (whose existence is guaranteed by Theorem 2.8) for which $\|A\|_A = \rho(A)$. ■

The theorem shows that the powers of A grow or shrink essentially as $\rho(A)^k$. We have to add “essentially” here because if a dominant eigenvalue is defective, the bound must be in terms of $\rho(A) + \epsilon$, where ϵ can be arbitrarily small. It is instructive to look at a simple case of a defective dominant eigenvalue.

Example 2.10. The powers of the Jordan block $J_2(0.5)$ are

$$J_2(0.5)^k = \begin{pmatrix} 2^{-k} & k2^{-k+1} \\ 0 & 2^{-k} \end{pmatrix}.$$

Hence

$$\|J_2(0.5)^k\|_\infty = 2^{-k}(1 + 2k),$$

and

$$\frac{\|J_2(0.5)^{k+1}\|_\infty}{\|J_2(0.5)^k\|_\infty} = \frac{1}{2} \cdot \frac{3 + 2k}{1 + 2k}. \quad (2.9)$$

The ratios in (2.9) are called LOCAL CONVERGENCE RATIOS because they measure the amount by which the successor of a member of the sequence is reduced (or increased). For $k = 1$, the convergence ratio is $\frac{5}{6}$, so that the norms decrease from the outset. The local convergence ratios are never quite as small as the asymptotic ratio one-half, but they quickly approach it. For $k = 10$, e.g., the ratio is about 0.55.

A matrix whose powers converge to zero is said to be *convergent*. It follows immediately from Theorem 2.9 that:

A matrix is convergent if and only if its spectral radius is less than one.

Limitations of the bounds

One should not be lulled by the behavior of $J_2(0.5)^k$, whose norms decrease monotonically at a rate that quickly approaches 2^{-k} . Here is a more extreme example.

Example 2.11. The ∞ -norm of the powers of $J_2(0.99)$ are

$$\|J_2(0.99)^k\|_\infty = \left\| \begin{pmatrix} 0.99^k & 0.99^{k-1}k \\ 0 & 0.99^k \end{pmatrix} \right\|_\infty = 0.99^k(1 + k/0.99).$$

The upper graph in Figure 2.1 (called the hump) plots these norms against the power k . Instead of decreasing, the norms increase to a maximum of about 37 when $k = 99$. Thereafter they decrease, but they fall below their initial value of 1.99 only when $k = 563$. The lower plot shows the natural logarithms of the local convergence ratios, which eventually approximate the asymptotic value of $\log 0.99 \approx -0.01$. From these graphs we see that the powers of $J_2(0.99)$ not only take a long time to assume their asymptotic behavior, but they exhibit a hellish transient in the process.

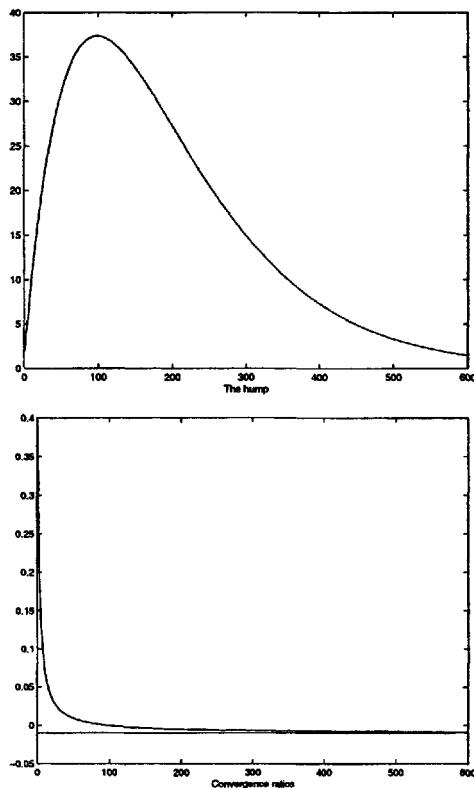


Figure 2.1: The hump and convergence ratios

Which example—2.10 or 2.11—is typical? The answer is: It depends. Many people will go through life powering matrices and never see anything as dramatic as $J_2(0.99)^k$. However, such examples do occur, and sometimes their effects are rather subtle. Be on guard.

The bound (2.8) gives no hint of the ill behavior of $J_2(0.99)$, and it is instructive to see why. If we choose $\epsilon = 0.001$, so that $\rho[J_2(0.99)] + \epsilon$ is near $\rho[J_2(0.99)]$, then the norm $\|\cdot\|_{A,\epsilon}$ given by Theorem 2.8 has the form

$$\|B\|_{A,\epsilon} = \left\| \begin{pmatrix} b_{11} & 0.001b_{12} \\ 1000b_{21} & b_{22} \end{pmatrix} \right\|_\infty.$$

The $(2, 2)$ -element of $J_2(0.99)^k$, which accounts for the hump, is damped by the factor 0.001, so that the norm remains small. Of course the norm of a general matrix would be unnaturally large, because the $(2, 1)$ -element is multiplied by 1000. But the $(2, 1)$ -element of $J_2(0.99)^k$ is zero and hence is not magnified by this factor.

There is a lesson to be learned from this. In matrix analysis one frequently introduces a norm designed to make a construction or a proof easy. Mathematically, this

is sound practice. But unless the norm bears a reasonable relation to the coordinate system one is working in (the usual unit vectors for Example 2.11), the results may be disappointing or misleading.

2.4. NOTES AND REFERENCES

Norms

For treatments of matrix and vector norms see the general references in §1.7 or in §I:1.4.1 of this series. The reader should be aware that some authors include consistency in the definition of a matrix norm. A matrix norm and a consistent vector norm are sometimes called *compatible* norms.

Norms and the spectral radius

In his *Solution of Equations and Systems of Equations* Ostrowski [194, 1960] attributes Theorem 2.7 to Frobenius, though he gives no reference. In the same book Ostrowski gives essentially Theorem 2.8, stating that it “is unpublished.”

Ostrowski’s proof is an elegant and amusing application of the Jordan form. Let Λ be the diagonal matrix of eigenvalues of A and let $X^{-1}(\epsilon^{-1}A)X = \epsilon^{-1}\Lambda + N$ be the Jordan form of $\epsilon^{-1}A$. Since the only nonzero elements of N are ones on its superdiagonal, we have

$$\|X^{-1}AX\|_\infty = \|\Lambda + \epsilon N\|_\infty \leq \rho(A) + \epsilon.$$

Matrix powers

For a detailed treatment of matrix powers — exact and computed — see [114, Ch. 17]. The existence of the hump illustrated in Figure 2.1 has lead to various attempts to bound $\max_k \|A^k\|$, which are described in [114, §17.1].

When computed in finite precision, the powers of a matrix can exhibit erratic behavior, and a nominally convergent matrix can even diverge. There is no one, simple explanation, but the gist of what is happening this. The computed j th column of A^k satisfies

$$a_j^{(k)} = (A + E_k)(A + E_{k-1}) \cdots (A + E_1)\mathbf{e}_j,$$

where $\|E_i\|/\|A\|$ is of order of the rounding unit. If A has a defective eigenvalue near one, the perturbations E_i may create eigenvalues of magnitude greater than one, in which case the asymptotic behavior will not be described by Theorem 2.9.

An important approach to this problem is the *pseudospectrum*

$$\Lambda_\epsilon = \{\lambda : \|A - \lambda I\| \leq \epsilon\}.$$

In many instances, the location of the pseudospectrum in the complex plane explains the behavior of the matrix powers. In the example of the last paragraph, the defective eigenvalue near one will, for sufficiently large ϵ , cause the pseudospectrum to extend outside the unit circle. However, the approach has the disadvantage that one must choose a suitable value for the free parameter ϵ — not an easy problem.

Although the pseudospectrum has been introduced independently several times, the current high state of the art is largely due to L. N. Trefethen [276, 277, 279]. For more see the Pseudospectra Gateway:

<http://web.comlab.ox.ac.uk/projects/pseudospectra/>

3. PERTURBATION THEORY

In this section we will be concerned with the sensitivity of eigenvalues and eigenvectors of a matrix to perturbations in the elements of the matrix. There are two reasons for studying this problem. First, the results are often of use in the design and analysis of algorithms. Second, many of the algorithms we will consider return approximate eigenpairs $(\tilde{\lambda}, \tilde{x})$ that satisfy $(A + E)\tilde{x} = \tilde{\lambda}\tilde{x}$, where $\|E\|/\|A\|$ is of the order of the rounding unit for the computer in question. A knowledge of the sensitivity of the eigenpair will enable us to predict the accuracy of the computed eigenvalue.

The perturbation theory of eigenvalues and eigenvectors is a very large subject, and we will only be able to skim the most useful results. In §3.1 we will consider the perturbation of individual eigenvalues, and in §3.2 we will consider the perturbation of eigenpairs. Most of the results in §3.2 generalize to eigenspace. We will treat these generalizations in Chapter 4.

3.1. THE PERTURBATION OF EIGENVALUES

This subsection is concerned with the problem of assessing the effects of a perturbation of a matrix on its eigenvalues. We will consider three topics. The first is a bound that establishes the continuity of the eigenvalues of a matrix. The second topic is Gershgorin's theorem, a useful tool for probing the sensitivity of individual eigenvalues. We conclude with results on the perturbation of the eigenvalues of Hermitian matrices and the singular values of general matrices.

The continuity of eigenvalues

A fundamental difficulty with the perturbation of eigenvalues is that they need not be differentiable functions of the elements of their matrix. For example, the eigenvalues of the matrix

$$\begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix}$$

are $1 \pm \sqrt{\epsilon}$, and this expression is not differentiable at $\epsilon = 0$. Nonetheless, the eigenvalues are continuous at $\epsilon = 0$. In fact, the eigenvalues of any matrix are continuous functions of the elements of the matrix, as the following theorem shows.

Theorem 3.1 (Elsner). Let A the (possibly multiple) eigenvalues be $\lambda_1, \dots, \lambda_n$. Let the eigenvalues of $\tilde{A} = A + E$ be $\tilde{\lambda}_1, \dots, \tilde{\lambda}_n$. Then there is a permutation j_1, \dots, j_n of the integers $1, \dots, n$ such that

$$|\tilde{\lambda}_{j_i} - \lambda_i| \leq 4(\|A\|_2 + \|\tilde{A}\|_2)^{1-\frac{1}{n}} \|E\|_2^{\frac{1}{n}}. \quad (3.1)$$

There are three points to be made about this theorem.

- The exponent $\frac{1}{n}$ in the factor $\|E\|_2^{\frac{1}{n}}$ is necessary. For example, the eigenvalues of

$$J_n(0) + \epsilon \mathbf{e}_n \mathbf{e}_1^T = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ \epsilon & 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \quad (3.2)$$

are

$$\tilde{\lambda}_i = \epsilon^{\frac{1}{n}} \omega^i,$$

where ω is the primitive n th root of unity. Thus a perturbation of order ϵ has the potential of introducing a perturbation of order $\epsilon^{\frac{1}{n}}$ in the eigenvalues of a matrix.

However, it is important to stress that this example is extreme — the eigenvalue in question is totally defective. Garden variety eigenvalues are more likely to suffer a perturbation that is linear in the error, though perhaps magnified by a large order constant. For this reason, the bound (3.1) is used more in theory than in practice.

- The theorem establishes the continuity of eigenvalues in a very strong sense. It is fairly easy to show that for every eigenvalue λ of A there must be an eigenvalue $\tilde{\lambda}$ of \tilde{A} satisfying the bound (3.1), and vice versa. However, this does not preclude the possibility of two eigenvalues of \tilde{A} attaching themselves to a simple eigenvalue of A and vice versa. Theorem 3.1, on the other hand, pairs up the eigenvalues of A and \tilde{A} in such a way that each pair satisfies the bound (3.1). In particular, to an eigenvalue λ of multiplicity m of A there correspond m eigenvalues of \tilde{A} , each satisfying the bound (3.1).
- The theorem has a useful implication for the behavior of the simple real eigenvalues of a real matrix under real perturbations. Let λ be such an eigenvalue and let E be small enough so that the right-hand side of (3.1) is less than half the distance between λ and its neighbors. Then there is exactly one eigenvalue $\tilde{\lambda}$ of \tilde{A} satisfying the bound (for otherwise there would be too few eigenvalues of \tilde{A} to pair up with the other eigenvalues of A). Moreover, $\tilde{\lambda}$ cannot be complex, since if it were its complex conjugate would also be an eigenvalue — a second eigenvalue that satisfies the bound. Hence:

The simple real eigenvalues of a real matrix remain real under sufficiently small real perturbations.

Gerschgorin theory

The problem with the bound in Theorem 3.1 is that it bounds the perturbation of all the eigenvalues of any matrix. It must therefore be large enough to accommodate the worst possible case, which makes it too large for everyday work. A cure for this problem is to derive bounds for individual eigenvalues. Gerschgorin's theorem provides a method for doing this.

Theorem 3.2 (Gerschgorin). *Let A be of order n and let the GERSCHGORIN DISKS of A be defined by*

$$\mathcal{G}_i \stackrel{\text{def}}{=} \{\mu : |\mu - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|\}.$$

Then

$$\Lambda(A) \subset \bigcup_{i=1}^n \mathcal{G}_i.$$

Moreover, if the union of k of the sets \mathcal{G}_i are disjoint from the others, then that union contains exactly k eigenvalues of A .

Proof. Let (λ, x) be an eigenpair of A . Let ξ_i be a component of x with maximum modulus, and assume without loss of generality that $\xi_i = 1$. From the i th row of the equation $Ax = \lambda x$ we have

$$a_{ii}\xi_i + \sum_{j \neq i} a_{ij}\xi_j = \lambda\xi_i,$$

or

$$\lambda - a_{ii} = \sum_{j \neq i} a_{ij}\xi_j.$$

Taking absolute values and remembering that $|\xi_j| \leq 1$, we find that

$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|,$$

which says that $\lambda \in \mathcal{G}_i$.

The statement about the disjoint union can be established by a continuity argument. Let $A = D + B$, where $D = \text{diag}(a_{11}, \dots, a_{nn})$. Let $\mathcal{G}_i(t)$ be the Gerschgorin disks for $A(t) = D + tB$. The eigenvalues of $A(0)$ are the diagonal elements a_{ii} , and they lie in the the (degenerate) disks $\mathcal{G}_i(0)$. As t proceeds from zero to one, the radii of the disks $\mathcal{G}_i(t)$ increase monotonically to the radii of the Gerschgorin disks of A .

Now suppose that, say, $\bigcup_{i=1}^k \mathcal{G}_{j_i}$ is disjoint from the other disks. Then for $0 \leq t \leq 1$, $\mathcal{U}(t) = \bigcup_{i=1}^k \mathcal{G}_{j_i}(t)$ is also disjoint from the other disks $\mathcal{G}_i(t)$. Initially the union

$\mathcal{U}(0)$ contains exactly k eigenvalues of $A(0)$. Moreover, as t increases, $\mathcal{U}(t)$ can neither lose nor acquire eigenvalues. For that would require an eigenvalue to jump between $\mathcal{U}(t)$ and the union of the other disks, which violates continuity. Hence, $\mathcal{U}(1) = \bigcup_{i=1}^k \mathcal{G}_i$ contains exactly k eigenvalues of A . ■

Gershgorin's theorem is a general-purpose tool that has a variety of uses. In perturbation theory, it is especially useful in obtaining bounds on clusters of eigenvalues.

Example 3.3. Consider the matrix

$$A = \begin{pmatrix} 1.0000e+00 & 2.9441e-04 & -6.9178e-04 & -1.4410e-03 \\ 5.9281e-05 & 2.0000e+00 & 8.5800e-04 & 5.7115e-04 \\ -9.5648e-05 & 7.1432e-04 & 2.0000e+00 & -3.9989e-04 \\ -8.3235e-04 & 1.6236e-03 & -1.5937e-03 & 2.0000e+00 \end{pmatrix}.$$

The radii of the Gershgorin disks for this matrix are

$$2.4e-03, \quad 1.5e-03, \quad 1.2e-03, \quad 4.0e-03.$$

The first disk is disjoint from the others, from which we conclude that A has an eigenvalue λ_1 that satisfies $|\lambda_1 - 1| \leq 2.4 \cdot 10^{-3}$. The remaining three disks are centered at 2 and are not disjoint. Hence, A has three eigenvalues λ_2, λ_3 , and λ_4 that satisfy

$$|\lambda_i - 2| \leq 4.0 \cdot 10^{-3}, \quad i = 2, 3, 4. \tag{3.3}$$

At first glance this would appear to be a perfectly satisfactory result. If A is regarded as a perturbation of $\text{diag}(1, 2, 2, 2)$, then the above examples show that the eigenvalues of A lie near its diagonal elements, and their deviation is approximately the size of the error. However, the actual deviations are

$$1.3e-06, \quad 1.7e-03, \quad -7.4e-04, \quad -9.6e-04.$$

Thus the Gershgorin disks locate the eigenvalues near two fairly well, but they badly overestimate the deviation of the eigenvalue near one.

We can remedy this defect by the *method of diagonal similarities*, which is illustrated in the following example.

Example 3.4. Let A be the matrix of Example 3.3, and let

$$D = \text{diag}(10^{-3}, 1, 1, 1).$$

Then

$$DAD^{-1} = \begin{pmatrix} 1.0000e+00 & 2.9441e-07 & -6.9178e-07 & -1.4410e-06 \\ 5.9281e-02 & 2.0000e+00 & 8.5800e-04 & 5.7115e-04 \\ -9.5648e-02 & 7.1432e-04 & 2.0000e+00 & -3.9989e-04 \\ -8.3235e-01 & 1.6236e-03 & -1.5937e-03 & 2.0000e+00 \end{pmatrix}$$

and the radii of the Gerschgorin disks are

$$2.4e-06, \quad 6.1e-02, \quad 9.7e-02, \quad 8.4e-01.$$

The similarity transformation DAD^{-1} has reduced the radius of the first disk while increasing the radii of the others. Nonetheless, the first disk remains disjoint from the others. Hence we know that A , whose eigenvalues are the same as DAD^{-1} , has an eigenvalue within $2.4 \cdot 10^{-6}$ of one. On the other hand, from (3.3) we know that the remaining eigenvalues are within $4.0 \cdot 10^{-3}$ of two.

There are two comments to be made about this example.

- Although we have proceeded informally, choosing the matrix D by inspection, it is possible to formalize the method of diagonal similarities. For more, see the notes and references.
- The matrix in Example 3.3 is an off-diagonal perturbation of a diagonal matrix. What we have shown — at least for this matrix — is that if a diagonal element is well separated from the other diagonal elements then an off-diagonal perturbation moves the eigenvalue by a small amount that is generally proportional to the square of the perturbation.

Hermitian matrices

In this subsection we will state without proof some useful results on the eigenvalues of Hermitian matrices. The first result is a far-reaching characterization of the eigenvalues of a Hermitian matrix.

Theorem 3.5 (Fischer). *Let the Hermitian matrix A have eigenvalues*

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

Then

$$\lambda_i = \min_{\dim(\mathcal{W})=n-i+1} \max_{\substack{w \in \mathcal{W} \\ \|w\|_2=1}} w^H A w \quad (3.4)$$

and

$$\lambda_i = \max_{\dim(\mathcal{W})=i} \min_{\substack{w \in \mathcal{W} \\ \|w\|_2=1}} w^H A w. \quad (3.5)$$

The formulas (3.4) and (3.5) are called respectively the min-max and the max-min characterizations. They have far-reaching consequences. A trivial but useful consequence is that

$$\|w\|_2 = 1 \implies \lambda_n \leq w^H A w \leq \lambda_1.$$

Another important consequence is the following interlacing theorem.

Theorem 3.6. Let A be Hermitian of order n with eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

Let $U \in \mathbb{C}^{n \times m}$ be orthonormal and let the eigenvalue of $U^H A U$ be

$$\mu_1 \geq \mu_2 \geq \cdots \geq \mu_m.$$

Then

$$\lambda_i \geq \mu_i \geq \lambda_{n-m+i}, \quad i = 1, \dots, m. \quad (3.6)$$

The theorem is called the interlacing theorem because when $m = n-1$, the eigenvalues λ_i and μ_i satisfy

$$\lambda_1 \geq \mu_1 \geq \lambda_2 \geq \mu_2 \geq \cdots \geq \lambda_{n-1} \geq \mu_{n-1} \geq \lambda_n;$$

i.e., they interlace. If U consists of m columns of the identity matrix, then $U^H A U$ is a principal submatrix of A of order m . Hence we have the following corollary.

Corollary 3.7. If B is a principal submatrix of A of order m , then the eigenvalues μ_i of B satisfy (3.6).

The final consequence of Fischer's theorem is the following theorem on the perturbation of eigenvalues.

Theorem 3.8. Let A and $\tilde{A} = A + E$ be Hermitian. Let the eigenvalues of A , \tilde{A} , and E be

$$\lambda_1 \geq \cdots \geq \lambda_n, \quad \tilde{\lambda}_1 \geq \cdots \geq \tilde{\lambda}_n, \quad \text{and} \quad \epsilon_1 \geq \cdots \geq \epsilon_n.$$

Then

$$\lambda_i + \epsilon_n \leq \tilde{\lambda}_i \leq \lambda_i + \epsilon_1, \quad i = 1, \dots, n. \quad (3.7)$$

Consequently,

$$|\tilde{\lambda}_i - \lambda_i| \leq \|E\|_2, \quad i = 1, \dots, n. \quad (3.8)$$

It is noteworthy that like Theorem 3.1 this theorem is about pairs of eigenvalues. The inequality (3.7) gives an interval in which the i th eigenvalue must lie in terms of the smallest and largest eigenvalues of E . However, there is a difference. In Theorem 3.1 the pairing is not explicitly given. In Theorem 3.8, the pairing is explicitly determined by the order of the eigenvalues.

In the terminology of perturbation theory, (3.8) says that the eigenvalues of a Hermitian matrix are perfectly conditioned—that is, a perturbation in the matrix makes a

perturbation that is no larger in the eigenvalues. The perfect condition of the eigenvalues of a Hermitian matrix does not, unfortunately, imply that all the eigenvalues are determined to high *relative* accuracy. In fact, the relative error in λ_i is

$$\frac{|\tilde{\lambda}_i - \lambda_i|}{|\lambda_i|} \leq \frac{\|E\|_2}{|\lambda_i|}. \quad (3.9)$$

A rule of thumb says that if the relative error in a quantity is about 10^{-k} , then the quantity is accurate to about k decimal digits. Thus (3.9) implies that the smaller the eigenvalue the fewer the accurate digits, as the following example shows.

Example 3.9. The matrix

$$A = \begin{pmatrix} 7.7623e-01 & 3.7696e-01 & -1.5849e-01 & 8.2979e-03 \\ 3.7696e-01 & 2.1945e-01 & -8.2625e-02 & -4.2335e-02 \\ -1.5849e-01 & -8.2625e-02 & 4.1530e-02 & 1.4781e-02 \\ 8.2979e-03 & -4.2335e-02 & 1.4781e-02 & 7.3796e-02 \end{pmatrix}$$

has eigenvalues $1, 10^{-1}, 10^{-2}$, and 10^{-3} . (We have reported only the first five figures of the element of A .) If we perturb A by a matrix E of 2-norm 10^{-4} we get a matrix whose eigenvalues are

$$1.0000e+00, \quad 1.0002e-01, \quad 9.9839e-03, \quad 1.0180e-03.$$

The relative errors in these eigenvalues are

$$2.8e-07, \quad 1.9e-04, \quad 1.6e-03, \quad 1.8e-02.$$

It is seen that the relative errors become progressively larger as the eigenvalues get smaller.

The example not only illustrates the relative sensitivity of the smaller eigenvalues, but it shows the limitations of bounds based on norms. The absolute errors in the eigenvalues,

$$2.8e-07, \quad 1.9e-05, \quad 1.6e-05, \quad 1.8e-05,$$

are in general smaller than the bound would lead one to expect.

Since the Frobenius norm of a matrix is greater than or equal to its 2-norm, Theorem 3.8 holds for the Frobenius norm. However, a stronger result is also true.

Theorem 3.10 (Hoffman–Wielandt). *In the notation of Theorem 3.8,*

$$\sqrt{\sum_i (\tilde{\lambda}_i - \lambda_i)^2} \leq \|E\|_F.$$

3.2. SIMPLE EIGENPAIRS

Up to now we have been concerned with eigenvalues. In this section we will examine the sensitivity of eigenvectors to perturbations in the matrix. It turns out that the perturbation theory for an eigenvector goes hand in hand with that of its eigenvalue, and we will actually treat the perturbation of eigenpairs. We will restrict ourselves to a simple eigenpair (λ, x) . This insures that (up to a scalar factor) we have a unique eigenvector to study.

Left and right eigenvectors

Let (λ, x) be a simple eigenpair of A . By Theorem 1.18 there is a nonsingular matrix $(x \ X)$ with inverse $(y \ Y^H)$ such that

$$\begin{pmatrix} y^H \\ Y^H \end{pmatrix} A(x \ X) = \begin{pmatrix} \lambda & 0 \\ 0 & M \end{pmatrix}. \quad (3.10)$$

It is easily seen by writing

$$\begin{pmatrix} y^H \\ Y^H \end{pmatrix} A = \begin{pmatrix} \lambda & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} y^H \\ Y^H \end{pmatrix}$$

that $y^H A = \lambda y^H$ and $Y^H A = M Y^H$. In particular, y is the left eigenvector of A corresponding to λ .

The angle between x and y will play an important role in the perturbation theory to follow. From the Cauchy inequality we know that

$$|x^H y| = \gamma \|x\|_2 \|y\|_2,$$

where $\gamma \leq 1$ with equality if and only if x and y are linearly dependent. In \mathbb{R}^2 and \mathbb{R}^3 it is easy to see that the constant γ is the cosine of the angle between the lines subtended by x and y . In general we define the angle between nonzero vectors x and y by

$$\angle(x, y) \stackrel{\text{def}}{=} \cos^{-1} \frac{|y^H x|}{\|x\|_2 \|y\|_2}. \quad (3.11)$$

Since in (3.10) above $(x \ X)^{-1} = (y \ Y^H)^H$, we have $y^H x = 1$. It follows that:

If (λ, x) is a simple eigenpair of A , then there is a left eigenvector y of λ such that $y^H x = 1$. In this case

$$\sec \angle(x, y) = \|x\|_2 \|y\|_2.$$

A corollary of this assertion is that if x is a simple eigenvector then its left eigenvector y is not orthogonal to x .

First-order perturbation expansions

We now turn to the perturbation theory of simple eigenpairs. Specifically, we are given a simple eigenpair (λ, x) of A , and a small perturbation $\tilde{A} = A + E$. It can be shown (Theorem 3.11 below) that if E is small enough, then there is a simple eigenpair $(\tilde{\lambda}, \tilde{x})$ that approaches (λ, x) as E approaches zero. Before establishing that fact, we will first address the problem of approximating the perturbed eigenpair.

We will suppose that A has been block-diagonalized as in (3.10). The first order of business is to find suitable representations of $\tilde{\lambda}$ and \tilde{x} . Since $\tilde{\lambda}$ is a scalar, we can write

$$\tilde{\lambda} = \lambda + \varphi,$$

where φ is to be determined.

The problem of representing \tilde{x} is complicated by the fact that \tilde{x} is determined only up to a scalar factor. Thus to get a unique representation we must impose a normalization on \tilde{x} . Here we will require that $y^H \tilde{x} = 1$. This is easily seen to be equivalent to representing \tilde{x} in the form

$$\tilde{x} = x + Xp,$$

where p is to be determined.

We must now set up an equation from which φ and p may be determined. The natural equation is $A\tilde{x} = \tilde{\lambda}x$, or

$$(A + E)(x + Xp) = (\lambda + \varphi)(x + Xp).$$

Since $Ax = \lambda x$, we have

$$AXp + Ex + EXp = \lambda Xp + \varphi x + \varphi Xp.$$

Unfortunately this equation is nonlinear in the parameters φ and p . The key to a first-order analysis is to note that since φ and p approach zero along with E , products of these terms will become negligible when E is small enough. Ignoring such products, we get the linearized equation

$$AXp + Ex \cong \lambda Xp + \varphi x. \quad (3.13)$$

We must now solve the linearized equation for φ and p . To solve for φ , multiply by y^H . Since $y^H X = 0$, we have $y^H AXp = \lambda y^H Xp = 0$ and $y^H Xp = 0$. Since $y^H x = 1$, we have

$$\varphi \cong y^H Ex \equiv \hat{\varphi}.$$

To solve for p multiply (3.13) by Y^H , and use the facts that

$$Y^H A = M Y^H, \quad Y^H X = I, \quad \text{and} \quad Y^H x = 0$$

to get $Mp + Y^HEx \cong \lambda p$. Since λ is a simple eigenvalue, the eigenvalues of M are not equal to λ . Hence $\lambda I - M$ is nonsingular and

$$p \cong (\lambda I - M)^{-1}Y^HEx \equiv \hat{p}. \quad (3.14)$$

In summary:

The first-order expansion of the eigenpair (λ, x) under the perturbation $A + E$ of A is

$$(\tilde{\lambda}, \tilde{x}) \cong (\lambda + y^HEx, x + X(\lambda I - M)^{-1}Y^HEx).$$

Rigorous bounds

We have derived approximations to the perturbed eigenpair. However, we have not established that a perturbed pair exists to be approximated; nor have we shown how good the approximations actually are. The following theorem, which we give without proof, fills in these gaps. For more, see the notes and references.

Theorem 3.11. *Let (λ, x) be a simple eigenpair of A . Let $(x \ X)$ be a nonsingular matrix such that*

$$\begin{pmatrix} y^H \\ Y^H \end{pmatrix} A(x \ X) = \begin{pmatrix} \lambda & 0 \\ 0 & M \end{pmatrix}, \quad (3.15)$$

where $(y \ Y)^H = (x \ X)^{-1}$. Let $\tilde{A} = A + E$ and set

$$\begin{pmatrix} y^H \\ Y^H \end{pmatrix} E(x \ X) = \begin{pmatrix} \varphi_{11} & f_{12}^H \\ f_{21} & F_{22} \end{pmatrix}.$$

Let $\|\cdot\|$ be a consistent norm and define

$$\text{sep}(\lambda, M) = \|(\lambda I - M)^{-1}\|^{-1}. \quad (3.16)$$

If

$$4\|f_{21}\|\|f_{12}^H\| < (\text{sep}(\lambda, M) - |\varphi_{11}| - \|F_{22}\|)^2, \quad (3.17)$$

then there is a scalar φ and a vector p such that

$$(\tilde{\lambda}, \tilde{x}) = (\lambda + \varphi, x + Xp)$$

is an eigenpair of \tilde{A} with

$$\tilde{\lambda} \notin \Lambda(M).$$

Moreover,

$$\|p\| < \frac{2\|f_{21}\|}{\text{sep}(\lambda, M) - |\varphi_{11}| - \|F_{22}\|}, \quad (3.18)$$

$$\|p - (\lambda I - M)^{-1}f_{21}\| < \frac{2\|p\|^2\|f_{12}^H\|}{\text{sep}(\lambda, M) - |\varphi_{11}| - \|F_{22}\|}, \quad (3.19)$$

and

$$\|\varphi - y^HEx\| \leq \|p\|\|f_{12}\|. \quad (3.20)$$

There are many things to be said about this theorem, enough to give a subsubsection to each.

Qualitative statements

The bounds in Theorem 3.11 say very specific things about how a simple eigenpair behaves under perturbation. But to orient ourselves, we will first describe the qualitative behavior. Specifically, we have that $|\varphi_{11}|$, $\|f_{12}\|$, $\|f_{21}^H\|$, and $\|F\|_2$ are all $O(\|E\|)$. Moreover from (3.18) it follows that $\|p\|$ is also $O(\|E\|)$.

Turning first to the eigenvalues, since $\varphi = \tilde{\lambda} - \lambda$, it follows from (3.20) that

$$|\tilde{\lambda} - \lambda| = O(\|E\|).$$

In other words, the simple eigenvalues are not only continuous, but they do not exhibit the $O(\|E\|^{\frac{1}{k}})$ behavior that we have observed in defective matrices [see (3.2)].

Again by (3.20) our first-order approximation y^HEx to φ is accurate up to terms of $O(\|E\|^2)$. Since $\lambda + y^HEx = y^H\tilde{A}x$, we can write

$$|\tilde{\lambda} - y^H\tilde{A}x| = O(\|E\|^2). \quad (3.21)$$

The quantity $y^H\tilde{A}x$ in (3.21) is called a *Rayleigh quotient*. Generalizations of this scalar quantity will play an important role in algorithms for large eigenproblems.

Equation (3.20) implies that λ is a differentiable function of the elements of its matrix. For if we take $E = \epsilon e_i e_j$, so that E represents a perturbation of size ϵ in the (i, j) -element of A , then

$$\tilde{\lambda} - \lambda = \epsilon \bar{y}_j x_j + O(\|E\|^2).$$

From this it follows that $\partial\lambda/\partial a_{ij}$ is $\bar{y}_j x_j$. In matrix terms:

If (λ, x) is a simple eigenpair of A and y is the corresponding left eigenvector, normalized so that $y^Hx = 1$, then

(3.22)

$$\left(\frac{\partial \lambda}{\partial a_{ij}} \right) = xy^H.$$

Turning now to the eigenvector x , since $\tilde{x} - x = Xp$, we have

$$\|\tilde{x} - x\| = O(\|E\|),$$

so that the eigenvector is also continuous. Moreover, from (3.19) it follows that

$$\|\tilde{x} - [x + X(\lambda I - M)^{-1}f_{21}]\| = O(\|E\|^2).$$

Thus we can expect $x + X(\lambda I - M)^{-1}f_{21}$ to be a very good approximation to \tilde{x} .

The condition of an eigenvalue

Turning now to more informative bounds, we begin with the eigenvalue λ . Since $\tilde{\lambda} = \lambda + y^HEx + (\varphi - y^HEx)$, we have from (3.20)

$$|\tilde{\lambda} - \lambda| \leq \|x\|_2\|y\|_2\|E\|_2 + O(\|E\|^2).$$

By (3.12), we know that $\|x\|_2\|y\|_2 = \sec \angle(x, y)$. Hence

$$|\tilde{\lambda} - \lambda| \leq \sec \angle(x, y)\|E\|_2 + O(\|E\|^2). \quad (3.23)$$

Ignoring the $O(\|E\|^2)$ term in (3.23), we see that the secant of the angle between x and y is a constant that mediates the effect of the error E on the eigenvalue λ . Such numbers are called *condition numbers*. If a condition number for a quantity is large, the quantity is said to be *ill conditioned*. If it is small, the quantity is said to be *well conditioned*. (For a general discussion of condition numbers see §I:2.4.) Thus:

The condition number of a simple eigenvalue is the secant of the angle between its left and right eigenvectors. (3.24)

Two caveats apply to condition numbers in general. First, they are useful only to the extent that the bound that generates them is sharp. In our case, the bound on λ was derived by taking norms in a sharp approximation to the perturbed eigenvalue $\tilde{\lambda}$. Hence the bound will generally be in the ballpark, and the condition number will be an accurate reflection of the sensitivity λ . Keep in mind, however, that it is easy to contrive large perturbations that leave λ unchanged.

Second, the condition number changes with the norm. In particular, if we had not chosen the 2-norm, we would not have ended up with as elegant a condition number as the secant of the angle between the left and the right eigenvectors. On the other hand, for most norms in common use the secant gives a reasonable idea of the sensitivity.

The condition of an eigenvector

Theorem 3.11 is useful in establishing the existence of a perturbed eigenpair, and, as we have just seen, it provides a meaningful condition number for the perturbed eigenvalue. But it does not directly provide a condition number for the perturbed eigenvector. The reason is that the vector p can be large, even when the perturbation in the directions between x and \tilde{x} is small.

To see this, let S be nonsingular. Then we can replace X in the block diagonal reduction (3.15) by XS and Y by YS^{-H} , so that the block diagonal form becomes $\text{diag}(\lambda, S^{-1}MS)$. It then follows that the approximation \hat{p} to p in (3.14) becomes $S^{-1}\hat{p}$. By choosing S appropriately, we can make \hat{p} as large as we like compared to $\|E\|$.

The cure for this problem is to bound the angle between x and \tilde{x} . We begin by choosing a distinguished basis for the reduction (3.15) to block diagonal form. Specifically, we will, as above, replace X by XS and Y by YS^{-H} , with S chosen so that the new Y is orthonormal. For the rest of this subsection we will assume that this change has been made.

Although we defined the angle between two vectors in terms of the cosine, we will actually calculate the sine of the angle. The following lemma shows how do do this.

Lemma 3.12. *Let x and \tilde{x} be nonzero, and let Y be an orthonormal basis for $\mathcal{R}(x)^\perp$. Then*

$$\sin \angle(x, \tilde{x}) = \frac{\|Y^H \tilde{x}\|_2}{\|\tilde{x}\|_2}.$$

Proof. Without loss of generality assume that $\|x\|_2 = \|\tilde{x}\|_2 = 1$. Then the matrix $(x \ Y)$ is unitary. From the unitary invariance of $\|\cdot\|_2$, we have

$$1 = \left\| \begin{pmatrix} x^H \\ Y^H \end{pmatrix} \tilde{x} \right\|_2^2 = |x^H \tilde{x}|^2 + \|Y^H \tilde{x}\|_2^2.$$

But $|x^H \tilde{x}| = \cos(x, \tilde{x})$. Hence

$$\|Y^H \tilde{x}\|_2^2 = 1 - \cos^2 \angle(x, \tilde{x}) = \sin^2 \angle(x, \tilde{x}). \quad \blacksquare$$

We are now in a position to bound the sine of the angle between x and \tilde{x} .

Theorem 3.13. *In Theorem 3.11 assume that Y is orthonormal, and that the condition (3.17) is satisfied in the 2-norm. Then*

$$\sin \angle(x, \tilde{x}) \leq \frac{\|E\|_2}{\text{sep}(\tilde{\lambda}, M)}. \quad (3.25)$$

Proof. We have $(A + E)\tilde{x} = \tilde{\lambda}\tilde{x}$ or

$$\tilde{\lambda}\tilde{x} - A\tilde{x} = E\tilde{x}.$$

Multiplying this relation by Y^H and recalling that $Y^H A = MY^H$, we have

$$(\tilde{\lambda}I - M)Y^H \tilde{x} = Y^H E\tilde{x}.$$

Since $\tilde{\lambda} \notin M$,

$$\|Y^H \tilde{x}\|_2 \leq \|(\tilde{\lambda}I - M)^{-1}\|_2 \|E\|_2 \|x\|_2,$$

and by Lemma 3.12 and the definition (3.16)

$$\sin \angle(x, \tilde{x}) = \frac{\|Y^H \tilde{x}\|_2}{\|\tilde{x}\|_2} \leq \frac{\|E\|_2}{\text{sep}(\tilde{\lambda}, M)}. \quad \blacksquare$$

From Theorem 3.11 it follows that $\tilde{\lambda} \rightarrow \lambda$ as $E \rightarrow 0$. Hence we may rewrite (3.25) in the form

$$\sin \angle(x, \tilde{x}) \lesssim \text{sep}^{-1}(\lambda, M) \|E\|_2.$$

Consequently, $\text{sep}^{-1}(\lambda, M)$ is a condition number for the eigenvector x . We will now consider some of the properties of the functions sep .

Properties of sep

The first and most natural question to ask of sep is how it got its name. The answer is that it is a lower bound on the separation of λ from the spectra of M . Specifically, we have the following theorem.

Theorem 3.14. *In any consistent norm,*

$$\text{sep}(\lambda, M) \leq \min_{\mu \in \Lambda(M)} |\lambda - \mu|.$$

Proof. By Theorem 2.7

$$\text{sep}^{-1}(\lambda, M) = \|(\lambda I - M)^{-1}\| \geq \rho[(\lambda I - M)^{-1}] = \max_{\mu \in \Lambda(M)} |\lambda - \mu|^{-1}. \quad (3.26)$$

The result now follows on inverting this relation. ■

Thus $\text{sep}(\lambda, M)$ is a lower bound on the separation of λ and the eigenvalues of M . Otherwise put if an eigenvalue of M is near λ we can expect the eigenvector x to be sensitive to perturbations in A . Actually, this is just common sense. For if an eigenvalue of M is near λ , then a small perturbation of A could make them equal and turn the one-dimensional space spanned by x into a two-dimensional space.

Unfortunately, $\text{sep}(\lambda, M)$ can be far smaller than the separation of λ from the spectrum of M , as the following example shows.

Example 3.15. Let $\lambda = 0$ and let $M = W_m$ be the $m \times m$ upper triangular matrix

$$W_m = \begin{pmatrix} 1 & -1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & -1 & -1 & \cdots & -1 \\ 0 & 0 & 1 & -1 & \cdots & -1 \\ 0 & 0 & 0 & 1 & \cdots & -1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}. \quad (3.27)$$

Since all the eigenvalues of W_m are one, the distance between λ and the spectrum of W_m is one. On the other hand, in the ∞ -norm

$$\text{sep}(0, W_m) = \|(0I - W_m)^{-1}\|_{\infty}^{-1} = \|W_m^{-1}\|_{\infty}^{-1}.$$

But

$$W_m^{-1} = \begin{pmatrix} 1 & 1 & 2 & 4 & \cdots & 2^{m-2} \\ 0 & 1 & 1 & 2 & \cdots & 2^{m-3} \\ 0 & 0 & 1 & 1 & \cdots & 2^{m-4} \\ 0 & 0 & 0 & 1 & \cdots & 2^{m-4} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix},$$

so that $\|W_m^{-1}\|_{\infty} = 2^{m-1}$. Hence

$$\text{sep}(0, W_m) = 2 \cdot 2^{-m},$$

which, with increasing m , goes quickly to zero in spite of the excellent separation of $\lambda = 0$ from the eigenvalues of W_m .

This example suggests the possibility that an eigenvector whose eigenvalue is well separated from its companions can still be sensitive to perturbations in its matrix. It is an instructive exercise to use W_m to construct an example of this phenomenon.

Hermitian matrices

If A is Hermitian then the right eigenvector x corresponding to the eigenvalue λ is also the left eigenvector y . It follows that the angle between x and y is zero, and $\sec \angle(x, y) = 1$ — the smallest possible value. Thus the simple eigenvalues of a Hermitian matrix are optimally conditioned. This result, of course, is weaker than (3.8), which shows that any eigenvalue of a Hermitian matrix, whatever its multiplicity, is perfectly conditioned.

The simple eigenvectors of a Hermitian matrix also behave nicely, as a consequence of the following theorem, whose proof is left as an exercise. [Hint: See (3.26).]

Theorem 3.16. *Let λ be real and M Hermitian. Then in the 2-norm,*

$$\text{sep}(\lambda, M) = \min_{\mu \in \Lambda(M)} |\lambda - \mu|.$$

Now in the decomposition (3.15) of Theorem 3.11, we may take $X = Y$, so that M is Hermitian. It follows that if x is an eigenvector of A and \tilde{x} is the corresponding eigenvector of $\tilde{A} = A + E$, then

$$\sin \angle(x, \tilde{x}) \lesssim \frac{\|E\|_2}{\min_{\mu \in \Lambda(M)} |\lambda - \mu|}. \quad (3.28)$$

Thus for Hermitian matrices the physical separation of eigenvalues — as distinguished from the lower bound sep — indicates the condition of its eigenvectors.

3.3. NOTES AND REFERENCES

General references

Most texts on numerical linear algebra contain at least some material on the perturbation of eigenvalues and eigenvectors (especially [121] and [300]). Books whose titles include the words “matrix analysis” or “matrix inequalities” usually contain material on perturbation theory — e.g., [18], [20], [39] [118], [119], [168], [177].

There are several books devoted more or less exclusively to perturbation theory. Kato’s *Perturbation Theory for Linear Operators* [146] presents the theory as seen by a functional analyst (although it includes a valuable treatment of the finite-dimensional case). Bhatia [22] treats the perturbation of eigenvalues with special emphasis on Hermitian and normal matrices. In his subsequent *Matrix Analysis* [23] he treats eigenspaces of Hermitian matrices. Both books are a valuable source of references. The approach taken in this section is along the line of Stewart and Sun’s *Matrix Perturbation Theory* [269].

Continuity of eigenvalues

For an extensive survey of general results on the perturbation of eigenvalues of non-normal matrices, see Bhatia’s *Matrix Analysis* [23, Ch. VIII]. Theorem 3.1 is due to Elsner [69, 70] as improved by Bhatia, Elsner, and Krause [24].

Gershgorin theory

Gershgorin’s theorem [82, 1931] is a consequence of the fact that a strictly diagonally dominant matrix is nonsingular. This fact has been repeatedly rediscovered, leading Olga Taussky to write a survey entitled “A Recurring Theorem on Determinants” [274]. Gershgorin also showed that an isolated union of k disks contains exactly k eigenvalues and introduced the use of diagonal similarities to reduce the radii of the disks. The technique is quite flexible, as Wilkinson has demonstrated in his *Algebraic Eigenvalue Problem* [300]. For a formalization of the procedure see [175, 285].

Eigenvalues of Hermitian matrices

Much of the material presented here can be found in §I:1.4.4 of this series. Also see [23, 118, 119, 269] for further references and historical notes.

Simple eigenpairs

The results in this subsection were, for the most part, developed for eigenspaces, which will be treated in §2, Chapter 4.

Left and right eigenvectors

The fact that left and right eigenvectors corresponding to a simple eigenvalue cannot be orthogonal is well known and can be easily deduced from a Schur decomposition.

The converse is not true—a multiple eigenvalue can fail to have nonorthogonal left and right eigenvectors, although for this to happen the eigenvalue must be defective. An analytic version of the result, also deducible from a Schur decomposition, is that if the left and right eigenvectors of a matrix are nearly orthogonal, then the matrix must be near one with a multiple eigenvalue [303].

First-order perturbation expansions

First-order perturbation expansions have long been used in the physical sciences to linearize nonlinear differential equations. A classroom example is the behavior of the pendulum under small oscillations. They have also been used to generate iterations for solving nonlinear equations; Newton's method is the prototypical example.

In matrix analysis first-order expansions are useful in three ways. First, the approximations they furnish are often used in deriving and analyzing matrix algorithms. Second, formulas for derivatives can often be read off from a perturbation expansion [the assertion (3.22) is an example]. Third, in perturbation theory a first-order expansion can yield very sharp asymptotic bounds. In fact, a first-order expansion is often the best way to get a handle on the perturbation theory of a new object.

Rigorous bounds

Theorem 3.11 is a consequence of more general theorems due to the author [250, 252]. We will return to these bounds when we treat eigenspaces in Chapter 4.

Condition numbers

Wilkinson [300, pp. 68–69] introduces the quantity $s = y^H x$, where x and y are normalized right and left eigenvectors corresponding to a simple eigenvalue λ , and on the basis of a first-order perturbation analysis notes that the sensitivity of λ is determined by $1/|s|$. $1/|s|$ is, of course, the secant of the angle between x and y .

If x and y are normalized so that $y^H x = 1$, the matrix xy^H is the *spectral projector corresponding to λ* . It is so called because $(xy^H)z$ is the projection of z on $\mathcal{R}(x)$ along $\mathcal{R}(y)^\perp$. Since $\|xy^H\|_2 = \|x\|_2\|y\|_2 = \sec \angle(x, y)$, the condition of a simple eigenvalue is the norm of its spectral projection. This result generalizes to representations of a matrix on an eigenspace [see (2.26), Chapter 4].

Theorem 3.13 is a generalization to non-Hermitian matrices of the $\sin \theta$ theorem of Davis and Kahan [54]. If Y is not orthonormal, the bound becomes

$$\sin \angle(x, \tilde{x}) \leq \kappa(Y) \frac{\|E\|_2}{\text{sep}(\tilde{\lambda}, M)},$$

where $\kappa(Y) = \|Y\|_2\|(Y^H Y)^{-1}Y^H\|$ (see, e.g., [126]).

Sep

The justification for the introduction of *sep* to replace the actual separation of eigenvalues in the complex plane is the simplicity of bounds it produces. Its drawback is

that sep is by no means a simple quantity, as Example 3.15 shows. On the other hand it has the nice property that

$$|\text{sep}(\lambda + \epsilon, B + F) - \text{sep}(\lambda, B)| \leq |\epsilon| + \|F\|,$$

where $\|\cdot\|$ is the norm used to define sep . Thus, a small perturbation in the arguments of sep makes an equally small perturbation in its value.

Relative and structured perturbation theory

Example 3.9 suggests that the small eigenvalues of a symmetric matrix are not well conditioned in a relative sense under perturbations of the matrix. This is generally true of normwise perturbation; however, if the perturbations are structured, small eigenvalues may be well conditioned. Consider, for example, the matrix

$$A = \begin{pmatrix} 2 & 10^{-3} \\ 10^{-3} & 10^{-6} \end{pmatrix},$$

whose eigenvalues are (to five figures) 2.0000 and $5.0000 \cdot 10^{-7}$. If we replace A by

$$\tilde{A} = \begin{pmatrix} 1.9995 & 1.0003 \cdot 10^{-3} \\ 1.0003 \cdot 10^{-3} & 10^{-6} \end{pmatrix},$$

we get eigenvalues 1.9995 and $4.9957 \cdot 10^{-7}$. In other words, the smallest eigenvalue is insensitive to small relative perturbations in the elements of A , even though the perturbation in the $(1, 1)$ -element is larger than the eigenvalue itself. This phenomenon is unexplained by our perturbation theory.

Over the last two decades researchers have begun to investigate these and related problems. The subject is still in flux, and it is impossible to survey it here. But we can discern some trends.

1. Relative perturbation of eigenvalues: Determine the relative error in eigenvalues resulting from perturbations in the matrix. Generally, the matrix and its perturbations must have some special structure.
2. Componentwise perturbation of eigenvectors: Determine the error (preferably relative) in the individual components of eigenvectors resulting from perturbations in the matrix.
3. Perturbation theory for matrices with structured nonzero elements: e.g., bidiagonal matrices, tridiagonal matrices, etc.
4. Perturbation theory for graded matrices.
5. Multiplicative perturbations: i.e., perturbations of the form $(I + E)A(I + F)$, where E and F are small.

The above categories are not disjoint, but overlap — often considerably.

It is also impossible to present a complete bibliography. The following references emphasize surveys or papers with extensive bibliographies: [56, 58, 112, 113, 125, 165, 166, 173, 266].

2

 THE QR ALGORITHM

All general-purpose eigenvalue algorithms are necessarily iterative. This statement is a consequence of Abel's famous proof that there is no algebraic formula for the roots of a general polynomial of degree greater than four. Specifically, to any polynomial $p(x) = x^n - a_{n-1}x^{n-1} - \dots - a_1x - a_0$ there corresponds a *companion matrix*

$$C_p = \begin{pmatrix} a_{n-1} & a_{n-2} & \cdots & a_1 & a_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

whose characteristic polynomial is p . Thus if we had a noniterative general-purpose eigenvalue algorithm, we could apply it to the companion matrix of an arbitrary polynomial to find its zeros in a finite number of steps — in effect producing a formula for the roots.

Thus the history of eigenvalue computations is the story of iterative methods for approximating eigenvalues and eigenvectors. An important theme in this story concerns methods based on powers of a matrix. The climax of that theme is the QR algorithm — the great success story of modern matrix computations. This algorithm, with proper attention to detail, can compute a Schur form of a general matrix in $O(n^3)$ operations. It can be adapted to reduce a real matrix in real arithmetic to a real variant of the Schur form. Other forms of the algorithm compute the eigenvalues and eigenvectors of a Hermitian matrix, the singular value decomposition of a general matrix, and the generalized Schur form of a matrix pencil. And the underlying theory of the method continues to suggest new algorithms.

The purpose of this chapter is to describe the QR algorithm and its variants. The algorithm is closely related to two algorithms that are important in their own right — the power method and the inverse power method. We will therefore begin with an exposition of those methods. In §2 we will present the general algorithm in the form generally used for complex matrices, and in the following section we will present the implicitly

shifted form for real matrices. Finally, in §4 we treat the generalized eigenvalue problem. The symmetric eigenvalue problem and the singular value decomposition will be treated separately in the next chapter.

A word on the algorithms. The QR algorithm is a complicated device that generates a lot of code. Nonetheless, its various manifestations all have a family resemblance. For this reason we will lavish attention on one variant — the explicitly shifted QR algorithm for complex matrices — and sketch the other variants in greater or lesser detail as is appropriate.

1. THE POWER AND INVERSE POWER METHODS

The power and inverse power methods are extremely simple methods for approximating an eigenvector of a matrix. In spite of their simplicity, they have a lot to teach us about the art of finding eigenvectors. For this reason and because of the connection of the methods to the QR algorithm, we devote this subsection to a detailed treatment of the algorithms. We begin with the power method.

1.1. THE POWER METHOD

The basic idea behind the power is simple. Let A be a nondefective matrix and let (λ_i, x_i) ($i = 1, \dots, n$) be a complete set of eigenpairs of A . Since the x_i are linearly independent, any nonzero vector u_0 can be written in the form

$$u_0 = \gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_n x_n.$$

Now $A^k x_i = \lambda_i^k x_i$, so that

$$A^k u_0 = \gamma_1 \lambda_1^k x_1 + \gamma_2 \lambda_2^k x_2 + \cdots + \gamma_n \lambda_n^k x_n. \quad (1.1)$$

If $|\lambda_1| > |\lambda_i|$ ($i > 2$) and $\gamma_1 \neq 0$, then as $k \rightarrow \infty$, the first term in (1.1) dominates, so that $A^k u$ becomes an increasingly accurate approximation to a multiple of the dominant eigenvector x_1 .

Although the basic idea of the power method is simple, its implementation raises problems that will recur throughout this volume. We will treat these problems in the following order.

1. Convergence of the method
2. Computation and normalization
3. Choice of starting vector
4. Convergence criteria and the residual
5. The Rayleigh quotient
6. Shifts of origin
7. Implementation
8. The effects of rounding error

Convergence

In motivating the power method, we made the simplifying assumption that A is non-defective. In fact, all that is needed is that A have a single dominant eigenpair (λ_1, x_1) in the sense of Definition 2.6, Chapter 1. Since this eigenvalue is necessarily simple, by Theorem 1.19, Chapter 1, we can find a nonsingular matrix $X = (x_1 \ X_2)$ such that

$$X^{-1}AX = \begin{pmatrix} \lambda_1 & 0 \\ 0 & M \end{pmatrix}. \quad (1.2)$$

Without loss of generality we may assume that

$$x_1^H x_1 = 1 \quad \text{and} \quad X_2^H X_2 = I. \quad (1.3)$$

Since x and the columns of X form a basis for \mathbb{C}^n , we can write any vector u_0 in the form

$$u_0 = \gamma_1 x_1 + X_2 c_2. \quad (1.4)$$

In this notation we have the following theorem.

Theorem 1.1. *Let A have a unique dominant eigenpair (λ_1, x_1) and let A be decomposed in the form (1.2), where $X = (x_1 \ X_2)$ satisfies (1.3). Let u_0 be a nonzero starting vector, decomposed in the form (1.4). If $\gamma_1 \neq 0$, then*

$$\sin \angle(x_1, A^k u_0) \leq \frac{|\lambda_1|^{-k} \|M^k\|_2 \|c_2/\gamma_1\|_2}{1 - |\lambda_1|^{-k} \|M^k\|_2 \|c_2/\gamma_1\|_2}. \quad (1.5)$$

In particular for any $\epsilon > 0$, there is a constant σ such that

$$\sin \angle(x_1, A^k u_0) \leq \frac{\sigma [\rho(M)/\lambda_1 + \epsilon]^k}{1 - \sigma [\rho(M)/\lambda_1 + \epsilon]^k}, \quad (1.6)$$

where $\rho(M)$ is the spectral radius of M .

Proof. It is easy to verify that

$$A^k u_0 = \gamma_1 \lambda_1^k x_1 + X_2 M^k c_2.$$

Let the columns of Y form an orthonormal basis for the subspace orthogonal to x_1 . Then by Lemma 3.12, Chapter 1,

$$\sin \angle(x_1, A^k u_0) = \frac{\|Y^H A^k u_0\|_2}{\|A^k u_0\|_2} = \frac{\|Y^H X_2 M^k c_2\|_2}{\|\gamma_1 \lambda_1^k x_1 + X_2 M^k c_2\|}. \quad (1.7)$$

But

$$\|Y^H X_2 M^k c_2\|_2 \leq \|M^k\|_2 \|c_2\|_2$$

and

$$\|\gamma_1 \lambda_1^k x_1 + X_2 M^k c_2\|_2 \geq |\gamma_1| |\lambda_1^k| - \|M^k\|_2 \|c_2\|_2.$$

Substituting these inequalities in (1.7) and dividing by $|\gamma_1| |\lambda_1^k|$, we get (1.5).

The inequality (1.6) follows directly from Theorem 2.9, Chapter 1. ■

Three comments on this theorem.

- Equation (1.5) shows that the error in the eigenvector approximation produced by the power method behaves like $\|M^k\|_2 / |\lambda_1|^k$. Since $\rho(M) < |\lambda_1|$, the error converges to zero at an asymptotic rate of $[\rho(M)/|\lambda_1|]^k$, a statement made precise by (1.6). However, if the powers of M are badly behaved, we may have to wait a long time to observe this asymptotic convergence; see Example 2.11, Chapter 1.
- By Theorem 2.9, Chapter 1, if M is nondefective then we may take $\epsilon = 0$ in (1.6). For example, if A has a complete system of eigenvectors with its eigenvalues ordered so that $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$, then the convergence is as $|\lambda_2/\lambda_1|^k$.
- From (1.6), we see that if γ_1 is small in comparison to c_2 , then the convergence will be retarded, although its asymptotic rate will be unchanged. In this case we say that u_0 is *deficient in λ_1* . We will return to this point later.

Computation and normalization

We have written the k th iterate \hat{u}_k in the power method in the form $\hat{u}_k = A^k u_0$. If we were to naively implement this formula, we would end up first computing $B_k = A^k$ and then computing $\hat{u}_k = B_k u_0$. This procedure requires about kn^3 flam to compute \hat{u}_k — a grotesquely large operation count.

An alternative is to observe that $\hat{u}_{k+1} = A \hat{u}_k$. This suggests the following algorithm.

1. $k = 0$
 2. **while** (not converged)
 3. $\hat{u}_{k+1} = A \hat{u}_k$
 4. $k = k+1$
 5. **end while**
- (1.8)

For a dense matrix, each iteration of this program requires about n^2 flam, where a flam denotes a pair of floating-point additions and multiplications. However, this operation count does not tell the entire story. In many applications the matrix A will be large and sparse—that is, it will have many zero elements. If the matrix is represented by an appropriate data structure, it may take far fewer than n^2 flam to compute its product with a vector. Let us consider the example of a tridiagonal matrix.

Example 1.2. A tridiagonal matrix of order n has the form illustrated below for $n =$

6:

$$T = \begin{pmatrix} a_1 & b_1 & 0 & 0 & 0 & 0 \\ c_1 & a_2 & b_2 & 0 & 0 & 0 \\ 0 & c_2 & a_3 & b_3 & 0 & 0 \\ 0 & 0 & c_3 & a_4 & b_4 & 0 \\ 0 & 0 & 0 & c_4 & a_5 & b_5 \\ 0 & 0 & 0 & 0 & c_5 & a_6 \end{pmatrix}.$$

If the nonzero diagonals of T are stored as indicated above in arrays a , b , and c , then the following program computes $y = Tx$.

1. $y[1] = a[1]*x[1] + b[1]*x[2]$
2. **for** $i = 2$ **to** $n-1$
3. $y[i] = c[i-1]*x[i-1] + a[i]*x[i] + b[i]*x[i+1]$
4. **end for** i
5. $y[n] = c[n-1]*x[n-1] + a[n]*x[n]$

This program has an operation count of $2n$ fladd + $3n$ flmlt. Thus we have reduced the work to form the product Ax from $O(n^2)$ to $O(n)$.

We shall not always get as dramatic improvement in operation counts as we did in the above example. But it serves to show that algorithms based on matrix-vector multiplies can often be economized.

As it stands the program (1.8) will tend to overflow or underflow. For example, suppose that $\lambda_1 = 10$. Then asymptotically we have $\hat{u}_k \cong 10^k \gamma_0 u_0$; i.e., the magnitude of \hat{u}_k increases by a factor of ten for each iteration. Such a process cannot continue long without overflowing the machine on which it is run. Similarly, if $\lambda_1 = 0.1$, the vectors \hat{u}_k will quickly underflow.

The cure for this problem is to note that only the direction of the vector \hat{u}_k matters: its size has no relevance to its quality as an approximate eigenvector. We can therefore scale \hat{u}_k to form a new vector u_k that is within the bounds of the machine. Usually, u_k is scaled so that $\|u_k\| = 1$ in some relevant norm. This can be done in the course of the iteration as the following program shows.

1. $k = 0$
2. **while** (not converged)
3. $u_{k+1} = Au_k$
4. $k = k+1$
5. $\sigma = 1/\|u_k\|$
6. $u_i = \sigma u_k$
7. **end while**

Since floating-point divides are expensive compared to multiplies, we do not divide u_k by $\|u_k\|$. Instead we multiply it by $1/\|u_k\|$. In most cases the savings will be small compared to the work in forming Au_k . In the sequel we will dispense with such minor economies and write, e.g., $u_k = u_k/\|u_k\|$.

Choice of a starting vector

The ratio $\|c_2\|/|\gamma_1|$ from the expansion $u_0 = \gamma_1 x_1 + X_2 c_2$ is a measure of how far the starting vector u_0 is from the dominant eigenvector x_1 . It is not surprising that a small value of γ_1 , which makes the ratio large, can retard convergence. In fact this ratio appears explicitly in the convergence bound (1.5). It is therefore important that the starting vector u_0 should have a substantial value of γ_1 .

In some cases one will have at hand an approximation to the eigenvector x_1 . For example, if we are trying to find the dominant eigenvectors of a sequence of nearby matrices A_1, A_2, \dots , then the dominant eigenvector of A_k will furnish a good starting approximation to the dominant eigenvector of A_{k+1} . Such approximations are natural starting values for the power method.

When no approximation to x_1 is available, it is customary to take u_0 to be a random vector—usually a vector of random normal deviates. The reasoning is as follows. If we write $(x_1 \ X_2)^{-1} = (y_1 \ Y_2)^H$, then it can be shown that

$$\|Y_2\|_F^2 = \|y_1\|_2^2 + n - 2.$$

Since $\gamma_1 = y_1^H u_0$ and $c_2 = Y_2^H u_0$, if u_0 is random, we may expect the ratio $\|c_2/\gamma_1\|_2$ to vary about the quantity

$$\sqrt{1 + \frac{n-2}{\|y_1\|_2^2}}. \quad (1.9)$$

Lacking special knowledge about x_1 , this is about as good as we can expect.

As a rule, one should avoid starting vectors consisting of small integers in a pattern—e.g., a vector whose elements alternate between plus and minus one. The problem is that a highly structured problem may have such a patterned eigenvector. If that eigenvector is not the dominant eigenvector, then γ_1 will be zero. On the other hand, the structure of the problem may actually dictate such a choice. For example, the vector e consisting of all ones is often a good choice for a matrix whose elements are positive.

Convergence criteria and the residual

The chief problem with determining the convergence of a sequence of vectors u_k is that the limit—in our case x_1 —is unknown. Thus we cannot simply quit when the norm of error $e_k = u_k - x_1$ is sufficiently small.

If the sequence converges swiftly enough, a natural criterion is to stop when $\|u_k - u_{k-1}\|$ is sufficiently small. For example, if $\|e_k\| \leq \alpha \|e_{k-1}\|$, where $0 < \alpha < 1$, then

$$\|u_k - u_{k-1}\| = \|e_k - e_{k-1}\| \geq \|e_{k-1}\| - \|e_k\| \geq (1 - \alpha) \|e_{k-1}\|,$$

or

$$\|e_{k-1}\| \leq \frac{\|u_k - u_{k-1}\|}{1 - \alpha}.$$

When α is, say, 0.5, $\|e_k\|$ can be no larger than twice $\|u_k - u_{k-1}\|$. But if α is near one the error can be large when $\|u_k - u_{k-1}\|$ is small. Only if we have an estimate of α —which is tricky to obtain on the fly—can we relate the distance between successive approximations to the error.

An alternative is to compute the *residual*

$$r_k = Au_k - \mu_k u_k,$$

where μ_k is a suitable approximation to λ_1 , and stop when r_k is sufficiently small. Although the residual does not give us a direct estimate of the error, it gives us something almost as good, as the following theorem shows (for generality we drop the iteration subscripts).

Theorem 1.3. *Let*

$$r = Au - \mu u.$$

Then there is a matrix

$$E = \frac{ru^H}{\|u\|_2^2} \quad (1.10)$$

such that

$$\frac{\|E\|_p}{\|A\|_p} = \frac{\|r\|_2}{\|A\|_p \|u\|_2}, \quad p = 2, F, \quad (1.11)$$

and

$$(A - E)u = \mu u. \quad (1.12)$$

If A is Hermitian and

$$\mu = \frac{u^H A u}{u^H u} \quad (1.13)$$

then there is a Hermitian matrix

$$E = \frac{ur^H + ru^H}{\|u\|_2^2}$$

satisfying (1.12). Moreover,

$$\frac{\|E\|_p}{\|A\|_p} = \begin{cases} \frac{\|r\|_2}{\|A\|_2 \|u\|_2}, & p = 2, \\ \sqrt{2} \frac{\|r\|_2}{\|A\|_F \|u\|_2}, & p = F. \end{cases} \quad (1.14)$$

Proof. The proof in the non-Hermitian case is a matter of direct verification of the equalities (1.11) and (1.12). For the Hermitian case we use the fact that the choice (1.13) of μ makes r orthogonal to u , from which (1.12) follows immediately. For the norm of E , note that

$$E^H E = \frac{rr^H}{\|u\|_2^2} + \|r\|_2^2 \frac{uu^H}{\|u\|_2^4}.$$

From this it follows that r and u are orthogonal eigenvectors of $E^H E$ with eigenvalues $\|r\|_2^2/\|u\|_2^2$. Since the remaining eigenvalues are zero, the equalities (1.14) follow from the fact that $\|E\|_2^2$ is the largest eigenvalue of $E^H E$ and $\|E\|_F^2$ is the sum of the eigenvalues of $E^H E$. ■

The theorem says that if the residual is small, then the pair (μ, u) is an exact eigenpair of a nearby matrix $A + E$. It suggests that we stop iterating when the residual is sufficiently small. The justification for this criterion is the following. In most problems we will not be given a pristine matrix A ; instead A will be contaminated with errors. For example, A may be computed, in which case A will be contaminated with rounding error — errors of order of the rounding unit. Again, A may be measured, in which case A will have errors that are, in general, much larger than the rounding unit. Now if the residual norm is smaller than the error level, then any errors in the approximate eigenvector can be accounted for by a perturbation of A that is smaller than the errors already in A . In other words, if your results are inaccurate, the inaccuracy is no greater than that induced by the error already present.

This is not the whole story. In many applications the matrix A will be sparse. Now the zeros in a sparse matrix usually arise from the underlying problem and are not subject to perturbation. But the matrix E in (1.10) will not generally share the sparsity pattern of A — in fact it is likely to be dense. Thus Theorem 1.3 gives a backward perturbation E that is unsuitable for sparse matrices.

Fortunately, there is more than one E that will make (μ, u) an exact eigenpair of A . For example, we may take

$$E = \text{diag} \left(\frac{r_1}{x_1}, \dots, \frac{r_n}{x_n} \right),$$

so that E alters only the diagonal elements of A . For general results on such structured backward perturbations see the notes and references.

Optimal residuals and the Rayleigh quotient

In computing the residual $r = Au - \mu u$ to determine convergence of the power method, we naturally take the vector u to be the k th iterate u_k . However, we have not specified how to choose μ . A natural choice is the value of μ that minimizes the residual in some norm. As is often the case, the problem is most easily solved when the norm in question is the 2-norm.

Theorem 1.4. Let $u \neq 0$ and for any μ set $r_\mu = Au - \mu u$. Then $\|r_\mu\|_2$ is minimized when

$$\mu = \frac{u^H A u}{u^H u}. \quad (1.15)$$

In this case $r_\mu \perp u$.

Proof. Without loss of generality we may assume that $\|u\|_2 = 1$. Let $(u \ U)$ be unitary and set

$$\begin{pmatrix} u^H \\ U^H \end{pmatrix} A(u \ U) = \begin{pmatrix} \nu & h^H \\ g & B \end{pmatrix}. \quad (1.16)$$

Then

$$\begin{pmatrix} u^H \\ U^H \end{pmatrix} r_\mu = \begin{pmatrix} \nu & h^H \\ g & B \end{pmatrix} \begin{pmatrix} u^H \\ U^H \end{pmatrix} u - \mu \begin{pmatrix} u^H \\ U^H \end{pmatrix} u = \begin{pmatrix} \nu - \mu \\ g \end{pmatrix}.$$

Since $\|\cdot\|_2$ is unitarily invariant,

$$\|r_\mu\|_2^2 = |\nu - \mu|^2 + \|g\|_2^2.$$

This quantity is clearly minimized when $\mu = \nu = u^H A u$.

The orthogonality of r and μ is established by direct verification. ■

A brief comment, then on to the main point.

- The proof characterizes the norm of the optimal residual as the norm of g in the partition (1.16). This characterization of the optimal residual norm will prove useful in the sequel.

The quantity $u^H A u / u^H u$ is an example of a Rayleigh quotient. We have already seen another Rayleigh quotient in (3.21), Chapter 1. There it had the form $y^H(A + E)x$ where x and y (normalized so that $y^H x = 1$) were left and right eigenvectors corresponding to a simple eigenvalue λ of A . This Rayleigh quotient provides a highly accurate approximation to the corresponding eigenvalue of $A + E$. Generalizing from these two Rayleigh quotients, we make the following definition.

Definition 1.5. Let u and v be vectors with $v^H u \neq 0$. Then the quantity

$$\frac{v^H A u}{v^H u}$$

is called a RAYLEIGH QUOTIENT.

If either u or v is an eigenvector corresponding to an eigenvalue λ of A , then the Rayleigh quotient reproduces that eigenvalue. By continuity when u or v is near an eigenvector, the Rayleigh quotient will approximate the corresponding eigenvalue. Thus in the power method, the Rayleigh quotients $u_k^H A u_k / u_k^H u_k$ provide a sequence of increasingly accurate approximations to λ . We will return to Rayleigh quotients in §2.2, Chapter 4, where we will treat the approximation of eigenspaces and their eigen-blocks.

Given matrix A , a shift κ , a convergence criterion ϵ , and a nonzero starting vector x , this algorithm iterates the shifted power method to convergence, returning the approximate eigenpair (μ, x) .

1. $Anorm = \|A\|_F$
2. $x = x/\|x\|_2$
3. **while** (true)
4. $y = Ax$
5. $\mu = x^H y$
6. $r = y - \mu x$
7. $x = y - \kappa x$
8. $x = x/\|x\|_2$
9. **if** ($\|r\|_2/Anorm \leq \epsilon$) **leave while fi**
10. **end while**

Algorithm 1.1: The shifted power method

Shifts and spectral enhancement

If A is nondefective and the eigenvalues of A satisfy

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|,$$

then the power method almost always converges to the dominant eigenvector x_1 with rate $|\lambda_2/\lambda_1|$. If this ratio is near one, the convergence will be slow.

Sometimes we can improve the rate of convergence by working with the matrix $A - \kappa I$, whose eigenvalues $\lambda_1 - \kappa, \dots, \lambda_n - \kappa$ are shifted by the constant κ . For example, if the eigenvalues of A are 1, 0, and -0.99 , then the power method converges with ratio 0.99. If we shift A by $\kappa = -0.5$, the eigenvalues become 1.5, .5, and -0.49 , and the convergence ratio becomes $\frac{1}{3}$ — a dramatic improvement.

This shifting of the matrix to improve the disposition of its eigenvalues is an example of *spectral enhancement*. Shifting has the advantage that it is cheap: we can calculate $(A - \kappa I)u$ in the form $Au - \kappa u$ at little additional cost. But it can only improve performance by so much. In the above example, $1/3$ is nearly the best convergence ratio we can obtain by shifting. In the next subsubsection we will consider a very powerful spectral enhancement — the shift and invert operation. However, we pay for this power in the coin of increased computational costs.

Implementation

Algorithm 1.1 implements the shifted power method. There are three things to say about this straightforward program.

- Both the residual and the next iterate are computed from the intermediate vector

$y = Ax$ to avoid an extra multiplication by A . Similarly, the norm of A is computed outside the loop. This parsimonious style of coding may seem natural, but it is easy to forget when one is writing in a high-level language like MATLAB.

- On termination the algorithm returns an approximate eigenvector (μ, x) . However, our convergence test is based on the pair (μ, \hat{x}) , where \hat{x} is the previous iterate. From Theorem 1.3 we know that (μ, \hat{x}) is an exact eigenpair of $A + E$, where the relative error $\|E\|_F/\|A\|_F$ is less than or equal to the convergence criterion ϵ . Strictly speaking, we cannot say the same thing about (μ, x) . Nonetheless, we return the vector x because it is almost certainly more accurate.
- In a working implementation one would specify a maximum number of iterations, and the code would give an error return when that number was exceeded.

The effects of rounding error

We must now consider the effects of rounding error on the power method. We will suppose that the computations are performed in standard floating-point arithmetic with rounding ϵ_M . For definiteness we will assume that $\|u_k\|_2 = 1$ and $\|A\|_F = 1$ in the following discussion.

The major source of error in the method is the computation of the product Au_k . An elementary rounding-error analysis shows that for a dense matrix

$$\text{fl}(Au_k) = (A + E)u_k, \quad (1.17)$$

where

$$\frac{\|E\|_F}{\|A\|_F} \leq n\epsilon_M. \quad (1.18)$$

Equation (1.17) and the bound (1.18) imply that at each stage the power method is trying to compute the dominant eigenvector of a slightly perturbed matrix. This perturbation induces an error in the eigenvector x_1 , and we cannot expect the power method to compute x_1 to an accuracy greater than that error. In particular, by Theorem 3.13, Chapter 1, we can expect no greater accuracy in the angle between the computed vector and x_1 than roughly $n\epsilon_M\|A\|_F/\text{sep}(\lambda_1, M)$. This is only n times as large as the error we could expect from simply rounding the matrix [see (2.2), Chapter 1].

The limiting accuracy of the power method is therefore quite satisfactory. But can the limit be attained? The answer is not easy — surprisingly, in view of the simplicity of the convergence proof for the power method. However, we can say that rounding error not only changes the vector Ax to $(A + E)x$, but it also changes the residuals corresponding to these vectors. Moreover, the change is very small, of the order of the rounding unit. Thus, if the residual corresponding to Ax is reduced far enough (and that need not be very far), then the computed residual corresponding to $(A + E)x$ will also be reduced. Thus we can expect to see a steady decrease in the residual until the limiting accuracy is attained.

Assessment of the power method

The advantages of the power method are that it requires only matrix-vector multiplications and that it is essentially globally convergent, provided the matrix has a single, simple dominant eigenvalue. It has the disadvantages that it only computes the dominant eigenvector, and the convergence can be very slow if the dominant eigenvalue is near in magnitude to the subdominant one. Moreover, if a real matrix has a complex dominant eigenvalue, then its conjugate is also an eigenvalue with the same absolute value, and the power method cannot converge. For these reasons, the power method in its natural form is little used these days. However, it plays an implicit role in the QR algorithm, and a variant—the inverse power method—is widely used. We now turn to that variant.

1.2. THE INVERSE POWER METHOD

The difficulties with the power method can be overcome provided we can find a spectral enhancement that makes an eigenvalue of interest the dominant eigenvalue of the matrix. The traditional enhancement for doing this is called the shift-and-invert enhancement, and its combination with the power method is called the inverse power method. We begin with the enhancement.

Shift-and-invert enhancement

Let A have eigenvalues $\lambda_1, \dots, \lambda_n$ in no particular order, and suppose that λ_1 is simple. If we have an approximation κ to λ_1 , then the eigenvalues of the matrix $(A - \kappa I)^{-1}$ are

$$\mu_1 = \frac{1}{\lambda_1 - \kappa}, \mu_2 = \frac{1}{\lambda_2 - \kappa}, \dots, \mu_n = \frac{1}{\lambda_n - \kappa}.$$

Now as $\kappa \rightarrow \lambda_1$, $\mu_1 \rightarrow \infty$. On the other hand, for $i > 1$ the μ_i approach $(\lambda_i - \lambda)^{-1}$, which are finite quantities. Thus by choosing κ sufficiently near λ_1 we can transform λ_1 into a dominant eigenvalue, and that dominance can be made as large as we like. This spectral enhancement is called the *shift-and-invert enhancement*.

The inverse power method

The *inverse power method* is simply the power method applied to the enhanced matrix $(A - \kappa I)^{-1}$. Specifically, given a vector x and a shift κ we generate a new iterate in the form

$$\hat{x} = \frac{(A - \kappa I)^{-1}x}{\|(A - \kappa I)^{-1}x\|_2}.$$

In principle, we could use Algorithm 1.1 to carry out the inverse power method. However, owing to the nature of the shift-and-invert enhancement, we can compute a residual almost for free.

Given a matrix A , a shift κ , a convergence criterion ϵ , and a nonzero starting vector x , this algorithm iterates the inverse power method to convergence, returning an approximate eigenpair (μ, x) .

1. **while** (true)
2. Solve the system $(A - \kappa I)y = x$
3. $\hat{x} = y/\|y\|_2$
4. $w = x/\|y\|_2$
5. $\rho = \hat{x}^H w$
6. $\mu = \kappa + \rho$
7. $r = w - \rho \hat{x}$
8. $x = \hat{x}$
9. **if** ($\|r\|_2/\|A\|_F \leq \epsilon$) **leave while fi**
10. **end while**

Algorithm 1.2: The inverse power method

Specifically, let

$$y = (A - \kappa I)^{-1}x \quad \text{and} \quad \hat{x} = y/\|y\|_2.$$

Since $(A - \kappa I)y = x$, we have

$$(A - \kappa I)\hat{x} = \frac{x}{\|y\|_2} \equiv w.$$

If we set

$$\rho = \hat{x}^H(A - \kappa I)\hat{x} = \hat{x}^H w,$$

then the optimal residual for \hat{x} is (see Theorem 1.4)

$$r = [A - (\kappa + \rho)I]\hat{x} = w - \rho \hat{x}.$$

Note that as \hat{x} converges to an eigenvector, the Rayleigh quotient $\mu = \kappa + \rho = \hat{x}^H A \hat{x}$ converges to the corresponding eigenvalue of A .

Algorithm 1.2 implements this scheme. It is evident that the major work in the algorithm consists in solving the system $(A - \kappa I)y = x$ in statement 2, and some care must be taken not to inadvertently increase this work.

Most linear system solvers first compute a decomposition and then use the decomposition to solve the system. We will say that this process requires a *factorization*. With dense systems, the decomposition is usually an LU decomposition computed by Gaussian elimination with partial pivoting, which costs about $\frac{1}{3}n^3$ fflam. The subsequent solution of the linear system, on the other hand, requires only n^2 fflam. It is therefore important that the computation of the decomposition be placed outside the while

loop in Algorithm 1.2. In particular, one should avoid software that solves a linear system by invisibly computing a decomposition.

When A is sparse, the operations counts for computing a decomposition or factorization may be less than $O(n^3)$. Nonetheless, they are usually an important, if not dominant, part of any calculation involving shift-and-invert enhancement. We will return to this point repeatedly in the second half of this volume, where the concern is with large eigenproblems.

An example

The convergence of the inverse power method can be stunningly fast when the shift is near the eigenvalue in question. For example, the matrix

$$\begin{array}{ccccc} 3.022e+00 & 6.298e-01 & -3.470e+00 & 2.209e+00 & -1.079e-01 \\ 1.907e+00 & 3.399e+00 & 4.475e+00 & -1.814e+00 & 7.095e-01 \\ -1.833e+00 & -3.820e-02 & 4.699e+00 & -2.307e+00 & 4.158e-01 \\ -1.513e+00 & -6.487e-01 & 9.776e-01 & 1.274e+00 & 1.877e-01 \\ -2.184e+00 & 8.906e-01 & -5.765e+00 & 2.510e+00 & 2.606e+00 \end{array}$$

(shown only to four figures) has eigenvalues 1, 2, 3, 4, 5. In the following experiment, we took $\kappa = 1.01$ and took x to be a normalized random vector. The errors in the Rayleigh quotients and the residual norms for several iterations of Algorithm 1.2 were

$ \lambda_1 - \mu $	$\ r\ $
9.2e-03	4.4e-04
7.7e-05	3.2e-06
7.5e-07	3.3e-08
7.6e-09	3.4e-10
7.7e-11	3.4e-12
7.8e-13	3.4e-14
8.4e-15	3.4e-16

It is seen that each iteration adds about two significant figures of accuracy to the eigenvalue until the eigenvalue is calculated to working accuracy ($\epsilon_M \cong 10^{-16}$). On the other hand if we take $\kappa = 1 + 10^{-8}$, we get the following results.

$ \lambda_1 - \mu $	$\ r\ $
9.2e-09	4.4e-10
1.3e-16	3.1e-18

The iteration converges to working accuracy in two iterations. If κ approximates λ_1 to working accuracy, the iteration converges in one iteration. This behavior is typical of the inverse power method applied to a well-conditioned eigenpair.

The effects of rounding error

The nearer the shift κ is to an eigenvalue, the faster the inverse power method converges. But if κ is near an eigenvalue, then $A - \kappa I$ is nearly singular, and the system

$(A - \kappa I)y = x$ in statement 2 will be solved inaccurately. In fact if κ approximates λ to working accuracy, the solution will be completely inaccurate. Yet the method still converges. Why?

The answer to this question is furnished by the rounding-error analysis of the solution of linear systems (see §I:3.4). If a stable algorithm (e.g., Gaussian elimination with partial pivoting) is used to solve the system $(A - \kappa I)y = x$, then the computed solution will satisfy

$$(A - \kappa I + E)y = x,$$

where $\|E\|_F/\|A\|_F$ is of the order of the rounding unit. If the eigenpair we are looking for is well conditioned, this perturbation in $A - \kappa I$ changes it only insignificantly. Thus the inverse power method moves toward an approximate eigenpair that is for practical purposes indistinguishable from the exact eigenpair.

To put the matter in another way, when κ is near an eigenvalue, the vector y is indeed computed inaccurately, but most of the inaccuracy is in the direction of the eigenvector being approximated. Since we are only concerned with the direction of the eigenvector, this error in y does not introduce any significant errors in the approximate eigenvector.

The Rayleigh quotient method

We have seen that as we proceed with the inverse power method, it produces Rayleigh quotients μ that are generally more accurate than the shift κ . Since the rate of convergence depends on the accuracy of κ , we can improve convergence by replacing κ with μ . This amounts to adding the statement $\kappa = \mu$ after statement 8 in Algorithm 1.2. The result is called the *Rayleigh quotient iteration*. It can be shown that once it starts converging to a simple eigenvalue, it converges at least quadratically.

The main difficulty with the Rayleigh quotient method is that each time the shift changes, one must decompose the matrix $A - \kappa I$ to solve the system $(A - \kappa I)y = x$. Thus for full, dense matrices the Rayleigh quotient method takes $O(n^3)$ operations for each iteration, which accounts for the fact that the Rayleigh quotient iteration in its natural form is not much used. However, in a disguised form it reappears in the QR algorithm, which is the subject of the rest of this chapter.

1.3. NOTES AND REFERENCES

The power method

As an explicit method for computing a dominant eigenpair, the power method goes back at least to 1913 [182]. Hotelling in his famous paper on variance components [120, 1933] used it to compute his numerical results. Aitken [2, 1937] gave an extensive analysis of the method.

Many people who use the power method do so unknowingly. For example, a simulation of a discrete-time discrete-state Markov chain amounts to applying the power method to the matrix of transition probabilities.

For a time, before the advent of the QR algorithm, the power method was one of the standard methods for finding eigenvalues and eigenvectors on a digital computer. Wilkinson, working at the National Physical Laboratory in England, honed this tool to as fine an edge as its limitations allow. A particularly important problem was to deflate eigenvalues as they are found so that they do not get in the way of finding smaller eigenvalues. For more see *The Algebraic Eigenvalue Problem* [300, pp. 570–599] and [295].

Residuals and backward error

The use of a residual to determine convergence is inevitable in most problems, since seldom will we have the wherewithal to determine the accuracy of the current approximation to an eigenvalue or eigenvector. Theorem 1.3, which says that a small residual implies a small backward error, provides a justification for tests based on the size of a suitably scaled residual. The first part is due to Wilkinson—e.g., see [299, p. 141]. The result for Hermitian matrices in a more general form is due to Powell [214], whose application was optimization.

The justification of a convergence criterion by appeal to a backward error result is part of the paradigm of backward rounding-error analysis. For further discussion of this matter see any general textbook on numerical linear algebra—e.g., [62, 95, 262, 278, 299, 300].

The failure of the backward perturbation in Theorem 1.3 to respect the zero structure of a sparse matrix is a drawback only if the eigenpair in question is especially sensitive to perturbations in the zero elements. Even in this case one can apply a very general theorem of Oettli and Prager [185], which allows one to specify the relative sizes of the elements in the backward error. Whether this result can be generalized to Hermitian matrices is an open question.

The Rayleigh quotient

The Rayleigh quotient is due to Lord Rayleigh (J. W. Strutt), who defined it for symmetric matrices. Some of its many generalizations will appear throughout this volume.

The fact that the Rayleigh quotient gives an optimal residual is due to Wilkinson [300, p. 172].

Shift of origin

Wilkinson used shifting to good effect at the National Physical Laboratory. The shift was actually entered by machine operators in binary by switches at the console before each iteration [300, p. 577].

The inverse power and Rayleigh quotient methods

The inverse power method is due to Wielandt [294, 1944]. Ipsen [124] gives a survey of the method and its properties, along with an extensive list of references. The

method is chiefly used to compute eigenvectors of matrices whose eigenvalues can be approximated by other means (e.g., see Algorithm 2.6, Chapter 3).

The fact that $A - \kappa I$ is ill conditioned when κ is near an eigenvalue of A makes it certain that the vector in the inverse power method will be inaccurately calculated. The analysis given here, which shows that the inaccuracies do no harm, is due to Wilkinson (e.g., see [300, pp. 619–622]).

It is important to use the Rayleigh quotient to compute the optimal residual in Algorithm 1.2. An obvious reason is that if we use the constant shift κ , the residual can never converge to zero. A more subtle reason is the behavior of the residual when the matrix is strongly nonnormal—that is, when the off-diagonal part of its Schur form dominates the diagonal part [111]. In that case, the first residual will generally be extremely small and then will increase in size before decreasing again. The use of the Rayleigh quotient to compute the optimal residual mitigates this problem. For more see [96] and [124].

The Rayleigh quotient iteration has been analyzed extensively in a series of papers by Ostrowski [188, 189, 190, 191, 192]. For additional results on the symmetric case, see [206, §4.6–4.9].

2. THE EXPLICITLY SHIFTED QR ALGORITHM

The QR algorithm is an iterative method for reducing a matrix A to triangular form by unitary similarity transformations. Its shifted form can be written simply—and cryptically—as follows. Let $A_0 = A$.

1. **for** $k = 0, 1, 2, \dots$
 2. Choose a shift κ_k
 3. Factor $A_k - \kappa_k I = Q_k R_k$, where Q is orthogonal
 and R is upper triangular
 4. $A_{k+1} = R_k Q_k + \kappa_k I$
 5. **end for** k
- (2.1)

In other words, to get from one iterate to the next—i.e., to perform one *QR step*—subtract the shift from the diagonal, compute a QR factorization, multiply the factors in reverse order, and add back the shift to the diagonal. The method is called the *explicitly shifted QR algorithm* because the shift is explicitly subtracted from the diagonal of A and added back at the end of the process.

The matrices produced by the algorithm (2.1) satisfy

$$\begin{aligned} A_{k+1} &= R_k Q_k + \kappa_k I \\ &= Q_k^H (A_k - \kappa_k I) Q_k + \kappa_k I \quad [\text{since } R_k = Q_k^H (A_k - \kappa_k I)] \\ &= Q_k^H A Q_k. \end{aligned}$$

Thus A_{k+1} is unitarily similar to A_k . However, this fact alone is not sufficient reason for performing this sequence of operations. Why, then, is the algorithm worth considering?

The answer is that the algorithm is an elegant way of implementing the inverse power method with shift κ_k . At the same time it is a variant of the power method. These relationships go far toward explaining the remarkable effectiveness of the algorithm, and the next two subsections are devoted to laying them out in detail. The remaining subsections are devoted to the practical implementation of the algorithm.

2.1. THE QR ALGORITHM AND THE INVERSE POWER METHOD

In this subsection we will show how the QR algorithm is related to the inverse power method. This relation allows us to investigate the convergence of the algorithm. For brevity we will drop the iteration subscripts and denote the current matrix by A and the next QR iterate by \hat{A} .

Schur from the bottom up

In establishing the existence of Schur decompositions (Theorem 1.12, Chapter 1) we showed that if we have knowledge of an eigenvector we could deflate the corresponding eigenvalue of the matrix by a unitary similarity transformation. A step of the QR algorithm can be regarded as an approximation to such a deflation, with the difference that the deflation occurs at the bottom of the matrix.

Specifically, let $Q = (Q_*, q)$ be unitary and write

$$Q^H A Q = \begin{pmatrix} Q_*^H A Q_* & Q_*^H A q \\ q^H A Q_* & q^H A q \end{pmatrix} \equiv \begin{pmatrix} \hat{B} & \hat{h} \\ \hat{g}^H & \hat{\mu} \end{pmatrix}. \quad (2.2)$$

If (λ, q) is a left eigenpair of A , then

$$\hat{g}^H = q^H A Q_* = \lambda q^H Q_* = 0 \quad \text{and} \quad \hat{\mu} = q^H A q = \lambda q^H q = \lambda.$$

It follows that

$$Q^H A Q = \begin{pmatrix} \hat{B} & \hat{h} \\ 0 & \lambda \end{pmatrix}. \quad (2.3)$$

Thus we have deflated the problem. If we repeat the procedure inductively on \hat{B} , we obtain an alternate proof of Schur's theorem — one in which the reduction is accomplished from the bottom up by left eigenvectors.

The QR choice

The above reduction is not an effective computational procedure because it requires that one be given eigenvectors of A , \hat{B} , etc. If, however, we choose q to be an approximate eigenvector of A , then g in (2.2) will be small because $\|g\|_2$ is the norm of the optimal residual $q^H A - \hat{\mu} q^H$ (see Theorem 1.4). The QR algorithm implicitly chooses q to be a vector produced by the inverse power method with shift κ .

To see this, let us rewrite the QR factorization of the shifted matrix $A - \kappa I$ in the form

$$\begin{pmatrix} Q_*^H \\ q^H \end{pmatrix} (A - \kappa I) = \begin{pmatrix} R_* \\ r^H \end{pmatrix}, \quad (2.4)$$

where because of the triangularity of R we have

$$r^H = r_{nn} \mathbf{e}_n^T.$$

The last row of (2.4) is $q^H(A - \kappa I) = r_{nn} \mathbf{e}_n^T$. Hence

$$q^H = r_{nn} \mathbf{e}_n^T (A - \kappa I)^{-1}. \quad (2.5)$$

Equation (2.5) shows that the last column of the matrix Q generated by the QR algorithm is the result of the inverse power method with shift κ applied to the vector \mathbf{e}_n^T . If κ is close to an eigenvalue of A , then we can expect q to be a good approximation to a left eigenvector of A and hence \hat{g}^H in (2.2) to be small.

The above analysis also suggests a choice of shift. Partition A in the form

$$A = \begin{pmatrix} B & h \\ g^H & \mu \end{pmatrix}. \quad (2.6)$$

Then (μ, \mathbf{e}_n) is an approximate left eigenpair whose residual norm is $\|g\|_2$. Hence if g is small, μ should approximate an eigenvalue of A and is a natural choice for a shift. This shift is called the *Rayleigh quotient shift* because μ is the Rayleigh quotient $\mathbf{e}_n^T A \mathbf{e}_n$.

Error bounds

To show that the QR algorithm converges to a deflated matrix, we must show that the vector g in (2.6) converges to zero. We begin by giving a bound that relates the norms of g and \hat{g} in successive iterates.

The key to the derivation is to work with a partitioned form of the relation of $A - \kappa I = QR$. Specifically, let us write

$$A - \kappa I \equiv \begin{pmatrix} B - \kappa I & h \\ g^H & \mu - \kappa \end{pmatrix} = \begin{pmatrix} P & f \\ e^H & \pi \end{pmatrix} \begin{pmatrix} S & r \\ 0 & \rho \end{pmatrix} \equiv QR, \quad (2.7)$$

and also write the relation $\hat{A} - \kappa I = RQ$ in the form

$$\hat{A} - \kappa I \equiv \begin{pmatrix} \hat{B} - \kappa I & \hat{h} \\ \hat{g}^H & \hat{\mu} - \kappa \end{pmatrix} = \begin{pmatrix} S & r \\ 0 & \rho \end{pmatrix} \begin{pmatrix} P & f \\ e^H & \pi \end{pmatrix} = RQ. \quad (2.8)$$

From these partitions we proceed in stages.

Since Q is unitary, the norms of its last row and last column must be one. Hence $\|e\|_2^2 + \pi^2 = \|f\|_2^2 + \pi^2 = 1$, and

$$\|e\|_2 = \|f\|_2. \quad (2.9)$$

Next we want to show that e is small when g is small. Computing the $(2, 1)$ -block of (2.7), we get $g^H = e^H S$. If we assume that S is nonsingular and set

$$\sigma = \|S^{-1}\|_2,$$

then

$$\|e\|_2 \leq \sigma \|g\|_2. \quad (2.10)$$

We now need a bound on ρ . If we use the fact that Q is unitary to rewrite (2.7) in the form

$$\begin{pmatrix} P^H & e \\ f^H & \pi \end{pmatrix} \begin{pmatrix} B - \kappa I & h \\ g^H & \mu - \kappa \end{pmatrix} = \begin{pmatrix} S & r \\ 0 & \rho \end{pmatrix},$$

then we find that $\rho = f^H h + \pi(\mu - \kappa)$. Hence from (2.9) and (2.10) and the fact that $|\pi| \leq 1$,

$$\rho \leq \sigma \|g\|_2 \|h\|_2 + |\mu - \kappa|. \quad (2.11)$$

Finally, from (2.8) we find that $\hat{g}^H = \rho e^H$. Hence from (2.10) and (2.11) we get

$$\|\hat{g}\|_2 \leq \sigma^2 \|h\|_2 \|g\|_2^2 + \sigma |\mu - \kappa| \|g\|_2. \quad (2.12)$$

This is our final bound.

Rates of convergence

Let us restore our iteration subscripts and write the bound (2.12) in the form

$$\|g_{k+1}\|_2 \leq \sigma_k^2 \|h_k\|_2 \|g_k\|_2^2 + \sigma_k |\mu_k - \kappa_k| \|g_k\|_2. \quad (2.13)$$

This bound can be used to show that if g_0 is sufficiently small and μ_0 is sufficiently near a simple eigenvalue λ of A then the iteration with Rayleigh quotient shift converges in the sense that $g_k \rightarrow 0$ and $\mu_k \rightarrow \lambda$. However, the proof is tedious and not particularly informative. We will therefore assume that the iteration converges and determine the rate of convergence.

We will assume that we have constants η and σ such that

$$\|h_k\|_2 \leq \eta \quad \text{and} \quad \sigma_k = \|S_k^{-1}\|_2 \leq \sigma.$$

Since the A_k are orthogonally similar, we can simply take $\eta = \|A_2\|_2$. The fact that σ_k can be bounded is a nontrivial corollary of the formal proof of convergences.

We may now replace (2.13) by

$$\|g_{k+1}\|_2 \leq \sigma^2 \eta \|g_k\|_2^2$$

(the second term vanishes because the Rayleigh quotient shift takes $\kappa_k = \mu_k$). Thus when the QR algorithm converges to a simple eigenvalue, the norm of g_{k+1} is bounded by a quantity proportional to the square of the norm of g_k . We say that $\|g_k\|_2$ converges at least quadratically to zero. Quadratic convergence, once it sets in, is very fast. For example, if $\sigma^2\eta = 1$ and $\|g_0\|_2 = 10^{-1}$, then

$$\begin{aligned}\|g_1\|_2 &\leq 10^{-2}, \\ \|g_2\|_2 &\leq 10^{-4}, \\ \|g_3\|_2 &\leq 10^{-8}, \\ \|g_4\|_2 &\leq 10^{-16}.\end{aligned}$$

Thus four iterations are sufficient to reduce the error by a factor corresponding to the IEEE double-precision rounding unit.

If A_0 is Hermitian, then so are the iterates A_k . In particular, we have $h_k = g_k$, so that we may replace η by $\|g_k\|_2$. Thus our bound becomes

$$\|g_{k+1}\|_2 \leq \sigma^2 \|g_k\|_2^3. \quad (2.14)$$

This type of convergence is called cubic, and it is blindingly fast.

General comments on convergence

Our convergence analysis seems to depend on the hypothesis that μ_k is converging to a simple eigenvalue λ . For example, if λ is multiple, the matrices S_k^{-1} can be unbounded, so that our uniform upper bound σ no longer exists. (It is a good exercise to see why this happens.) Nonetheless, if λ is nondefective, the convergence of the QR algorithm remains quadratic, whatever its multiplicity. In addition, if A is Hermitian, λ is necessarily nondefective, and the convergence remains cubic. If λ is defective on the other hand, the convergence may be only linear. For more on these points see the notes and references.

We have used the Rayleigh quotient shift in our analysis, which removes the term $\sigma_k|\mu_k - \kappa_k| \|g_k\|_2$ in (2.13). However, any shift for which $|\mu_k - \kappa_k| = O(\|g\|_k)$ will give us quadratic convergence. In particular, this is true of the Wilkinson shift defined by Algorithm 2.7 below.

The complete convergence analysis is local; that is, it assumes that μ_0 is sufficiently near a simple eigenvalue of A . There are no theorems for the global convergence of the QR algorithm applied to a general matrix. The typical behavior seems to be that the algorithm spends its first few iterations floundering and then homes in on an eigenvalue. Subsequent eigenvalues are found with fewer and fewer iterations. Why this happens is partly explained by the relation of the QR algorithm to the power method, which we turn to now.

2.2. THE UNSHIFTED QR ALGORITHM

The relation of the QR algorithm to the inverse power method explains its ability to converge swiftly to a particular eigenvalue. In practice, however, the algorithm also

produces approximations to the other eigenvalues, albeit more slowly. The reason for this is that the QR algorithm is also related to the power method. The explication of this relation is the subject of this subsection. We begin with a far-reaching characterization of the k th QR iterate.

The QR decomposition of a matrix polynomial

Each step of the QR algorithm consists of a unitary similarity transformation $A_{k+1} = \check{Q}_k^H A Q_k$. When the algorithm has finished its work, we will need a single transformation that moves us from the initial matrix $A = A_0$ to the final matrix A_{k+1} . This can be accomplished by multiplying the transformations together as we go along, a process called *accumulating the transformations*. The result is a matrix

$$\check{Q}_k = Q_0 Q_1 \cdots Q_k$$

satisfying

$$\check{Q}_k^H A_0 \check{Q}_k = A_{k+1}.$$

The matrix \check{Q}_k has another useful characterization.

Theorem 2.1. *Let Q_0, \dots, Q_k and R_0, \dots, R_k be the orthogonal and triangular matrices generated by the QR algorithm with shifts $\kappa_0, \dots, \kappa_k$ starting with the matrix A . Let*

$$\check{Q}_k = Q_0 \cdots Q_k \quad \text{and} \quad \check{R}_k = R_k \cdots R_0.$$

Then

$$\check{Q}_k \check{R}_k = (A - \kappa_0 I) \cdots (A - \kappa_k I). \quad (2.15)$$

Proof. We have

$$\begin{aligned} R_k &= (A_{k+1} - \kappa_k I) Q_k^H \\ &= \check{Q}_k^H (A - \kappa_k I) \check{Q}_k Q_k^H \\ &= \check{Q}_k^H (A - \kappa_k I) \check{Q}_{k-1}. \end{aligned}$$

Postmultiplying this relation by \check{R}_{k-1} , we get

$$\check{R}_k = \check{Q}_k^H (A - \kappa_k I) \check{Q}_{k-1} \check{R}_{k-1}.$$

Hence

$$\check{Q}_k \check{R}_k = (A - \kappa_k I) \check{Q}_{k-1} \check{R}_{k-1}.$$

An obvious induction on the product $\check{Q}_{k-1} \check{R}_{k-1}$ completes the proof. ■

Theorem 2.1 can be interpreted as follows. Let

$$p(\tau) = \tau^k + a_{k-1}\tau^{k-1} + \cdots + a_0 = (\tau - \kappa_1)\cdots(\tau - \kappa_k)$$

be a polynomial of degree k whose zeros are $\kappa_1, \dots, \kappa_k$. We can then define $p(A)$ by either of the formulas

$$p(A) = \begin{cases} A^k + a_{k-1}A^{k-1} + \cdots + a_0I, \\ (A - \kappa_0I)\cdots(A - \kappa_kI). \end{cases}$$

(It is a good exercise to verify that these forms are equivalent.) These formulas give us two ways of evaluating the QR factorization of $p(A)$. If we use the first formula, we simply form $p(A)$ and calculate the QR decomposition of the result, say by Householder triangularization. However, if we know the zeros κ_i of $p(\tau)$, then we can simply apply the shifted QR algorithm with shifts $\kappa_1, \dots, \kappa_k$, accumulating the transformations Q_j and the product of the triangular matrices R_j . The result is the QR factorization $\check{Q}_k \check{R}_k$ of $p(A)$.

The QR algorithm and the power method

Theorem 2.1 has more than one application. For the present, we are interested in what it says about the relation of the QR algorithm to the power method.

Let us say that a sequence of QR steps is *unshifted* if the shifts κ_k are zero. Then by (2.15), the matrices \check{Q}_k and \check{R}_k from the unshifted QR algorithm satisfy

$$\check{Q}_k \check{R}_k = A^k.$$

Multiplying this relation by e_1 and recalling that \check{R}_k is upper triangular, we have

$$\check{r}_{11}^{(k)} \check{q}_1^{(k)} = \check{r}_{11}^{(k)} \check{Q}_k e_1 = A^k e_1.$$

Thus the first column of \check{Q}_k is the normalized result of applying k iterations of the power method to e_1 .

Now if the conditions for convergence of the power method are met, the vectors $\check{q}_1^{(k)}$ approach the dominant eigenvector of A . It follows that if we partition

$$A_k = \check{Q}_k^H A \check{Q}_k = \begin{pmatrix} \mu_k & h_k^H \\ g_k & B_k \end{pmatrix}$$

then $g_k \rightarrow 0$ and $\mu_k \rightarrow \lambda_1$, where λ_1 is the dominant eigenvalue of A .

Convergence of the unshifted QR algorithm

The relation of the QR algorithm to the power method is a special case of a more general theorem — under appropriate conditions the unshifted QR algorithm actually triangularizes A .

Theorem 2.2. Let

$$X^{-1}AX = \Lambda \equiv \text{diag}(\lambda_1, \dots, \lambda_n),$$

where

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n| > 0. \quad (2.16)$$

Suppose that X^{-1} has an LU decomposition $X^{-1} = LU$, where L is unit lower triangular, and let $X = QR$ be the QR decomposition of X . If A^k has the QR decomposition $A^k = \check{Q}_k \check{R}_k$, then there are diagonal matrices D_k with $|D_k| = I$ such that $\check{Q}_k D_k \rightarrow Q$.

Proof. We have

$$\begin{aligned} A^k &= X\Lambda^k X^{-1} \\ &= QR\Lambda^k LU \\ &= QR(\Lambda^k L\Lambda^{-k})(\Lambda^k U). \end{aligned}$$

Now for $i > j$ the (i, j) -element of $\Lambda^k L\Lambda^{-k}$ is $\ell_{ij}(\lambda_i/\lambda_j)^k$, which by (2.16) approaches zero—i.e., $\Lambda^k L\Lambda^{-k}$ approaches the identity matrix.

Write $\Lambda^k L\Lambda^{-k} = I + E_k$, where $E_k \rightarrow 0$. Then

$$A^k = Q(I + RE_k R^{-1})(R\Lambda^k U).$$

Let $\hat{Q}_k \hat{R}_k$ be the QR decomposition of $I + RE_k R^{-1}$. Then

$$A^k = (Q\hat{Q}_k)(\hat{R}_k R\Lambda^k U).$$

Since $I + RE_k R^{-1} \rightarrow I$, by the continuity of the QR factorization $\hat{Q}_k \rightarrow I$. Let the diagonals of $\hat{R}_k R\Lambda^k U$ be $\delta_1, \dots, \delta_n$, and set

$$D_k = \text{diag}(\bar{\delta}_1/|\delta_1|, \dots, \bar{\delta}_n/|\delta_n|).$$

Then the triangular factor in the decomposition

$$A^k = (Q\hat{Q}_k D_k^{-1})(D_k \hat{R}_k R\Lambda^k U)$$

has positive diagonal elements. It follows by the uniqueness of the QR decomposition that $\check{Q}_k = Q\hat{Q}_k D_k^{-1}$, and

$$\check{Q}_k D_k = Q\hat{Q}_k \rightarrow Q. \blacksquare$$

Four comments on this theorem.

- We know [see (1.9), Chapter 1] that the columns of the Q-factor of X are the Schur vectors of A corresponding to the eigenvalues in the order (2.16). Consequently, with

its columns suitably scaled the matrix \check{Q}_k converges to the orthogonal part of the Schur decomposition of A . It follows that the QR iterates $A_k = \check{Q}_k^H A \check{Q}_k$ must converge to the triangular factor of the Schur decomposition of A . Thus under the conditions of Theorem 2.2 the unshifted QR algorithm produces a sequence of matrices tending to a triangular matrix, which is what we set out to show.

- We have assumed that the matrix X of right eigenvectors has a QR decomposition and that the matrix X^{-1} of left eigenvectors has an LU decomposition. The former decomposition always exists, but the latter need not. For example, if $A = \text{diag}(0.5, 1.0)$, then

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} A \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \text{diag}(1.0, 0.5),$$

so that the matrix X that orders the eigenvalues in descending order, and

$$X = X^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

has no LU decomposition. It is easily verified that the unshifted QR algorithm converges — actually it is stationary in this example — to a matrix whose eigenvalues are not in descending order. This phenomenon is called *disorder in the eigenvalues*.

It should be kept in mind that disorder is a creature of exact computations. Rounding error will generally cause the eigenvalues to be found in descending order. But the QR algorithm can be very slow in extricating itself from disorder.

- We can get further information about the behavior of the unshifted algorithm by partitioning the matrices involved in the proof of Theorem 2.2. Specifically, write

$$R(\Lambda^k L \Lambda^{-k}) = \begin{pmatrix} R_{11} & r_{1i} & R_{1,i+1} \\ 0 & r_{ii} & r_{i,i+1}^H \\ 0 & 0 & R_{i+1,i+1} \end{pmatrix} \begin{pmatrix} L_{11}^{(k)} & 0 & 0 \\ \ell_{i1}^{(k)H} & 1 & 0 \\ L_{i+1,1}^{(k)} & \ell_{i+1,i}^{(k)} & L_{i+1,i+1}^{(k)} \end{pmatrix}. \quad (2.17)$$

If $\ell_{i1}^{(k)H}$, $L_{i+1,1}^{(k)}$, and $\ell_{i+1,i}^{(k)}$ are zero, the product $R(\Lambda^k L \Lambda^{-k})(\Lambda^k U)$ has the form

$$R(\Lambda^k L \Lambda^{-k})(\Lambda^k U) = \begin{pmatrix} R_{11} L_{11}^{(k)} & r_{1i} & R_{1,i+1} L_{i+1,i+1} \\ 0 & r_{ii} & r_{i,i+1}^H L_{i+1,i+1} \\ 0 & 0 & R_{i+1,i+1} L_{i+1,i+1} \end{pmatrix} (\Lambda^k U).$$

This product is block upper triangular, and hence its Q-factor has the form

$$\hat{Q}_k = \text{diag}(\hat{Q}_{11}^{(k)}, \omega, \hat{Q}_{i+1,i+1}^{(k)}), \quad \omega = \bar{r}_{ii}/|r_{ii}|.$$

Now $A_k = \check{Q}_k^H A \check{Q}_k = \hat{Q}_k^H Q^H A Q \hat{Q}_k = \hat{Q}_k^H T \hat{Q}_k$, where T is the upper triangular factor of the Schur decomposition of A . It follows that

$$A_k = \begin{pmatrix} A_{11}^{(k)} & a_{1i}^{(k)} & A_{1,i+1}^{(k)} \\ 0 & \lambda_i & A_{i,i+1}^{(k)} \\ 0 & 0 & A_{i+1,i+1}^{(k)} \end{pmatrix}; \quad (2.18)$$

i.e., A_k decouples at its i th diagonal element.

The block triangularization in (2.18) occurs only when $\ell_{i1}^{(k)H}$, $L_{i+1,1}^{(k)}$, and $\ell_{i+1,i}^{(k)}$ in (2.17) are exactly zero. Now in the unshifted algorithm these matrices tend to zero, and by continuity A_k will tend toward a matrix of the form (2.18) at the same rate as these matrices approach zero. By inspection, we see that the most slowly converging elements are $\ell_{i,i-1}^{(k)}$ and $\ell_{i+1,i}^{(k)}$, which converge to zero as $|\lambda_i/\lambda_{i-1}|^k$ and $|\lambda_{i+1}/\lambda_i|^k$ respectively. It follows that:

Under the hypotheses of Theorem 2.2, the (i, i) -element of A_k converges to λ_i while the other elements of the submatrix $A_k[i:n, 1:i]$ converge to zero. The rate of convergence is at least as fast as the approach of $\max\{|\lambda_i/\lambda_{i-1}|, |\lambda_{i+1}/\lambda_i|\}^k$ to zero.

- Theorem 2.2 explicitly assumes that the eigenvalues of A are distinct and have distinct absolute values. However, by considering a suitable partitioning of the matrices in question, the proof of the theorem can be adapted to show that if there are groups of eigenvalues of equal magnitude, the QR iterates A_k tend to a block triangular form, whose diagonal blocks — which themselves may not converge — contain the equimodular eigenvalues.
-

The foregoing does not make an impressive case for the unshifted QR algorithm. Its convergence is unreliable; and even when the conditions of Theorem 2.2 are satisfied, the convergence can be slow. Nonetheless, the results on the unshifted algorithm give us insight into how the shifted algorithm behaves.

Specifically, suppose that the shifted algorithm is converging to an eigenvalue A , so that the last row is going swiftly to zero. As the shifts κ_k converge, we are in effect applying the unshifted algorithm to the matrix $A - \kappa_k I$. Hence we may expect some reduction in the subdiagonal elements of A other than those in the last row. This reduction may not be much, but for a large matrix, on which we must perform many iterations, its cumulative effect may be significant. This, in fact, is what happens in practice.

2.3. HESSENBERG FORM

We now turn to the practical implementation of the QR algorithm. The first point to note is that one step of the algorithm requires the computation of a QR decomposition

of A_k . The computation of this decomposition requires $O(n^3)$ floating-point operations. Since at least one QR step is needed to compute a single eigenvalue, the operation count to find all the eigenvalues is $O(n^4)$. (This is true even if we take advantage of the fact that as the matrix deflates, we work with ever smaller submatrices.) Such an operation count is far too high for the algorithm to be practical.

The cure to this problem is to first reduce the matrix A to a form in which the algorithm is cheaper to apply. The natural form, it turns out, is *upper Hessenberg form*; that is, a matrix of the form illustrated below for $n = 6$:

$$\begin{pmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & X & X \end{pmatrix}.$$

(This representation, in which an 0 stands for a zero element and an X stands for an element that may be nonzero, is called a *Wilkinson diagram*.) As we shall see later, a QR step preserves Hessenberg form. Moreover, a QR step for a Hessenberg matrix requires only $O(n^2)$ operations. This brings the total operation count down to an acceptable $O(n^3)$.

Householder transformations

The reduction to Hessenberg form is traditionally effected by a class of transformations called *Householder transformations*. Real Householder transformations have been widely treated in the literature and in textbooks (in particular, in §I:4.1 in this series). Here we will give a brief treatment of the basics, emphasizing the complex case.

A *Householder transformation* or *elementary reflector* is a matrix of the form

$$H = I - uu^H, \quad (2.19)$$

where

$$\|u\|_2 = \sqrt{2}.$$

It is easy to see that a Householder transformation is Hermitian and unitary.

The product of a Householder transformation and a matrix A can be cheaply computed. Specifically,

$$HA = (I - uu^H)A = A - u(u^H A).$$

This equation suggests the following algorithm, which overwrites A with HA .

1. $v^H = u^H A$
2. $A = A - uv^H$

If A is $m \times n$, this algorithm requires about $2mn$ flam. A similar algorithm will compute the product AH .

Householder transformations are useful because they can introduce zeros into a matrix. We begin with the special case of a vector.

Theorem 2.3. Let a be a vector such that $\|a\|_2 = 1$ and a_1 is real and nonnegative. Let

$$u = \frac{a + e_1}{\sqrt{1 + a_1}}.$$

Then

$$Ha = (I - uu^H)a = -e_1.$$

The proof of this theorem is purely computational and is left as an exercise.

The condition that the vector a have norm one and that its first component be non-negative is rather restrictive. However, to reduce a general vector a to a multiple of e_1 we can replace a by

$$\rho \frac{a}{\|a\|_2},$$

where ρ is a scalar of absolute value one that makes the first component of a nonnegative. This observation leads to Algorithm 2.1, which generates a Householder transformation that reduces a vector a to a multiple of e_1 . Five comments.

- If $a = 0$, the routine returns $u = \sqrt{2}$, which makes H the identity with its $(1, 1)$ -element changed to -1 .
- There is no danger of cancellation in statement 11 because the preceding statement forces $u[1]$ to be nonnegative.
- If a is a row vector, the algorithm returns a row vector u such that $a(I - u^H u) = \nu e_1^T$. Since we generally represent row vectors as a^H , u^H , and so on, we will always invoke *housegen* in the form

$$\text{housegen}(a^H, u^H, \nu).$$

In this case our Householder transformation is $I - uu^H$, as usual.

- If a is of order n and we replace the index 1 in *housegen* with n , we obtain a new function

$$\text{housegen2}(a, u, \nu) \tag{2.20}$$

that produces a Householder transformation $H = I - uu^H$ such that

$$Hu = \nu e_n.$$

This algorithm takes a vector a and produces a vector u that generates a Householder transformation $H = I - uu^H$ such that $Ha = \nu e_1$.

```

1. housegen( $a, u, \nu$ )
2.    $u = a$ 
3.    $\nu = \|a\|_2$ 
4.   if ( $\nu = 0$ )  $u[1] = \sqrt{2}$ ; return ; fi
5.   if ( $u[1] \neq 0$ )
6.      $\rho = \bar{u}[1]/|u[1]|$ 
7.   else
8.      $\rho = 1$ 
9.   end if
10.   $u = (\rho/\nu)*u$ 
11.   $u[1] = 1 + u[1]$ 
12.   $u = u/\sqrt{u[1]}$ 
13.   $\nu = -\bar{\rho}\nu$ 
14. end housegen
```

Algorithm 2.1: Generation of Householder transformations

For an example of *housegen2* applied to a row vector, see Algorithm 1.3, Chapter 5.

- When a is real, so is u . However, in this case such constructions as $\rho = \bar{u}[1]/|u[1]|$ are unnecessarily elaborate, and the generation of Householder transformations for real vectors is usually done somewhat differently (see Algorithm I:4.1.1). Nonetheless, in this volume we will regard *housegen* as a generic program, applicable to both real and complex vectors.

Reduction to Hessenberg form

We have already mentioned that the principal use of Householder transformations is to introduce zeros into a matrix. We will illustrate the technique with the reduction by unitary similarities of a matrix to Hessenberg form.

Let A be of order n and partition A in the form

$$A = \begin{pmatrix} \alpha_{11} & a_{12}^H \\ a_{21} & A_{22} \end{pmatrix}.$$

Let \hat{H}_1 be a Householder transformation such that

$$\hat{H}_1 a_{21} = \nu_1 e_1.$$

$$\begin{pmatrix} X & X & X & X & X \\ \hat{X} & X & X & X & X \\ \hat{X} & X & X & X & X \\ \hat{X} & X & X & X & X \\ \hat{X} & X & X & X & X \end{pmatrix} \xrightarrow{H_1} \begin{pmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & \hat{X} & X & X & X \\ 0 & \hat{X} & X & X & X \\ 0 & \hat{X} & X & X & X \end{pmatrix} \xrightarrow{H_2} \begin{pmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & \hat{X} & X & X \\ 0 & 0 & \hat{X} & X & X \end{pmatrix}$$

$$\xrightarrow{H_3} \begin{pmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{pmatrix}$$

Figure 2.1: Reduction to upper Hessenberg form

If we set $H_1 = \text{diag}(1, \hat{H}_1)$, then

$$H_1 A H_1 = \begin{pmatrix} \alpha_{11} & a_{12}^H \hat{H}_1 \\ \hat{H}_1 a_{21} & \hat{H}_1 A_{22} \hat{H}_1 \end{pmatrix} = \begin{pmatrix} \alpha_{11} & a_{12}^H \hat{H}_1 \\ \nu_1 \mathbf{e}_1 & \hat{H}_1 A_{22} \hat{H}_1 \end{pmatrix}.$$

Thus we have annihilated all the elements in the first column below the first subdiagonal.

For the general step, suppose that we have generated Householder transformations H_1, \dots, H_{k-1} such that

$$H_{k-1} \cdots H_1 A H_1 \cdots H_{k-1} = \begin{pmatrix} A_{11} & a_{1k} & A_{1,k+1}^H \\ 0 & \alpha_{kk} & a_{k,k+1}^H \\ 0 & a_{k+1,k} & A_{k+1,k+1} \end{pmatrix},$$

where A_{11} is a Hessenberg matrix of order $k - 1$. Let \hat{H}_k be a Householder transformation such that

$$\hat{H}_k a_{k+1,k} = \nu_k \mathbf{e}_1.$$

If we set $H_k = \text{diag}(I_k, \hat{H}_k)$, then

$$H_k H_{k-1} \cdots H_1 A H_1 \cdots H_{k-1} H_k = \begin{pmatrix} A_{11} & a_{1k} & A_{1,k+1}^H \hat{H}_k \\ & \alpha_{kk} & a_{k,k+1}^H \hat{H}_k \\ 0 & \nu_k \mathbf{e}_1 & \hat{H}_k A_{k+1,k+1} \hat{H}_k \end{pmatrix},$$

which reduces the elements below the subdiagonal of the k th column to zero.

Figure 2.1 illustrates the course of the reduction on a matrix of order five. The elements with hats are the ones used to generate the Householder transformation that

This algorithm reduces a matrix A of order n to Hessenberg form by Householder transformations. The reduced matrix is returned in the matrix H , and the transformations are accumulated in the matrix Q .

```

1. hessreduce( $A, H, Q$ )
    Reduce A
2.  $H = A$ 
3. for  $k = 1$  to  $n-2$ 
    Generate the transformation
4. housegen( $H[k+1:n, k]$ ,  $u$ ,  $H[k+1, k]$ )
5.  $Q[k+1:n, k] = u$ 
    Premultiply the transformation
6.  $v^H = u^H * H[k+1:n, k+1:n]$ 
7.  $H[k+1:n, k+1:n] = H[k+1:n, k+1:n] - u * v^H$ 
8.  $H[k+2:n, k] = 0$ 
    Postmultiply the transformation
9.  $v = H[1:n, k+1:n] * u$ 
10.  $H[1:n, k+1:n] = H[1:n, k+1:n] - v * u^H$ 
11. end  $k$ 
    Accumulate the transformations
12.  $Q[:, n] = \mathbf{e}_n$ ;  $Q[:, n-1] = \mathbf{e}_{n-1}$ 
13. for  $k = n-2$  to 1 by  $-1$ 
14.    $u = Q[k+1:n, k]$ 
15.    $v^H = u^H * Q[k+1:n, k+1:n]$ 
16.    $Q[k+1:n, k+1:n] = Q[k+1:n, k+1:n] - u * v^H$ 
17.    $Q[:, k] = \mathbf{e}_k$ 
18. end for  $k$ 
19. end hessreduce
```

Algorithm 2.2: Reduction to upper Hessenberg form

produces the next matrix. Note that after three (i.e., $n-2$) transformations, the reduction is complete.

Algorithm 2.2 implements the reduction to upper Hessenberg form. There are several comments to make about it.

- The matrix Q is the product $H_1 H_2 \cdots H_{n-2}$. The natural way to accumulate the transformations is to set $Q = I$, and then accumulate the transformations in the order

$$(\cdots(((QH_1)H_2)H_3)\cdots)H_{n-2}.$$

This has the advantage that the reduction and the accumulation can be done in a single loop. However, since H_1 is almost full, we will have to apply the subsequent H_k to the array $Q[1:n, k+1, n]$.

An alternative is to save the vectors generating the Householder transformations and accumulate them after the reduction in the order

$$H_1(\cdots(H_{n-4}(H_{n-3}(H_{n-2}Q)))\cdots).$$

It is easy to see that this order fills up Q from the southeast to the northwest, so that the application of H_k only affects $Q[k+1:n, k+1:n]$. In this implementation we have saved the generating vectors in Q itself and initialize Q to the identity column by column as the loop regresses.

- An operation count for the algorithm can be obtained as follows. The generation of the transformations is negligible. At the k th stage the premultiplication requires about $(n-k)^2$ floating-point additions and multiplications (flams) to compute v and a like number to update H . Thus the count for the premultiplication is about

$$2 \int_0^n (n-k)^2 dk = \frac{2}{3} n^3 \text{ flam.}$$

The postmultiplication at the k stage requires about $2n(n-k)$ flam, which gives a total of n^3 flam. The operation count for the accumulation of Q is the same as for the premultiplication. Hence:

The operation count for Algorithm 2.2 is

$$\frac{5}{3} n^3 \text{ flam for the reduction of } A$$

$$\frac{2}{3} n^3 \text{ flam for the accumulation of } Q$$

- As mentioned above, the term *flam* is an abbreviation for “floating-point addition and multiplication.” When A is real, the flam actually represents a floating-point addition and multiplication on the machine in question. When A is complex, however, additions and multiplications are composite operations. Specifically, a complex addition requires two ordinary floating-point additions, and a complex multiplication requires two additions and four multiplications. Thus in complex arithmetic the flam is

equivalent to four ordinary flams, and the operation counts above must be adjusted by a factor of four. In the sequel we will report operation counts in terms of the arithmetic operations — real or complex — in the algorithm and leave the conversion to machine operations to the reader.

- The algorithm generates H *in situ*. If we replace H by A , then the algorithm overwrites A with its Hessenberg form. Most packages choose this option to save storage.
- An extremely important observation is the following.

The first column of the orthogonal matrix Q generated by Algorithm 2.2 is \mathbf{e}_1 . (2.21)

This fact can be verified directly from the algorithm. We will use it later in deriving the implicit shift version of the QR algorithm.

- The algorithm is backward stable. Specifically, let \hat{H}_k be the exact Householder transformation generated from the computed A_k and let $\hat{Q} = \hat{H}_1 \cdots \hat{H}_{n-2}$. Then there is a matrix E satisfying

$$\frac{\|E\|_2}{\|A\|_2} = \gamma_n \epsilon_M$$

such that the computed Hessenberg matrix H satisfies

$$H = \hat{Q}^H (A + E) \hat{Q}.$$

Here γ_n is a slowly growing function of n .

In other words, the computed Hessenberg form is the exact Hessenberg form of $A + E$, where $\|E\|$ is of the order of the rounding unit compared to $\|A\|$. For the implications of this result see the discussion following Theorem 1.3.

The stability result just cited is typical of many algorithms that proceed by a sequence of orthogonal transformations. For later reference, we make the following definition.

Definition 2.4. Let the sequence A_1, A_2, \dots, A_m be generated by the recursion

$$A_{k+1} = \text{fl}[R_k A_k S_k], \quad k = 1, \dots, m-1, \quad (2.22)$$

where R_k and S_k are unitary matrices generated from A_k . Here fl denotes floating-point computation with rounding unit ϵ_M and includes any errors made in generating and applying R_k and S_k . Set

$$P = R_{m-1} \cdots R_1 \quad \text{and} \quad Q = S_1 \cdots S_{m-1}.$$

Then we say that the sequence (2.22) is STABLE IN THE USUAL SENSE if there is a matrix E and a slowly growing function γ_m such that

$$\frac{\|E\|_2}{\|A\|_2} = \gamma_m \epsilon_M$$

and

$$A_m = P(A_1 + E)Q.$$

Plane rotations

We have mentioned that a QR step leaves a Hessenberg matrix invariant. The proof is constructive and involves the use of 2×2 unitary transformations. We could easily use Householder transformations for this purpose, but it is more efficient to use another class of transformation — the plane rotations.

A *plane rotation* (also called a *Givens rotation*) is a matrix of the form

$$P = \begin{pmatrix} c & s \\ -\bar{s} & \bar{c} \end{pmatrix},$$

where

$$|c|^2 + |s|^2 = 1.$$

The transformation is called a rotation because in the real case its action on a two vector x is to rotate x clockwise through the angle $\theta = \cos^{-1} c$.

Plane rotations can be used to introduce zeros into a 2-vector. Specifically, let a and b be given with $a \neq 0$. If we set $\nu = \sqrt{|a|^2 + |b|^2}$ and

$$c = \frac{|a|}{\nu} \quad \text{and} \quad s = \frac{a}{|a|} \frac{\bar{b}}{\nu}, \tag{2.23}$$

then (note that c is real)

$$\begin{pmatrix} c & s \\ -\bar{s} & \bar{c} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \nu a / |a| \\ 0 \end{pmatrix}. \tag{2.24}$$

We can apply plane rotations to a matrix to introduce zeros into the matrix. Specifically, a *plane rotation in the (i, j) -plane* is a matrix of the form

$$P_{ij} = \begin{pmatrix} & & & i & & j & & & \\ & & & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & & & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & 0 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ i & 0 & \cdots & 0 & c & \cdots & s & 0 & \cdots & 0 \\ & \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ j & 0 & \cdots & 0 & -\bar{s} & \cdots & \bar{c} & 0 & \cdots & 0 \\ & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & \cdots & 0 \\ & \vdots & & \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

In other words, a rotation in the (i, j) -plane is an identity matrix in which a plane rotation has been embedded in the submatrix corresponding to rows and columns i and j .

To see the effect of a rotation in the (i, j) -plane on a matrix, let X be a matrix and let

$$Y = P_{ij}X.$$

If X and Y are partitioned by rows, then

1. $y_i^H = cx_i^H + sx_j^H$,
2. $y_j^H = cx_j^H - sx_i^H$,
3. $y_k^H = x_k^H$, whenever $k \neq i, j$.

Thus premultiplication by a rotation in the (i, j) -plane combines rows i and j and leaves the others undisturbed. By an appropriate choice of c and s we can introduce a zero anywhere we want in the i th or j th row.

Postmultiplication by P_{ij}^H affects the columns in a similar way, and we can introduce a zero anywhere in the i th or j th column.

Algorithm 2.3 generates a plane rotation satisfying (2.24). Four comments on this algorithm.

- If b is zero, then the routine returns the identity rotation. If a is zero, the routine returns the rotation that exchanges two components of a vector.
- The constant τ defined in statement 9 is a scaling factor designed to avoid overflows in statement 10 while rendering underflows harmless.
- For the sake of exposition we have stayed close to the formulas (2.23). These formulas can be considerably economized (for example, we could replace τ by $|\operatorname{Re}(a)| + |\operatorname{Im}(a)| + |\operatorname{Re}(a)| + |\operatorname{Im}(a)|$, thus avoiding some multiplications and a square root in statement 9). They can also be simplified when a and b are real.
- The final values of the application of the rotation to a and b are returned in a and b . Because our pseudocode can alter the arguments of routines this means that if a and b are elements of a matrix then those elements are reset to the values they would assume if the rotation were applied.
- A simpler and in many ways more natural definition of c and s is

$$c = \bar{a}/\nu \quad \text{and} \quad s = \bar{b}/\nu,$$

which makes the final value of a equal to ν . The definition we have chosen makes c real, which saves work when we apply the rotation to a matrix.

Since the application of a rotation to a matrix alters only two rows of the matrix, a simple routine to combine two vectors will serve to implement the general application

The following algorithm generates a plane rotation satisfying (2.24) from the quantities a and b . The cosine of the rotation is real. The program overwrites a with its final value and b with zero.

```

1.  rotgen( $a, b, c, s$ )
2.  if ( $b = 0$ )
3.     $c = 1; s = 0$ ; return
4.  end if
5.  If ( $a = 0$ )
6.     $c = 0; s = 1; a = b; b = 0$  return
7.  end if
8.   $\mu = a/|a|$ 
9.   $\tau = |a|+|b|$ 
10.  $\nu = \tau * \sqrt{|a/\tau|^2 + |b/\tau|^2}$ 
11.  $c = |a|/\nu$ 
12.  $s = \mu * \bar{b}/\nu$ 
13.  $a = \nu * \mu$ 
14.  $b = 0$ 
15. end rotgen

```

Algorithm 2.3: Generation of a plane rotation

This algorithm takes a plane rotation P specified by c and s and two vectors x and y and overwrites them with

$$P \begin{pmatrix} x^T \\ y^T \end{pmatrix}.$$

```

1.  rotapp( $c, s, x, y$ )
2.   $t = c*x+s*y$ 
3.   $y = c*y-\bar{s}*x$ 
4.   $x = t$ 
5.  end rotapp

```

Algorithm 2.4: Application of a plane rotation

of plane rotations. Algorithm 2.4 is such a routine. Here are some comments on the routine.

- The routine *rotapp* has many applications. Suppose that P is a rotation in the (i, j) -plane defined by c and s . Then the following table gives the calling sequences for four distinct applications of P to a matrix X .

$$\begin{aligned} P X &: \text{rotapp}(c, s, X[i, :], X[j, :]) \\ P^H X &: \text{rotapp}(c, -s, X[i, :], X[j, :]) \\ X P &: \text{rotapp}(c, -\bar{s}, X[:, i], X[:, j]) \\ X P^H &: \text{rotapp}(c, \bar{s}, X[:, i], X[:, j]) \end{aligned}$$

- The routine *rotapp* uses a vector t to contain intermediate values. A practical implementation would do something (e.g., write out the operations in scalar form) to insure that a storage allocator is not invoked every time *rotapp* is invoked.
 - Because the cosine is real, the cost of combining two components of x and y is $12 \text{ flm}t + 8 \text{ fladd}$. In the real case the count becomes $4 \text{ flm}t + 2 \text{ fladd}$. In some algorithms, we must multiply real rotations by complex vectors, in which case the count for each pair of elements is $8 \text{ flm}t + 4 \text{ fladd}$. We will often refer to these counts collectively as a *frot*, leaving context to decide the actual count.
-

Transformations that can introduce a zero into a matrix also have the power to destroy zeros that are already there. In particular, plane rotations are frequently used to move a nonzero element around in a matrix by successively annihilating it in one position and letting it pop up elsewhere — a process called bulge chasing. In designing algorithms like this, it is useful to think of the transformations as a game played on a Wilkinson diagram. Here are the rules for one step of the game.

Begin by selecting two rows of the matrix (or two columns):

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \hline X & \hat{X} & 0 & X & 0 \\ X & X & 0 & 0 & X \end{array}$$

Then after the transformation the situation is as follows.

1. The element of your choice becomes zero (\hat{X} in column 2).
2. A pair of X 's remains a pair of X 's (column 1).
3. A pair of 0's remains a pair of 0's (column 3).
4. A mixed pair becomes a pair of X 's (columns 4 and 5).

Thus our transformed pair of rows is

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \hline X & 0 & 0 & X & X \\ X & X & 0 & X & X \end{array}$$

In this example we have actually lost a zero element.

$$\begin{array}{c}
 \left(\begin{array}{ccccc} X & X & X & X & X \\ \hat{X} & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{array} \right) \xrightarrow{P_{12}} \left(\begin{array}{ccccc} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & \hat{X} & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{array} \right) \xrightarrow{P_{23}} \left(\begin{array}{ccccc} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & \hat{X} & X & X \\ 0 & 0 & 0 & X & X \end{array} \right) \\
 \xrightarrow{P_{34}} \left(\begin{array}{ccccc} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & \hat{X} & X \end{array} \right) \xrightarrow{P_{45}} \left(\begin{array}{ccccc} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right)
 \end{array}$$

Figure 2.2: Reduction to triangular form in QR step

————— ◊ —————

Invariance of Hessenberg form

The reason for reducing a matrix to Hessenberg form before applying the QR algorithm is that the form is invariant under a QR step and the operation count for performing a QR step is $O(n^2)$. To establish these facts, we will use plane rotations to implement the basic QR step. Since shifting the step consists only of altering the diagonal elements at the beginning and end of the step, we consider only the unshifted step.

Let H be the upper Hessenberg matrix in question. The QR step is divided into two parts. In the first part, we determine rotations $P_{12}, \dots, P_{n-1,n}$ with $P_{k,k+1}$ acting in the $(k, k+1)$ -plane so that $T = P_{n-1,n} \cdots P_{12}H$ is upper triangular. This step is illustrated by the Wilkinson diagrams in Figure 2.2. Note that if $Q^H = P_{n-1,n} \cdots P_{12}$, then Q is the Q-factor of H — i.e., the unitary Q of the QR algorithm.

The second step is to form $\hat{H} = TQ = TP_{12}^H \cdots P_{n-1,n}^H$, so that \hat{H} is the result of the single, unshifted QR step. Figure 2.3 illustrates this postmultiplication step. The point to observe is that the postmultiplication by $P_{i,i+1}^H$ introduces a nonzero in the $(i+1, i)$ -element but leaves the other zero elements undisturbed. Thus the final matrix \hat{H} is in Hessenberg form, and we have shown:

If a QR step is performed on an upper Hessenberg matrix, the result is upper Hessenberg. (2.25)

Algorithm 2.5 implements these two steps. Two comments are in order.

- The application of plane rotations dominates the calculation. It is easy to see that the algorithm requires about n^2 flops. Whether the matrix is real or complex, this translates into $O(n^2)$ operations. Thus we have shown that a QR step preserves upper Hessenberg form and requires $O(n^2)$ operations.
- The pre- and postmultiplications can be interleaved. Specifically, after the second

$$\begin{array}{c}
 \left(\begin{array}{cccccc} X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \xrightarrow{P_{12}^H} \left(\begin{array}{ccccc} X & X & X & X & X \\ X & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \xrightarrow{P_{23}^H} \left(\begin{array}{ccccc} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & 0 & X & X \\ p0 & 0 & 0 & 0 & X \end{array} \right) \\
 \xrightarrow{P_{34}^H} \left(\begin{array}{ccccc} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \xrightarrow{P_{45}^H} \left(\begin{array}{ccccc} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & 0 & X & X \end{array} \right)
 \end{array}$$

Figure 2.3: Back multiplication in QR step

This algorithm applies one unshifted QR step to an upper Hessenberg matrix H .

1. **for** $k = 1$ **to** $n-1$
2. **rotgen**($H[k, k]$, $H[k+1, k]$, c_k , s_k)
3. **rotapp**(c_k , s_k , $H[k, k+1:n]$, $H[k+1, k+1:n]$)
4. **end for** k
5. **for** $k = 1$ **to** $n-1$
6. **rotapp**(c_k , \bar{s}_k , $H[1:k+1, k]$, $H[1:k+1, k+1]$)
7. **end for** k

Algorithm 2.5: Basic QR step for a Hessenberg matrix

rotation (P_{23} in Figure 2.2) has been premultiplied we can postmultiply by P_{12}^H . The process continues with the generation and application of $P_{k,k+1}$ being followed by the application of $P_{k-1,k}^H$.

- The method is stable in the usual sense (Definition 2.4).

2.4. THE EXPLICITLY SHIFTED ALGORITHM

Algorithm (2.5) exists primarily to show the invariance of upper Hessenberg form under a QR step. In this subsection we will work our way to a practical algorithm for reducing an upper Hessenberg matrix H to Schur form.

Detecting negligible subdiagonals

From the convergence theory in §2.1 we have reason to expect that when the shifted QR algorithm is applied to an upper Hessenberg matrix H the element $h_{n,n-1}$ will tend rapidly to zero. At the same time, the results in §2.2 suggest that other subdiagonal elements may also tend to zero, albeit slowly. If any subdiagonal element can be regarded as negligible, then the problem deflates, and we can save computations. However, to claim this bonus, we must first decide what it means for a subdiagonal element to be negligible.

A natural criterion is to choose a small number ϵ and declare $h_{i+1,i}$ negligible provided

$$|h_{i+1,i}| \leq \epsilon \|A\|_F, \quad (2.26)$$

where A is the matrix we started with. To see why, let us examine the consequences of setting $h_{i+1,i}$ to zero. To simplify the discussion, we will assume that the current value \hat{A} of A has been computed without error, so that there is an orthogonal matrix Q such that $Q^H A Q = \hat{A}$. Now setting $h_{i+1,i}$ to zero amounts to replacing \hat{A} by $\tilde{A} = A - h_{i+1,i} \mathbf{e}_{i+1} \mathbf{e}_i^H$. If we set

$$E = Q(h_{i+1,i} \mathbf{e}_{i+1} \mathbf{e}_i^H) Q^H,$$

then

$$\tilde{A} = Q^H (A + E) Q.$$

Moreover, if $h_{i+1,i}$ satisfies (2.26), then

$$\frac{\|E\|_F}{\|A\|_F} \leq \epsilon. \quad (2.27)$$

In other words, setting $h_{k-1,k}$ to zero amounts to making a normwise relative perturbation in A of size ϵ .

When ϵ equals the rounding unit ϵ_M , the perturbation E is of a size with the perturbation due to rounding the elements of A . If the elements of A are roughly the same

size, then this is as small a perturbation as we can reasonably expect. For more on this point see the discussion following Theorem 1.3.

It is worth observing that we used the Frobenius norm in (2.26) only because its unitary invariance made the bound (2.27) easy to derive. In practice, one can replace $\|A\|_F$ by any reasonably balanced norm — say the 1- or ∞ -norm.

When the elements of A are not balanced — say when the matrix shows a systematic grading from larger to smaller elements as we pass from the northwest corner to the southeast — the backward error argument for the criterion (2.26) becomes less compelling. In such cases, the matrix frequently has small eigenvalues that are insensitive to small relative perturbations in its elements. In this case it is important that we set $h_{i-1,i}$ to zero only if it is small compared to the nearby elements of the matrix, something the criterion (2.26) does not insure. The usual cure for this problem is to adopt the criterion

$$|h_{i+1,i}| \leq (|h_{i,i}| + |h_{i+1,i+1}|)\epsilon. \quad (2.28)$$

If by chance $|h_{i,i}| + |h_{i+1,i+1}|$ is zero, one can revert to (2.26).

Deflation

Once a subdiagonal element has been deemed insignificant, we can deflate the problem. In the shifted QR algorithm subdiagonal elements will usually deflate beginning with the $(n, n-1)$ -element and proceeding backward one element at a time. As the algorithm progresses, however, subdiagonal elements in other positions may also deflate. A typical situation is illustrated by the following Wilkinson diagram:

$$\left(\begin{array}{ccc|cccc|ccc} h & h & h & d & d & d & d & h & h & h \\ h & h & h & d & d & d & d & h & h & h \\ 0 & h & h & d & d & d & d & h & h & h \\ \hline 0 & 0 & 0 & b & b & b & b & c & c & c \\ 0 & 0 & 0 & b & b & b & b & c & c & c \\ 0 & 0 & 0 & 0 & b & b & b & c & c & c \\ 0 & 0 & 0 & 0 & 0 & b & b & c & c & c \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & h & h & h \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h & h \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h \end{array} \right). \quad (2.29)$$

Here the seventh, eighth, and ninth subdiagonal elements have become negligible, revealing three eigenvalues at the eighth, ninth, and tenth diagonal elements. The third subdiagonal element has also become negligible, so that the remaining eigenvalues are contained in the leading principal submatrix of order three and in the principal submatrix of order four whose elements are labeled b . If we decide to treat the latter first, then we need only perform QR steps between rows and columns four and seven.

In applying the rotations, however, we must distinguish two cases. The first is when we only wish to find the eigenvalues of A . In that case we need only transform

This algorithm takes a Hessenberg matrix H of order n and an index ℓ ($1 < \ell \leq n$) and determines indices $i1, i2 \leq \ell$ such that one of the following two conditions holds.

1. $1 \leq i1 < i2 \leq \ell$, in which case the matrix deflates at rows $i1$ and $i2$.
 2. $1 = i1 = i2$, in which case all the subdiagonal elements of H in rows 1 through ℓ are zero.
-
1. ***backsearch***($H, \ell, i1, i2$)
 2. $i1 = i2 = \ell$
 3. **while** ($i1 > 1$)
 4. **if** ($H[i1, i1-1]$ is negligible)
 5. $H[i1, i1-1] = 0$
 6. **if** ($i1 = i2$)
 7. $i2 = i1 = i1-1$
 8. **else**
 9. **leave while**
 10. **end if**
 11. **else**
 12. $i1 = i1-1$
 13. **end if**
 14. **end while**
 15. **end** *backsearch*

Algorithm 2.6: Finding deflation rows in an upper Hessenberg matrix

the elements labeled b in (2.29). On the other hand, if we wish to compute a Schur decomposition of A , we must apply the transformations to the entire matrix. In particular, the transformations must be premultiplied into the elements labeled b and c and postmultiplied into the elements labeled b and d . In this case the transformations must also be accumulated into the matrix Q from the initial reductions to Hessenberg form. For details see Algorithm 2.8.

Back searching

In (2.29) we can determine by inspection the rows — call them $i1$ and $i2$ — that define where the QR step is to be applied. In practice, we must determine them on the fly by applying whatever criterion we choose to decide if a subdiagonal element is zero. This is usually done by searching back along the subdiagonal from the current value of $i2$. Algorithm 2.6 implements such a loop. It either returns distinct indices $i1$ and $i2$, in which case the QR step is to be taken between these indices; or the matrix is completely deflated, and the QR algorithm is finished. Negligible subdiagonal elements are set to

zero as they are detected.

The Wilkinson shift

Once we have found a range $[i1, i2]$ within which to perform a QR step, we must choose a shift. The natural candidate is the Rayleigh-quotient shift $\kappa = H[i2, i2]$, since it gives local quadratic convergence to simple eigenvalues. However, if H is real, the Rayleigh-quotient shift is also real and cannot approximate a complex eigenvalue. The cure is to use the *Wilkinson shift*, which is defined as the eigenvalue of the matrix

$$\begin{pmatrix} H[i2-1, i2-1] & H[i2-1, i2] \\ H[i2, i2-1] & H[i2, i2] \end{pmatrix}$$

that is nearest $H[i2, i2]$. This shift can move the QR iteration into the complex plane. Moreover, as the method converges — that is, as $H[i2, i2-1]$ approaches zero, the shift approaches the Rayleigh-quotient shift so that we retain the quadratic convergence of the latter.

The computation of the shift requires the solution of a 2×2 eigenvalue problem. The procedure is to compute the eigenvalues from the characteristic polynomial using the usual formula for the roots of a quadratic equation. However, some care must be taken to insure the stability of the resulting algorithm. To simplify our notation, we will consider the matrix

$$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Thus we want the eigenvalue of B nearest to d .

The problem simplifies if we work with the shifted matrix

$$B - dI = \begin{pmatrix} a - d & b \\ c & 0 \end{pmatrix}.$$

The Wilkinson shift then becomes $\kappa = \lambda + d$, where λ is the smallest eigenvalue of $B - dI$.

The characteristic equation of $B - dI$ is

$$\lambda^2 - (a - d)\lambda - bc \equiv \lambda^2 - 2p\lambda - q = 0.$$

The roots of this equation are

$$\lambda = p \pm \sqrt{p^2 + q} \equiv p \pm r.$$

To avoid cancellation we should not compute the smallest root directly from this formula. Instead we should compute the largest root λ_{\max} first and then compute the smallest root λ_{\min} from the relation $\lambda_{\min}\lambda_{\max} = -q$. To determine the largest root, note that

$$|\lambda|^2 = |p \pm r|^2 = |p|^2 \pm 2 \operatorname{Re}(p\bar{r}) + |r|^2. \quad (2.30)$$

Given the elements of the matrix

$$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

wilkshift returns the eigenvalue of B that is nearest d .

1. *wilkshift*(a, b, c, d, κ)
2. $\kappa = d$
3. $s = |a| + |b| + |c| + |d|$
4. **if** ($s = 0$) **return** **fi**
5. $q = (b/s)*(c/s)$
6. **if** ($q \neq 0$)
7. $p = 0.5*((a/s) - (d/s))$
8. $r = \sqrt{p^2 + q}$
9. **if** ($\text{Re}(p)*\text{Re}(r) + \text{Im}(p)*\text{Im}(r) < 0$) $r = -r$ **fi**
10. $\kappa = \kappa - s*(q/(p+r))$
11. **end if**
12. **end** *wilkshift*

Algorithm 2.7: Computation of the Wilkinson shift

Consequently we want to choose the sign \pm so that the middle term in (2.30) is positive.

Algorithm 2.7 implements the computation described above. To avoid overflow, the algorithm scales B so that its largest element is one in magnitude. This scaling factor can be replaced by the sum of the absolute values of the real and imaginary parts of the elements of B , which is cheaper to compute.

Reduction to Schur form

We can now assemble the algorithmic fragments developed previously into an algorithm for the unitary triangularization of an upper Hessenberg matrix. Algorithm 2.8 computes the Schur form of an upper Hessenberg matrix by the QR algorithm. Here are some comments on the algorithm.

- The routine *hqr* combined with *hessreduce* (Algorithm 2.2) gives an algorithm for computing the Schur form of a general matrix.
- The routine *hqr* is stable in the usual sense (Definition 2.4), provided an appropriate deflation strategy is used.
- The convergence theory developed in §2.1 says that once *hqr* starts converging to a simple eigenvalue, it will converge quadratically. Unfortunately, there is no theory

Given an upper Hessenberg matrix H , hqr overwrites it with a unitarily similar triangular matrix whose diagonals are the eigenvalues of H . The similarity transformation is accumulated in Q . The routine gives an error return if more than $maxiter$ iterations are required to deflate the matrix at any eigenvalue.

```

1.  hqr( $H, Q, maxiter$ )
2.   $i1 = 1; i2 = n$ 
3.   $iter = 0$ 
4.  while (true)
5.     $iter = iter + 1$ 
6.    if ( $iter > maxiter$ ) error return; fi
7.     $oldi2 = i2$ 
8.    backsearch( $H, i2, i1, i2$ )
9.    if ( $i2 = 1$ ) leave hqr fi
10.   if ( $i2 \neq oldi2$ )  $iter = 0$  fi
11.   wilkshift( $H[i2-1, i2-1], H[i2-1, i2],$ 
                   $H[i2, i2-1], H[i2, i2], \kappa$ )
12.    $H[i1, i1] = H[i1, i1] - \kappa$ 
13.   for  $i = i1$  to  $i2-1$ 
14.     rotgen( $H[i, i], H[i+1, i], c_i, s_i$ )
15.      $H[i+1, i+1] = H[i+1, i+1] - \kappa$ 
16.     rotapp( $c_i, s_i, H[i, i+1:n], H[i+1, i+1:n]$ )
17.   end for  $i$ 
18.   for  $i = i1$  to  $i2-1$ 
19.     rotapp( $c_i, \bar{s}_i, H[1:i+1, i], H[1:i+1, i+1]$ )
20.     rotapp( $c_i, \bar{s}_i, Q[1:n, i], Q[1:n, i+1]$ )
21.      $H[i, i] = H[i, i] + \kappa$ 
22.   end for  $i$ 
23.    $H[i2, i2] = H[i2, i2] + \kappa$ 
24. end while
25. end hqr
```

Algorithm 2.8: Schur form of an upper Hessenberg matrix

that says that the algorithm is globally convergent. Typically, *hqr* takes several iterations to deflate the first eigenvalue, somewhat less to compute the second eigenvalue, and so on. Eventually the subdiagonals of H become small enough so that quadratic convergence is seen from the outset. Still later, if the matrix is large enough, the matrix deflates at the upper end, so that $i1 > 0$, which gives further savings.

- To guard against the possibility of nonconvergence, *hqr* gives an error return when in the course of computing any one eigenvalue a maximum iteration count is exceeded. Most real-life implementations do not return immediately but instead try an ad hoc shift to get the algorithm back on track. For more see §2.5.
- The operation count is dominated by the application of plane rotations. Given $i1 < i2$, the pre- and postmultiplication of H requires about $n(i2 - i1)$ flrot. The accumulation of the transformations in Q requires the same amount of work to bring the total to $2n(i2 - i1)$ flrot.

An overall operation count depends on the number of iterations to convergence and the behavior of the ranges $[i1, i2]$, neither of which is known a priori. We can derive an upper bound by assuming that $i1 = 1$ and that it takes k iterations to find an eigenvalue. The bound is approximately

$$\int_0^n 2kni \, di = kn^3 \text{ flrot.} \quad (2.31)$$

- If we are only interested in computing the eigenvalues of H , we may restrict the range of our transformations to the elements labeled b in the Wilkinson diagram (2.29). Moreover, we no longer need to accumulate the transformations. This reduces the operation count for a QR step in the range $[i1, i2]$ to $2(i2 - i1)^2$ flrot, and the upper bound (2.31) becomes $\frac{2}{3}kn^3$ flrot.

Eigenvectors

We have now seen how to compute the Schur decomposition of a general matrix A . In some applications a Schur decomposition is all we need. In most applications, however, what is wanted is not the Schur vectors but the eigenvectors of A . It should be stressed that the eigenvalue-eigenvector decomposition is not a stable decomposition—in fact, if A is defective it does not exist. Nonetheless, the demand for eigenvectors is so great that all packages have an option for their computation.

If $A = QTQ^H$ is the Schur decomposition of A and X is the matrix of right eigenvectors of T , then the matrix of right eigenvectors of A is QX . Thus we can compute the eigenvectors of A by computing the eigenvectors of T and transforming them by Q .

We have already derived a formula for eigenvectors of a triangular matrix (see p. 3). Specifically, if we partition

$$T = \begin{pmatrix} T_{11} & t_{1k} & t_{1,k+1} \\ 0 & T_{kk} & t_{k,k+1}^H \\ 0 & 0 & T_{k+1,k+1} \end{pmatrix}$$

and τ_{kk} is a simple eigenvalue of T , then

$$\begin{pmatrix} -(T_{11} - \tau_{kk}I)^{-1}t_{1k} \\ 1 \\ 0 \end{pmatrix}$$

is an eigenvector of T . It follows that if we write the k th eigenvector in the form

$$\begin{pmatrix} x_1^{(k)} \\ 1 \\ 0 \end{pmatrix}$$

then we can obtain $x_1^{(k)}$ by solving the upper triangular system

$$(T_{11} - \tau_{ii}I)x_1^{(k)} = -t_{1k}. \quad (2.32)$$

To derive an algorithm for solving (2.32), let us drop subscripts and superscripts and consider a general algorithm for solving the equation $(T - \mu I)x = b$. If we partition this equation in the form

$$\begin{pmatrix} T_* - \mu I & t_* \\ 0 & \tau - \mu \end{pmatrix} \begin{pmatrix} x_* \\ \xi \end{pmatrix} = \begin{pmatrix} b_* \\ \beta \end{pmatrix}, \quad (2.33)$$

then the second row of the partition gives the equation $(\tau - \mu)\xi = \beta$, from which we get

$$\xi = \frac{\beta}{(\tau - \mu)}.$$

From the first row of (2.33) we have $(T_* - \mu I)x_* + \xi t_* = b_*$, whence

$$(T_* - \mu I)x_* = b_* - \xi t_*.$$

This is a system of order one less than the original, which we further reduce by a recursive application of the same procedure. All this gives the following algorithm.

1. $x = b$
2. **for** $j = n$ **to** 1 **by** -1
3. $x[j] = x[j]/(T[j, j] - \mu)$
4. $x[1:j-1] = x[1:j-1] - x[j]*T[1:j-1, j]$
5. **end for** k

This is essentially the algorithm we will use to compute eigenvectors. However, it has two drawbacks.

First, in our final algorithm the shift μ will be a diagonal of our original matrix T , as in (2.32). If T has multiple eigenvalues, then some diagonal of T_{11} may be equal to τ_{kk} , in which case our algorithm will terminate attempting to divide by zero. The cure

for this problem is to not allow $T[k, k] - \mu$ to be too small. In Algorithm 2.9 below we restrict it to be greater than $\epsilon_M \mu$, unless that number is itself too small, in which case we replace it with a number whose reciprocal is safely below the overflow point.

Many people are somewhat nervous about replacing an essentially zero quantity by an arbitrary quantity of an appropriate size. However, by projecting the error made back to the original matrix A [see the discussion leading to (2.27)], we see that the replacement corresponds to a small perturbation in A . This means that for the procedure to introduce great inaccuracies in the computed eigenvector, the eigenvector itself must be sensitive to small perturbations in A — i.e., it must be ill conditioned.

The second problem is that in some applications eigenvectors can have elements of widely varying sizes. In such cases, the attempt to compute the eigenvectors can result in overflow. Since simple eigenvectors are determined only up to a scalar factor, we are free to scale the solution. In particular, if a component of the solution is in danger of overflowing, we may scale the vector so that the offending component has modulus one. This may result in the underflow of very small components; however, if underflows are set to zero, the results will generally be satisfactory.

Algorithm 2.9 computes the right eigenvectors of a triangular matrix. Here are some comments.

- As we mentioned above, the diagonal element is adjusted to keep the divisor d from becoming too small. The strategy is a variant of one found in LAPACK, in which *smallnum* is taken to be $\frac{n}{\epsilon_M} \text{underflow}$, where *underflow* is a number just above the underflow point.
- If there is no rescaling, the algorithm has an operation count of $\frac{1}{6}n^3 \text{flam}$.
- The cost of rescaling can be comparatively large. For example, if scaling is done at every step in the computation of every eigenvector, the cost is $\frac{1}{3}n^3 \text{flmt}$, which is larger than the cost of the unscaled algorithm. For this reason the algorithm rescales only when there is a serious danger of overflow — specifically, when the component to be computed would be bigger than a number near the overflow point of the computer.
- The algorithm is column oriented in the sense that the matrix T is traversed along its columns. This orientation is to be preferred for programming languages like FORTRAN that store arrays by columns. There is a row-oriented version of the basic algorithm for solving triangular systems, which can be adapted to compute right eigenvectors. This algorithm is to be preferred in languages like C that store their arrays by rows. For a further discussion of these matters see §I:2.3.
- The algorithm is stable in the following sense. Each eigenvector x_i satisfies

$$(T + E_i)x_i = t_{ii}x_i, \quad (2.34)$$

where $\|E_i\|/\|T\| \leq \gamma \epsilon_M$ for some modest constant γ . This implies that if the eigenvector is not ill conditioned then it will be accurately computed. It is worth remem-

Given an upper triangular matrix T , this routine returns an upper triangular matrix X of the right eigenvectors of X . The columns of X are normalized to have 2-norm one.

```

1.  righteigvec( $T$ ,  $X$ )
2.   $smallnum =$  a small number safely above the underflow point
3.   $bignum =$  a number near the overflow point
4.  for  $k = n$  to 1 by  $-1$ 
5.     $X[1:k-1, k] = -T[1:k-1, k]$ 
6.     $X[k, k] = 1$ 
7.     $X[k+1:n, k] = 0$ 
8.     $dmin = \max\{\epsilon_M * |T[k, k]|, smallnum\}$ 
9.    for  $j = k-1$  to 1 by  $-1$ 
10.        $d = T[j, j] - T[k, k]$ 
11.       if ( $|d| \leq dmin$ )  $d = dmin$  fi
12.       if ( $|X[j, k]| / bignum \geq |d|$ )
13.           $s = |d| / |X[j, k]|$ 
14.           $X[1:k, k] = s * X[1:k, k]$ 
15.       end if
16.        $X[j, k] = X[j, k] / d$ 
17.        $X[1:j-1, k] = X[1:j-1, k] - X[j, k] * T[1:j-1, j]$ 
18.     end for  $j$ 
19.      $X[1:k, k] = X[1:k, k] / \|X[1:k, k]\|_2$ 
20.   end for  $k$ 
21. end righteigvec
```

Algorithm 2.9: Right eigenvectors of an upper triangular matrix

bering that E_i is different for each eigenvector x_i , so that as a whole the eigenvalue-eigenvector decomposition is not necessarily stable.

- It follows from (2.34) that the residual $r_i = Tx_i - t_{ii}x_i$ satisfies

$$\frac{\|r_i\|}{\|T\|\|x_i\|} \leq \gamma \epsilon_M. \quad (2.35)$$

Thus the norm of the scaled residual for each computed eigenvector is near the rounding unit. This fact, which continues to hold for the eigenvectors of the original matrix, provides an easy diagnostic for routines that purport to compute eigenvectors: compare the scaled residual against the rounding unit.

- Although we have chosen to compute the entire upper triangular array of eigenvectors, essentially the same algorithm could be used to compute a few selected eigenvectors.

A similar algorithm can be derived to compute left eigenvectors. Specifically if we partition T as usual in the form

$$T = \begin{pmatrix} T_{11} & t_{1k} & t_{1,k+1} \\ 0 & \tau_{kk} & t_{k,k+1}^H \\ 0 & 0 & T_{k+1,k+1} \end{pmatrix}$$

and assume that τ_{kk} is a simple eigenvector of T , then

$$(0 \ 1 \ -t_{k,k+1}^H(T_{k+1,k+1} - \tau_{kk}I)^{-1})$$

is a left eigenvector of T corresponding to τ_{kk} . Thus we can also find left eigenvectors by solving triangular systems.

However, there is an alternative. If X is the matrix of right eigenvectors of a diagonal matrix, then $Y^H = X^{-1}$ is the matrix of left eigenvectors. Thus we can also obtain the left eigenvectors of a matrix by inverting the matrix of right eigenvectors.

Unfortunately, the methods are not numerically equivalent. If we compute the left eigenvectors separately, the resulting matrix may not be the inverse of the matrix of right eigenvectors — we say that the right and left eigenvectors have lost biorthogonality. If biorthogonality is required, then directly inverting the matrix of right eigenvectors will give better results. But this will not work in cases where we want only a few left and right eigenvectors.

2.5. BITS AND PIECES

The algorithm sketched in the last subsection is an effective method for finding the eigenvalues (and eigenvectors, if desired) of a general complex matrix. However, there are modifications and extensions that can improve the performance of the algorithm. A detailed treatment of these topics would lengthen this chapter beyond reasonable

bounds. Instead, we will give a brief survey of the following topics: the ad hoc shift, deflation at consecutive small subdiagonal elements, sorting for eigenvalues, and balancing. The subsection concludes with a discussion of graded matrices.

Ad hoc shifts

Convergence of the Hessenberg QR algorithm is not guaranteed, even with the Wilkinson shift. The problem usually manifests itself by the algorithm exceeding a preset limit on the number of iterations (see statement 6 in Algorithm 2.8). In such an event the natural action is to give an error return. An alternative is to restart the iteration with a new shift, commonly referred to as an ad hoc shift. If that fails one can either try again or give an error return. It does not make much sense to try more than one or two ad hoc shifts.

There is little general theory to guide the choice of ad hoc shifts (which is why they are called ad hoc). The LAPACK routine CGEEV takes the shift to be $|\operatorname{Re}(h_{i,i-1})| + |\operatorname{Re}(h_{i-1,i-2})|$, where i is the row where the problem is supposed to deflate.

Aggressive deflation

If two consecutive subdiagonals of H are small but neither is small enough to permit deflation, we can sometimes start the QR step at the diagonal to the right of the first small element. Specifically, consider the Wilkinson diagram

$$\begin{array}{c} i \\ \downarrow \\ \begin{matrix} x & x & x & x \\ i \rightarrow 0 & \epsilon_1 & d & x \\ 0 & 0 & \epsilon_2 & x \\ 0 & 0 & 0 & x \end{matrix} \end{array}$$

where the upper-left-hand x represents the $(i-2, i-1)$ -element of H . For purposes of illustration we will assume that the elements in this diagram are real and that d is positive. Suppose we begin the QR step at row i . After the reduction to triangular form, we have the following diagram.

$$\begin{array}{c} i \\ \downarrow \\ \begin{matrix} x & x & x & x \\ i \rightarrow 0 & c\epsilon_1 & \nu & x \\ 0 & s\epsilon_1 & 0 & x \\ 0 & 0 & 0 & x \end{matrix} \end{array}$$

where

$$\nu = \sqrt{d^2 + \epsilon_2^2}, \quad c = \frac{d}{\nu}, \quad \text{and} \quad s = \frac{\epsilon_1}{\nu}.$$

The postmultiplication by the rotations does not change elements in the second column.

Now suppose that ϵ_2 is so small that $\text{fl}(d^2 + \epsilon_2^2) = d^2$. Then the computed value of c will be one, and the computed value of the element ce_1 will simply be ϵ_1 — unchanged from the original. Moreover, the computed value of the element $s\epsilon_1 = \epsilon_1\epsilon_2/\nu$ is $\epsilon_1\epsilon_2/d$. Consequently, if the product $\epsilon_1\epsilon_2$ is so small that it is negligible by whatever deflation criterion is in force, we can ignore it. Thus for ϵ_1 and ϵ_2 sufficiently small, we can assume that the matrix is deflated at the element ϵ_1 — at least for the purpose of starting a QR step.

Note that it is the square of ϵ_1 and the product $\epsilon_1\epsilon_2$ that says whether we can start a QR step. Since the product of two small numbers is much smaller than either, we may be able to start the QR step even when ϵ_1 or ϵ_2 are not small enough to deflate the matrix.

Cheap eigenvalues

If the original matrix A is highly structured, it is possible that some eigenvalues are available almost for free. For example, if A has the form

$$\begin{pmatrix} a & a & a & a & a \\ 0 & a & 0 & 0 & 0 \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \end{pmatrix},$$

then the single nonzero element in the second row is an eigenvalue. It takes at most n^2 comparisons to isolate such a row, if it exists. At an additional cost of $O(n^2)$ operations, this matrix can be permuted so that it has the form

$$\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & 0 & 0 & 0 & a \end{pmatrix}.$$

We can then repeat the process on the leading principal matrix of order $n-1$, and so on until it fails to isolate an eigenvalue. The result is a permutation of A that has the form

$$\begin{pmatrix} \hat{A} & B \\ 0 & T \end{pmatrix},$$

where T is upper triangular.

Turning now to \hat{A} , if it has, say, the form

$$\begin{pmatrix} a & a & 0 & a & a \\ a & a & 0 & a & a \\ a & a & a & a & a \\ a & a & 0 & a & a \\ a & a & 0 & a & a \end{pmatrix},$$

then it can be permuted to the form

$$\begin{pmatrix} a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix}.$$

Proceeding as above, we can permute A into the form

$$\begin{pmatrix} S & C & D \\ 0 & \tilde{A} & E \\ 0 & 0 & T \end{pmatrix},$$

where S is upper triangular. Thus we have deflated the problem at both ends.

Properly implemented the process requires only $O(n^2)$ operations per eigenvalue, the same as for a single QR step. Because of its cheapness, the process is usually made an option in matrix packages.

Balancing

Many applications generate matrices whose elements differ greatly in size. In some cases these disparities are a consequence of the structure of the underlying problem. In others, however, they are accidental, caused, for example, by a poor choice of units of measurement. In this case balance can often be restored by the following procedure.

We proceed by balancing the elements of a matrix one row and column at a time. Let us consider the case of the first row and column. Let $\rho \neq 0$, and let

$$D = \text{diag}(\rho, 1, 1, \dots, 1).$$

Then (for $n = 4$)

$$DAD^{-1} = \begin{pmatrix} a_{11} & \rho a_{12} & \rho a_{13} & \rho a_{14} \\ \rho^{-1} a_{21} & a_{22} & a_{23} & a_{24} \\ \rho^{-1} a_{31} & a_{32} & a_{33} & a_{34} \\ \rho^{-1} a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}.$$

If we now require that ρ be chosen so that the first row and column are balanced in the sense that

$$\rho \sum_{j \neq 1} |a_{1j}| = \rho^{-1} \sum_{i \neq 1} |a_{i1}|,$$

then we find that

$$\rho^2 = \frac{\sum_{i \neq 1} |a_{i1}|}{\sum_{j \neq 1} |a_{1j}|}.$$

Having balanced the first row and column, we can go on to balance the second, the third, and so on, cycling back to the first if necessary. Although the balancing of one row and column undoes the balancing of the others, the method can be shown to converge to a matrix in which all the rows and columns are balanced.

The balancing procedure is relatively inexpensive. Each step of the iteration requires only $O(n)$ operations. Moreover, a very loose convergence criterion will be sufficient to effectively balance the matrix. Balancing often improves the accuracy of computed eigenvalues and eigenvectors. However, at times it can make things worse. For this reason, any package that incorporates balancing should include an option to turn it off.

Graded matrices

Loosely speaking, a matrix is graded if it shows a systematic decrease or increase in its elements as one proceeds from one end of the matrix to the other. For example, the matrix

$$A = \begin{pmatrix} -1.5940e+00 & -3.9990e-09 & 7.1190e-17 \\ -1.4410e-08 & 6.9000e-17 & 1.2900e-24 \\ 5.7110e-17 & 8.1560e-25 & 6.6860e-33 \end{pmatrix}$$

exhibits a symmetric grading from large to small as we proceed along the diagonals. We therefore say that A is *graded downward by diagonals*. If we set

$$D = \text{diag}(1, 10^{-8}, 10^{-16}),$$

then

$$D^{-1}AD = \begin{pmatrix} -1.5940e+00 & -3.9990e-17 & 7.1190e-33 \\ -1.4410e+00 & 6.9000e-17 & 1.2900e-32 \\ 5.7110e-01 & 8.1560e-17 & 6.6860e-33 \end{pmatrix}$$

is *graded downward by rows*, while

$$DAD^{-1} = \begin{pmatrix} -1.5940e+00 & -3.9990e-01 & 7.1190e-01 \\ -1.4410e-16 & 6.9000e-17 & 1.2900e-16 \\ 5.7110e-33 & 8.1560e-33 & 6.6860e-33 \end{pmatrix}$$

is *graded downward by columns*.

It turns out that the QR algorithm usually works well with matrices that are graded downward by diagonals. For example, the computed eigenvalues of A are

$$\begin{aligned} & 5.1034150572761617e-33 \\ & 1.0515156210790460e-16 \\ & -1.5940000000000001e+00 \end{aligned}$$

which are almost fully accurate. On the other hand, the algorithm does not do as well with other forms of grading. For example, reverse the rows and columns of A to get the matrix

$$B = \begin{pmatrix} 6.6860e-33 & 8.1560e-25 & 5.7110e-17 \\ 1.2900e-24 & 6.9000e-17 & -1.4410e-08 \\ 7.1190e-17 & -3.9990e-09 & -1.5940e+00 \end{pmatrix},$$

which is graded upward by diagonals. If we compute the eigenvalues of B , we get the following results.

$$\begin{aligned} & 6.685999999999994e-33 \\ & 1.0515156534681920e-16 \\ & -1.5940000000000005e+00 \end{aligned}$$

The relative errors in these eigenvalues are

$$3.1e-01, \quad 3.1e-08, \quad 2.8e-16,$$

from which it is seen that the smallest eigenvalue has no accuracy at all, the second smallest is accurate to only eight decimal digits, and the largest is correct to working precision.

Row and column grading can also cause the algorithm to fail. The computed eigenvalues of $D^{-1}AD$ have relative errors of

$$1.0e+16, \quad 2.3e+00, \quad 1.4e-16,$$

while the computed eigenvalues of DAD^{-1} have relative errors of

$$1.3e+00, \quad 3.4e-01, \quad 0.0e+00.$$

The two smaller eigenvalues of both matrices have no accuracy.

There is no formal analysis of why the algorithm succeeds or fails on a graded matrix. However, inspection of special cases will often show that the algorithm causes a loss of important information by combining large and small elements. For example, consider the first column of B and the corresponding first column of the Hessenberg form of B :

$$\begin{array}{ll} 6.685999999999994e-33 & 6.685999999999994e-33 \\ 1.290000000000000e-24 & -7.119000000000004e-17 \\ 7.118999999999992e-17 & 0.000000000000000e+00 \end{array}$$

This display shows that the first column of the Hessenberg form is the same as the one obtained by first setting the element $1.290e-24$ to zero. This is a small error compared to the norm of B , but the smallest eigenvalue of B is quite sensitive to it. In particular the relative errors in the eigenvalues of the altered matrix B are

$$1.6e+00, \quad 2.3e-16, \quad 0.0e+00.$$

Balancing can help in this case. For example, balancing DAD^{-1} and $D^{-1}AD$ restores the lost accuracy. On the other hand, balancing B has no effect, since it cannot change the direction of the grading. This suggests the following rule of thumb.

When dealing with a graded matrix arrange the grading downward by rows, columns, or diagonals, and balance the matrix. (2.36)

We do not claim that this strategy is foolproof, and therefore anyone employing it should take a hard look at the results to see if they make sense.

2.6. NOTES AND REFERENCES

Historical

The lineage of the QR algorithm can be traced back to a theorem of König [151, 1884], which states that if the function defined by the power series

$$a_0 + a_1 z + a_2 z^2 + \dots \quad (2.37)$$

has one simple pole r at its radius of convergence then $r = \lim_{k \rightarrow \infty} a_k/a_{k+1}$. Later Hadamard [106, 1892] gave determinantal expressions in terms of the coefficients of (2.37) for poles beyond the radius of convergence. Aitken [1, 1926] independently derived these expressions and gave recurrences for their computation. Rutishauser consolidated and extended the recurrences in his qd-algorithm [222, 223, 1954]. (For a systematic treatment of this development see [122].)

Rutishauser [224, 1955] then showed that if the numbers from one stage of the qd-algorithm were arranged in a tridiagonal matrix T then the numbers for the next stage could be found by computing the LU factorization of T and multiplying the factors in reverse order. He generalized the process to a full matrix and showed that under certain conditions the iterates converged to an upper triangular matrix whose diagonals were the eigenvalues of the original matrix. The result was what is known as the LR algorithm. Later Rutishauser introduced a shift to speed convergence [225, 1958].

Except for special cases, like positive definite matrices, the LU algorithm is numerically unstable. Kublanovskaya [154, 1961] and Francis [76, 1961] independently proposed substituting the stable QR decomposition for the LU decomposition. However, Francis went beyond a simple substitution and produced the algorithm that we use today, complete with a preliminary reduction to Hessenberg form and the implicit double shift for real matrices.

The name of the QR decomposition comes from the fact that in Francis's notation the basic step was to decompose the matrix A into the product QR of an orthogonal matrix and an upper triangular matrix. Thus the algorithm preceded the decomposition. I have heard that the Q was originally O for "orthogonal" and was changed to Q to avoid confusion with zero.

For more historical material see Parlett's survey [204] and the paper [73] by Fernando and Parlett.

Variants of the QR algorithm

There are four factorizations of a matrix into a unitary and a triangular matrix, which we may represent symbolically by QR, QL, RQ, and LQ, where R and L stand for upper and lower triangular. The theory of the QR algorithm, *mutatis mutandis*, applies to the other three variants. The shifted variants of the QR and *QL* algorithms exhibit fast convergence in the last and first rows; those of the RQ and the LQ, in the first and last columns. For more details on the QL algorithm, see §1.3, Chapter 3.

The QR algorithm, the inverse power method, and the power method

The relation of the QR algorithm to the inverse power method was known to J. H. Wilkinson and W. Kahan (see [206, p. 173]), although the first explicit statement of the relation is in a textbook by Stewart [253, p. 352]. The relation to the power method was known from the first, as was the convergence result in Theorem 2.2, which was originally established using determinants. The proof given here is due to Wilkinson [301], [300, §§29–30, Ch. 8]. Theorem 2.1, which is so important to the implicitly double shifted algorithm, is due to Francis [76].

Local convergence

The local convergence results are adapted from [264]. As mentioned in the text, the iteration converges quadratically to a nondefective multiple eigenvalue. For the special case of the Rayleigh-quotient shift, this result is implied by work of Ostrowski on the Rayleigh-quotient method [193]; however, no general analysis seems to have appeared in the literature.

Watkins and Elsner [292] have given a very general convergence theory that holds for both the QR and LR algorithms.

Householder transformations and plane rotations

See §I:4.1.5 for notes and references on Householder transformations and plane rotations. The treatment here differs from that in many textbooks in its emphasis on complex transformations. For more on complex Householder transformations see [159].

Hessenberg form

The reduction of general matrix to Hessenberg form is due to Householder [123] and Wilkinson [297], who worked out the numerical details. The stability analysis is due to Wilkinson. His general approach to backward error analysis of orthogonal transformations is given in his *Algebraic Eigenvalue Problem* [300] and is the basis for most of the rounding-error results cited in this and the next chapter.

It is possible to use nonorthogonal transformations to reduce a symmetric matrix to tridiagonal form—a process that bears the same relation to Givens' and Householder's method as Gaussian elimination bears to orthogonal triangularization. The case for nonorthogonal reduction, however, is not as compelling as the case for Gaus-

sian elimination, and the method is not used in the major packages. For details and an error analysis see [300, §6.8–19].

The Hessenberg QR algorithm

As this subsection makes clear, the implementation of the QR algorithm is a large task, and our final result — Algorithm 2.8 — cannot be regarded as a polished implementation. For further details, there is little to do but go to the better packages and look at the code. The Handbook ALGOL codes [305] are well documented but show their age, as do their EISPACK translations into FORTRAN [81]. The LAPACK [3] codes are state of the art — at least for large matrices — but they are undocumented, sparsely commented, and difficult to read.

The observation that the QR step preserves Hessenberg form is due to Francis [76].

Deflation criteria

A personal anecdote may illustrate the difficulties in finding satisfactory criteria for declaring a subdiagonal element zero. In 1965 I attended a series of lectures on numerical analysis at the University of Michigan, where Jim Wilkinson presented the QR algorithm. His talk was punctuated by phrases like, “When a subdiagonal element becomes sufficiently small,” After one of the lectures I asked him, “How small is small?” He began, “Here is what I do,” and described the relative criterion (2.28). He ended by saying, “But if you want to argue about it, I would rather be on your side.” So it remains to this day.

The Wilkinson shift

The Wilkinson shift was proposed by Wilkinson [302] for symmetric tridiagonal matrices, where it insures global convergence of the QR algorithm. Its use in the general QR algorithm is traditional.

Aggressive deflation

The idea of deflating when there are two consecutive small subdiagonal elements is due to Francis [76]. The computational advantages of this procedure are obvious. In addition, many people felt that such small elements could retard the convergence of the implicitly shifted QR algorithm (to be discussed in the next section). However, Watkins [289] has shown that the shift is accurately transmitted through the small elements.

Preprocessing

The balancing algorithm (p. 107) is due to Osborne [187] and was used, along with the strategy for the initial deflation of eigenvalues, in the Handbook [208] and in subsequent packages. For another scaling strategy, which works well with sparse matrices, see [50].

Graded matrices

There is little in the literature about general graded matrices. For a general theory and references see [266]. The fact that the direction of the grading must be taken into account was noted for symmetric matrices by Martin, Reinsch, and Wilkinson [169]. Contrary to (2.36), they recommend that the matrix be graded upward, because their version of the reduction to Hessenberg form starts at the bottom of the matrix and works up. This matrix is then passed on to a variant of the QR algorithm—the QL algorithm—that also works from the bottom up.

3. THE IMPLICITLY SHIFTED QR ALGORITHM

In the previous section we described a variant of the QR algorithm that is suitable for computing the Schur decomposition of a general complex matrix. Unfortunately, complex arithmetic is quite expensive and to be avoided if at all possible. Since real matrices can have complex eigenvalues and eigenvectors, it is generally impossible to avoid complex arithmetic if the actual Schur form is to be computed. In this section we will show that there is a real equivalent of the Schur form and that the QR algorithm can be adapted to compute it in real arithmetic. The adaptation involves a technique called implicit shifting, which can also be used to solve generalized eigenproblems and compute the singular value decomposition.

We begin the section with the real Schur form and an inefficient algorithm for computing it. It turns out that these inefficiencies can be removed if the matrix in question is Hessenberg. Hence in §3.2 we establish an important result that enables us to take advantage of the Hessenberg form. Finally, in §3.3 we pull the material together into an algorithm for computing the real Schur form.

Since the concern is with solving the real eigenproblem:

In this section A will be a real matrix of order n .

3.1. REAL SCHUR FORM

In this subsection we derive the real Schur form and present a preliminary algorithm for its computation.

Real Schur form

A real Schur form is a Schur form with 2×2 blocks on its diagonal, each such block containing a conjugate pair of eigenvalues. The following theorem establishes the existence of such decompositions.

Theorem 3.1 (Real Schur form). Let A be real of order n . Then there is an orthogonal matrix U such that $T = U^T A U$ is block upper triangular of the form

$$U^T A U = \begin{pmatrix} T_{11} & T_{12} & T_{13} & \cdots & T_{1k} \\ 0 & T_{22} & T_{23} & \cdots & T_{2k} \\ 0 & 0 & T_{33} & \cdots & T_{3k} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & T_{kk} \end{pmatrix}. \quad (3.1)$$

The diagonal blocks of T are of order one or two. The blocks of order one contain the real eigenvalues of A . The blocks of order two contain the pairs of complex conjugate eigenvalues of A . The blocks can be made to appear in any order.

Proof. The proof is essentially the same as the proof of Theorem 1.12, Chapter 1, except that we use two real vectors to deflate a complex eigenvalue. Specifically, let (λ, x) be a complex eigenpair with

$$\lambda = \mu + i\nu \quad \text{and} \quad x = y + iz.$$

From the relations $2y = x + \bar{x}$ and $2zi = x - \bar{x}$, we find that $Ay = \mu y - \nu z$ and $Az = \nu y + \mu z$, or

$$A(y \ z) = (y \ z) \begin{pmatrix} \mu & \nu \\ -\nu & \mu \end{pmatrix} \equiv (y \ z)L. \quad (3.2)$$

By computing the characteristic polynomial of L , one can verify that the eigenvalues of L are λ and $\bar{\lambda}$.

Now let

$$(y \ z) = (X_1 \ X_2) \begin{pmatrix} R \\ 0 \end{pmatrix}$$

be a QR decomposition of $(y \ z)$. Then $(y \ z) = X_1 R$. Since y and z are independent (see Theorem 1.3, Chapter 1), R is nonsingular. Hence $X_1 = (y \ z)R^{-1}$, and

$$AX_1 = A(y \ z)R^{-1} = (y \ z)LR^{-1} = X_1 R L R^{-1}.$$

It follows as in the proof of Theorem 1.12, Chapter 1, that

$$\begin{pmatrix} X_1^T \\ X_2^T \end{pmatrix} A(X_1 \ X_2) = \begin{pmatrix} RLR^{-1} & X_1^T AX_2 \\ -0 & X_2^T AX_2 \end{pmatrix},$$

which completes the deflation of the complex conjugate pair λ and $\bar{\lambda}$. Because RLR^{-1} is similar to L , its eigenvalues are λ and $\bar{\lambda}$. ■

Three comments.

- A matrix of the form (3.1) in which the blocks are of order one or two is said to be *quasi-triangular*.
- Like the proof of Schur's theorem, the proof of this theorem is almost constructive. Given the necessary eigenvectors, one can reduce a matrix to real Schur form using Householder transformations. It is an instructive exercise to work out the details.
- The relation

$$AX_1 = X_1(RLR^{-1}) \quad (3.3)$$

implies that A maps the column space of X_1 into itself. Such a space is called an *eigen-space* or *invariant subspace*. Such subspaces will be the main concern of the second half of this volume.

A preliminary algorithm

The QR algorithm can be adapted to compute a real Schur form of a real matrix A in real arithmetic. Here we will start with an expensive algorithm that points in the right direction.

We begin by noting that the reduction of A to Hessenberg form (Algorithm 2.2) can be done in real arithmetic, resulting in a real matrix H . We can then apply the shifted QR algorithm (Algorithm 2.8) in real arithmetic until such time as a complex shift is generated. At that point, we need a new approach to avoid the introduction of complex elements in our matrix.

The starting point is the observation that if the Wilkinson shift κ is complex, then its complex conjugate $\bar{\kappa}$ is also a candidate for a shift. Suppose that we apply two steps of the QR algorithm, one with shift κ and the other with shift $\bar{\kappa}$ to yield a matrix \hat{H} . By Theorem 2.1, if

$$\check{Q}\check{R} = (H - \kappa I)(H - \bar{\kappa} I)$$

is the QR decomposition of $(H - \kappa I)(H - \bar{\kappa} I)$, then $\hat{H} = \check{Q}^H H \check{Q}$. But

$$(H - \kappa I)(H - \bar{\kappa} I) = H^2 - 2 \operatorname{Re}(\kappa)H + |\kappa|^2 I$$

is real. Hence so are \check{Q} and \hat{H} . Thus the QR algorithm with two complex conjugate shifts preserves reality. This strategy of working with complex conjugate Wilkinson shifts is called the *Francis double shift* strategy.

We can even avoid complex arithmetic in the intermediate quantities by forming the matrix $H^2 - 2 \operatorname{Re}(\kappa)H + |\kappa|^2 I$, computing its Q-factor \check{Q} , and then computing $\hat{H} = \check{Q}^T H \check{Q}$. Unfortunately, the very first step of this algorithm—the formation of $H^2 - 2 \operatorname{Re}(\kappa)H + |\kappa|^2 I$ —requires $O(n^3)$ operations, which makes the algorithm as a whole unsatisfactory. However, it turns out that we can use a remarkable property of Hessenberg matrices to sidestep the formation of $A^2 - 2 \operatorname{Re}(\kappa)A + |\kappa|^2 I$. We now turn to this property.

3.2. THE UNIQUENESS OF HESSENBERG REDUCTION

Let A be of order n and let $H = Q^H A Q$ be a unitary reduction of A to upper Hessenberg form. This reduction is clearly not unique. For example, the QR algorithm produces a sequence of Hessenberg matrices each of which is similar to the original matrix A . However, there is a limit to this nonuniqueness, as the following argument suggests.

In reducing A to Hessenberg form H by a unitary similarity, we must introduce $(n-1)(n-2)/2$ zeros into A . Now a unitary matrix has $n(n-1)/2$ degrees of freedom (see the discussion preceding Theorem 1.12, Chapter 1). Since we must use $(n-1)(n-2)/2$ of the degrees of freedom to introduce zeros in A , we have $n-1$ degrees of freedom left over in Q , just enough to specify the first column of Q . It is therefore reasonable to conjecture that, given A , the matrices Q and H are determined by the first column of Q . It turns out that this is essentially true, but with two provisos.

First, let $H = Q^H A Q$ be a unitary reduction of A to Hessenberg form. If $\hat{Q} = QD$, where $|D| = I$, then $\hat{H} = \hat{Q}^H A \hat{Q}$ is also a reduction to upper Hessenberg form. The two reductions are not essentially different, since \hat{Q} and Q differ only in the scaling of their columns by factors of modulus one. The choice of scaling will also affect the elements of H ; specifically, h_{ij} is replaced by $\bar{\delta}_i h_{ij} \delta_j$. In either case the rescaling makes no essential difference, and we will say that the reduction is *determined up to column scaling of Q* .

Second, we must assume that the subdiagonals of H are nonzero. Formally we make the following definition.

Definition 3.2. Let H be upper Hessenberg of order n . Then H is **UNREDUCED** if $h_{i+1,i} \neq 0$ ($i = 1, \dots, n-1$).

We are now in a position to give conditions under which a reduction to upper Hessenberg form is essentially unique.

Theorem 3.3 (Implicit Q theorem). Let A be of order n and let $H = Q^H A Q$ be a unitary reduction of A to upper Hessenberg form. If H is unreduced then up to column scaling of Q the matrices Q and H are uniquely determined by the first column of Q or the last column of Q .

Proof. Consider the relation

$$AQ = QH. \quad (3.4)$$

If we partition $Q = (q_1 \ \cdots \ q_n)$ by columns, then the first column of this relation is

$$Aq_1 = h_{11}q_1 + h_{21}q_2. \quad (3.5)$$

Multiplying (3.5) and remembering that $q_1^H q_1 = 1$ and $q_1^H q_2 = 0$, we have

$$h_{11} = q_1^H A q_1.$$

We then have

$$h_{21}q_2 = Aq_1 - h_{11}q_1.$$

Since $h_{21} \neq 0$, the vector $Aq_1 - h_{11}q_1$ is nonzero. Since $\|q_2\|_2 = 1$ we may take $h_{21} = \|Aq_1 - h_{11}q_1\|$, which determines q_2 uniquely up to a factor of modulus one.

Now suppose that q_1, \dots, q_k have been determined. Then the k th column of (3.4) gives the relation

$$Aq_k = h_{1k}q_1 + \dots + h_{kk}q_k + h_{k+1,k}q_{k+1}.$$

Since q_1, \dots, q_{k+1} are orthonormal, we have

$$h_{ik} = q_i^H A q_k, \quad i = 1, \dots, k.$$

Moreover,

$$h_{k+1,k}q_{k+1} = Aq_k - h_{1k}q_1 - \dots - h_{kk}q_k.$$

Since $h_{k+1,k}$ is nonzero, $Aq_k - h_{1k}q_1 - \dots - h_{kk}q_k$ is nonzero, and we may take

$$h_{k+1,k} = \|Aq_k - h_{1k}q_1 - \dots - h_{kk}q_k\|_2,$$

which determines q_{k+1} up to a factor of modulus one.

Finally, from the last column of (3.4) we get

$$h_{in} = q_i^H A q_n, \quad i = 1, \dots, n.$$

To show that Q is uniquely determined by its last column we start with the relation

$$Q^H A = H Q^H$$

and repeat the above argument on the rows of Q^H starting with the last. ■

The theorem is sometimes called the implicit Q theorem, because it allows the determination of Q implicitly from its first or last column. Its proof is actually an orthogonalization algorithm for computing Q and H . It is called the Arnoldi method and will play an important role in the second half of this volume. The algorithm itself has numerical drawbacks. For example, cancellation in the computation of $Aq_k - h_{1k}q_1 - \dots - h_{kk}q_k$ can cause the computed q_{k+1} to deviate from orthogonality with q_1, \dots, q_k .

3.3. THE IMPLICIT DOUBLE SHIFT

We now return to our preliminary algorithm and show how it may be modified to avoid the expensive computation of $H^2 - 2 \operatorname{Re}(\kappa)H + |\kappa|^2 I$. We will begin with a very general strategy and then fill in the details.

A general strategy

Let κ be a complex Francis shift of H . We have seen above that if we compute the Q-factor \check{Q} of the matrix $H^2 - 2 \operatorname{Re}(\kappa)H + |\kappa|^2 I$ then $\hat{H} = \check{Q}^T H \check{Q}$ is the result of applying two steps of the QR algorithm with shifts κ and $\bar{\kappa}$. The following is a general algorithm for simultaneously determining \check{Q} and \hat{H} .

1. Determine the first column c of $C = H^2 - 2 \operatorname{Re}(\kappa)H + |\kappa|^2 I$.
 2. Let Q_0 be a Householder transformation such that $Q_0^H c = \sigma \mathbf{e}_1$.
 3. Set $H_1 = Q_0^H A Q_0$.
 4. Use Householder transformations to reduce H_1 to upper Hessenberg form \hat{H} . Call the accumulated transformations Q_1 .
 5. Set $\check{Q} = Q_0 Q_1$.
- (3.6)

To establish that this algorithm works, we must show that \hat{H} and \check{Q} are the matrices that would result from two steps of the QR algorithm with shifts κ and $\bar{\kappa}$. We will proceed in stages.

1. The first column of Q_0 and the first column of \check{Q} are, up to scaling, the same. To see this, partition the QR factorization of C in the form

$$(c \ C_*) = (\check{q} \ \check{Q}_*) \begin{pmatrix} \rho & r^H \\ 0 & R_* \end{pmatrix}.$$

Then $c = \rho \check{q}$. Now if we partition $Q_0 = (q^{(0)} Q_*^{(0)})$, then from the relation $c = \sigma Q_0 \mathbf{e}_1$ we have $c = \sigma q^{(0)}$. Hence \check{q} and $q^{(0)}$ are proportional to c .

2. The first column of $Q_0 Q_1$ is proportional to \check{q} . This is because $Q_1 \mathbf{e}_1 = \mathbf{e}_1$ [see (2.21)].
3. By construction the matrix $(Q_0 Q_1)^T H (Q_0 Q_1)$ is Hessenberg. The first column of the reducing transformation is proportional to \check{q} . Hence by the implicit Q theorem (Theorem 3.3), if \hat{H} is unreduced, then $\check{Q} = Q_0 Q_1$ and $\hat{H} = (Q_0 Q_1)^T A (Q_0 Q_1)$.

The key computations in (3.6) are the computation of the first column of C and the reduction of $Q_1^T H Q_1$ to Hessenberg form. It turns out that because H is upper Hessenberg we can effect the first calculation in $O(1)$ operations and the second in $O(n^2)$ operations. We now turn to the details.

Getting started

The computation of the first column of $C = H^2 - 2 \operatorname{Re}(\kappa)H + |\kappa|^2 I$ requires that we first compute the scalars $2 \operatorname{Re}(\kappa)$ and $|\kappa|^2$. To do this we do not need to compute κ itself. For it is easily verified that

$$t \equiv 2 \operatorname{Re}(\kappa) = \operatorname{trace} \begin{pmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{nn} \end{pmatrix} \quad (3.7)$$

and

$$d \equiv |\kappa|^2 = \det \begin{pmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{nn} \end{pmatrix}. \quad (3.8)$$

Because H is upper Hessenberg, only the first three components of the first column of H^2 are nonzero. They are given by

$$\begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ 0 & h_{32} \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{21} \end{pmatrix} = \begin{pmatrix} h_{11}^2 + h_{12}h_{21} \\ h_{21}(h_{11} + h_{22}) \\ h_{21}h_{32} \end{pmatrix}.$$

Thus the first three components of the first column of c are given by

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} h_{11}^2 + h_{12}h_{21} - th_{11} + d \\ h_{21}(h_{11} + h_{22}) - th_{21} \\ h_{21}h_{32} \end{pmatrix}.$$

Inserting the definition (3.7) of t and (3.8) of d , we obtain the following formula for c :

$$c = h_{21} \begin{pmatrix} [(h_{nn} - h_{11})(h_{n-1,n-1} - h_{11}) - h_{n,n-1}h_{n-1,n}] / h_{21} + h_{12} \\ (h_{22} - h_{11}) - (h_{nn} - h_{11}) - (h_{n-1,n-1} - h_{11}) \\ h_{32} \end{pmatrix}.$$

Since two proportional vectors determine the same Householder transformation, we can ignore the multiplicative factor h_{21} in the definition of c .

The formulas for the components of $h_{21}^{-1}c$ involve products of the elements of H and are subject to overflow or underflow. The cure is to scale before calculating c . This, in effect, multiplies c by the scaling factor, which does not affect the final Householder transformation.

Algorithm 3.1 calculates the vector u generating the Householder transformation to start a doubly shifted QR step. In practice, when the problem deflates, the QR step will not be applied to the entire matrix, but between certain rows and columns of H . For this reason we do not invoke the algorithm with the matrix H , but with the specific elements of H used to generate c . Note that we may assume that h_{21} is nonzero, for otherwise the problem would deflate. Hence the scaling factor s in statement 2 is well defined. The particular form of the formulas has been chosen to avoid certain computational failures involving small off-diagonal elements. For more on this, see the notes and references.

An important consequence of our substitution of t for $2 \operatorname{Re}(\kappa)$ and d for $|\kappa|^2$ is that our algorithm works even if the Francis double shifts are real. Specifically, suppose that the matrix (3.7) and (3.8) has real eigenvalues λ and μ . Then

$$(A - \lambda I)(A - \mu I) = A^2 - (\lambda + \mu)A + \lambda\mu I = A^2 - tA + dI.$$

Thus the formulas in Algorithm 3.1 generate a vector u that starts the iteration with shifts of λ and μ .

This program generates a Householder transformation that reduces the first column of $C = H^2 - 2 \operatorname{Re}(\kappa)H + |\kappa|^2 I$ to a multiple of e_1 .

1. *startqr2step(h11, h12, h21, h22, h32, hn1n1, hn1n, hnn1, hnn, u)*
2. $s = 1 / \max\{|h11|, |h12|, |h21|, |h22|, |h32|\}$
 $|hn1n1|, |hn1n|, |hnn1|, |hnn| \}$
3. $h11 = s * h11; h12 = s * h12; h21 = s * h21;$
 $h22 = s * h22; h32 = s * h32;$
 $hn1n1 = s * hn1n1; hn1n = s * hn1n;$
 $hnn1 = s * hnn1; hnn = s * hnn$
4. $p = hnn - h11$
5. $q = hn1n1 - h11$
6. $r = h22 - h11$
7. $c = \begin{pmatrix} (p * q - hnn1 * hn1n) / h21 + h12 \\ r - p - q \\ h32 \end{pmatrix}$
8. *housegen(c, u, v)*
9. **end startqrstep**

Algorithm 3.1: Starting an implicit QR double shift

Reduction back to Hessenberg form

Let R_0 denote the Householder transformation corresponding to the vector u returned by Algorithm 3.1. Then premultiplication by R_0 acts only on the first three rows of H , so that $R_0 H$ has the form

$$\xrightarrow{R_0 H} \begin{pmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & X & X \end{pmatrix}.$$

Similarly, postmultiplication by R_0 operates on only the first three columns of H so that after the complete transformation has been applied, we get the matrix

$$\xrightarrow{HR_0} \begin{pmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ \hat{X} & X & X & X & X & X \\ \hat{X} & X & X & X & X & X \\ 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & X & X \end{pmatrix}.$$

We must now reduce this matrix to Hessenberg form. In the usual reduction to Hessenberg form we would annihilate all the elements below the first subdiagonal. But here only two of the elements are nonzero, and we can use a Householder transformation R_1 that acts only on rows and columns two through four. The result of applying this transformation is

$$R_1 H R_1 \Rightarrow \begin{pmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & \hat{X} & X & X & X & X \\ 0 & \hat{X} & X & X & X & X \\ 0 & 0 & 0 & 0 & X & X \end{pmatrix}.$$

We now continue with a transformation R_2 that annihilates the hatted elements in this matrix to give

$$R_2 H R_2 \Rightarrow \begin{pmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & \hat{X} & X & X & X \\ 0 & 0 & \hat{X} & X & X & X \end{pmatrix}.$$

Yet another transformation R_3 annihilates the hatted elements in the third column:

$$R_3 H R_3 \Rightarrow \begin{pmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & \hat{X} & X & X \end{pmatrix}.$$

The last transformation is a special case, since there is only one element to annihilate. This can be done with a 2×2 Householder transformation or a plane rotation R_4 :

$$R_3 H R_3^T \Rightarrow \begin{pmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & X & X \end{pmatrix}.$$

Note that the transpose in the above diagram is necessary if R_3 is a rotation; it makes no difference when R_3 is a Householder transformation, which is symmetric.

Algorithm 3.2 implements a double implicit QR shift between rows and columns

Given an upper Hessenberg matrix H and the vector u from Algorithm 3.1 (*startqrstep*), this program applies u to H and then reduces H to Hessenberg form. The transformations are applied between rows $i1$ and $i2$ and are accumulated in a matrix Q .

```

1.  qr2step( $H, u, i1, i2, Q$ )
2.  for  $i = i1$  to  $i2 - 2$ 
3.     $j = \max\{i-1, i1\}$ 
4.     $v^T = u^T * H[i:i+2, j:n]$ 
5.     $H[i:i+2, j:n] = H[i:i+2, j:n] - u * v^T$ 
6.     $iu = \min\{i+3, i2\}$ 
7.     $v = H[1:iu, i:i+2]*u$ 
8.     $H[1:iu, i:i+2] = H[1:iu, i:i+2] - v * u^T$ 
9.     $v = Q[1:n, i:i+2]*u$ 
10.    $Q[1:n, i:i+2] = Q[1:n, i:i+2] - v * u^T$ 
11.   if ( $i \neq i2 - 2$ ) housegen( $H[i+1:i+3, i], u, nu$ ) fi
12.   if ( $i \neq i1$ )  $H[i+1, j] = H[i+2, j] = 0$ ; fi
13. end for  $i$ 
14. rotgen( $H[i2-1, i2-2], H[i2, i2-2], c, s$ )
15. rotapp( $c, s, H[i2-1, i2-1:n], H[i2, i2-1:n]$ )
16. rotapp( $c, s, H[1:i2, i2-1], H[1:i2, i2]$ )
17. rotapp( $c, s, Q[1:n, i2-1], Q[1:n, i2]$ )
18. end qr2step

```

Algorithm 3.2: Doubly shifted QR step



$i1$ and $i2$ of H . The coding is a bit intricate, since there are special cases at both ends of the loop on i . It helps to think of i as pointing to the row containing the subdiagonal at which the current Householder transformation starts.

It is easy to do an operation count on this algorithm, with the result that:

Algorithm 3.2 requires $6n(i2-i1)$ flam.

Deflation

The behavior of the doubly shifted QR algorithm once asymptotic convergence has set in depends on the eigenvalues of the matrix

$$\begin{pmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{nn} \end{pmatrix},$$

which determines the shift. If the eigenvalues are complex and nondefective, then $h_{n-1,n-2}$ converges quadratically to zero. If the eigenvalues are real and nondefective, both the $h_{n-1,n-2}$ and $h_{n-1,n}$ converge quadratically to zero. In either case, we must allow for the deflation of a 2×2 block at the bottom of the matrix.

As usual, the subdiagonal elements other than $h_{n-1,n-2}$ and $h_{n-1,n}$ may show a slow convergence to zero so that we must allow for deflation in the middle of the matrix. Algorithm 3.3 implements a backsearch for deflation. It differs from Algorithm 2.6 in recognizing deflation of both 1×1 and 2×2 blocks. A 2×2 block may require additional processing. At the very least, if it contains real eigenvalues, it should be rotated into upper triangular form so that the eigenvalues are exhibited on the diagonal.

As with the singly shifted algorithm, we can start a QR step at a point where two consecutive subdiagonals are small, even when they are not small enough to cause a deflation. For more on this topic, see the notes and references.

Computing the real Schur form

Algorithm 3.4 reduces a real Hessenberg matrix to real Schur form. There are several comments to be made about the algorithm.

- We have already observed that the routine *qr2step* requires $6n^2(i2-i1)$ flam. If we assume that $i1 = 1$ throughout the reduction and that it requires k iterations to compute an eigenvalue, then the total operation count is

$$2kn^3 \text{ flam} = 2kn^3 \text{ flmlt} + 2kn^3 \text{ fladd.}$$

On the other hand, the explicitly shifted QR algorithm (Algorithm 2.8) requires $k'n^3$ complex flrots, where k' is the number of iterations to find an eigenvalue. In terms of real arithmetic this is

$$12k'n^3 \text{ flmlt} + 8k'n^3 \text{ fladd.}$$

This algorithm takes a Hessenberg matrix H of order n and an index ℓ ($1 < \ell \leq n$) and determines indices $i1, i2 \leq \ell$ such that one of the following two conditions hold.

1. $1 \leq i1 < i2 \leq \ell$, in which case the matrix deflates at rows $i1$ and $i2$.
2. $1 = i1 = i2$, in which case no two consecutive subdiagonal elements of H in rows 1 through ℓ are nonzero.

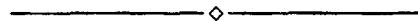
In processing 2×2 blocks all transformations are accumulated in Q .

```

1.  backsearch2( $H, Q, \ell, i1, i2$ )
2.   $i1 = i2 = \ell$ 
3.  while ( $i1 > 1$ )
4.    if ( $H[i1-1, i1]$  is negligible or  $i2 = 2$ )
5.      if ( $i2 \neq 2$ )  $H[i1-1, i1] = 0$ ; fi
6.      if ( $i1 = i2-1$  or  $i2=2$ )
7.        Process  $2 \times 2$  block
8.        if ( $i2 \neq 2$ )
9.           $i1 = i2 = i1-1$ 
10.         else
11.            $i1 = i2 = 1$ 
12.         end if
13.         else if ( $i1 = i2$ )
14.            $i1 = i2 = i1-1$ 
15.         else
16.           leave while
17.         end if
18.       else
19.          $i1 = i1-1$ 
20.       end if
21.     end while
22.   end backsearch2

```

Algorithm 3.3: Finding deflation rows in a real upper Hessenberg matrix



Given a real upper Hessenberg matrix H , hqr overwrites it with a orthogonally similar quasi-triangular matrix. The similarity transformation is accumulated in Q . The routine gives an error return if more than $maxiter$ iterations are required to deflate the matrix at any eigenvalue.

```

1.   hqr( $H$ ,  $Q$ ,  $maxiter$ )
2.    $i1 = 1; i2 = n$ 
3.    $iter = 0$ 
4.   while (true)
5.     if ( $iter > maxiter$ ) error return; fi
6.      $oldi2 = i2$ 
7.     backsearch2( $H$ ,  $i2$ ,  $i1$ ,  $i2$ )
8.     if ( $i2 = 1$ ) leave hqr fi
9.     if ( $i2 = oldi2$ )  $iter = iter + 1$  else  $iter = 0$  fi
10.    startqr2step( $H[i1, i1]$ ,  $H[i1, i1+1]$ ,
                   $H[i1+1, i1]$ ,  $H[i1+1, i1+1]$ ,  $H[i1+2, i1+1]$ ,
                   $H[i2-1, i2-1]$ ,  $H[i2-1, i2]$ ,
                   $H[i2, i2-1]$ ,  $H[i2, i2]$ ,  $u$ )
11.    qr2step( $H$ ,  $Q$ ,  $u$ ,  $i1$ ,  $i2$ )
12.    end while
13.  end hqr
```

Algorithm 3.4: Real Schur form of a real upper Hessenberg matrix

Even if $k = k'$, the superiority of the double shift algorithm is clear. In fact, one would expect that $k < k'$, since each iteration of the double shift algorithm is equivalent to two steps of the single shift algorithm.

- Like the single shift algorithm, the double shift algorithm is numerically stable in the usual sense.
- It should not be thought that because Algorithms 2.8 and 3.4 are both backward stable that they produce essentially the same matrices. For example, since the real Schur form is real, the doubly shifted algorithm produces a matrix whose complex eigenvalues occur in conjugate pairs. The singly shifted algorithm, on the other hand, works in the complex field, and the imaginary parts of a pair of complex eigenvalues can drift away from conjugacy — very far away if the eigenvalue is ill conditioned. This is yet another argument for computing the real Schur form.

Eigenvectors

The eigenvectors of the original matrix A can be found by computing the eigenvectors of the Schur form T produced by Algorithm 3.4 and transforming them back using the orthogonal transformation Q . Thus the problem of finding the eigenvectors of the original matrix A is reduced to computing the eigenvectors of a quasi-triangular matrix T . The method is a generalization of the back-substitution process of Algorithm 2.9. Rather than present a complete program, which would necessarily be lengthy, we will sketch the important features of the algorithm. It will be sufficient to illustrate the procedure on a quasi-triangular matrix T consisting of three blocks. We will assume that the eigenvalues of a diagonal block are distinct from those of the other diagonal blocks.

We begin with the case of a real eigenvalue. Suppose T has the form

$$T = \begin{pmatrix} T_{11} & t_{12} & t_{13} \\ 0 & \tau_{22} & \tau_{23} \\ 0 & 0 & \tau_{33} \end{pmatrix}.$$

If we seek the eigenvector corresponding to $\lambda = \tau_{33}$ in the form $(x_1^T \ \xi_2 \ 1)^T$, then we have the equation

$$\begin{pmatrix} T_{11} & t_{12} & t_{13} \\ 0 & \tau_{22} & \tau_{23} \\ 0 & 0 & \tau_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ \xi_2 \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 \\ \xi_2 \\ 1 \end{pmatrix} \lambda. \quad (3.9)$$

(The reason for putting λ to the right of the eigenvector will become clear a little later.)

From the second row of (3.9) we have

$$\tau_{22}\xi_2 - \xi_2\lambda = -\tau_{23},$$

whence $\xi_2 = -\tau_{23}/(\tau_{22} - \lambda)$. This is just the usual back-substitution algorithm.

From the first row of (3.9) we have

$$T_{11}x_1 - x_1\lambda = -t_{13} - t_{13}\xi_2.$$

Hence we may obtain x_1 as the solution of the linear system $(T_{11} - \lambda I)x_1 = -t_{13} - \xi_2 t_1$. Here we have deviated from the usual back-substitution algorithm: the two components of the eigenvector in x_1 are obtained by solving a 2×2 linear system. The system is nonsingular because by hypothesis λ is not an eigenvalue of T_{11} .

For a general quasi-triangular matrix T the back-substitution proceeds row by row as above. If the row in question has a scalar on the diagonal, then the process reduces to the usual back-substitution. On the other hand, if the diagonal block is 2×2 , then two components of the eigenvector are obtained by solving a system of order 2.

The case of a complex conjugate pair of eigenvalues is sufficiently well illustrated by the matrix

$$T = \begin{pmatrix} T_{11} & t_{12} & T_{13} \\ 0 & \tau_{22} & t_{23}^T \\ 0 & 0 & T_{33} \end{pmatrix}.$$

We will look for an “eigenvector” (later we will call it an *eigenbasis*) in the form

$$\begin{pmatrix} T_{11} & t_{12} & T_{13} \\ 0 & \tau_{22} & t_{23}^T \\ 0 & 0 & T_{33} \end{pmatrix} \begin{pmatrix} X_1 \\ x_2^T \\ X_3 \end{pmatrix} = \begin{pmatrix} X_1 \\ x_2^T \\ X_3 \end{pmatrix} L \quad (3.10)$$

for some matrix L of order 2. We already have two examples of such block eigenvectors in (3.2) and (3.3).

We begin by letting X_3 be nonsingular but otherwise arbitrary. From the third row of (3.10), we have

$$T_{33}X_3 = X_3L,$$

from which it follows that

$$L = X_3^{-1}T_{33}X_3.$$

Since L is similar to T_{33} it contains the complex conjugate pair of eigenvalues of T_{33} .

From the second row of (3.10) we have

$$\tau_{22}x_2^T - x_2^T L = -t_{23}^T X_3.$$

Hence we may compute x_2^T as the solution of the 2×2 system $x_2^T(\tau_{22} - L) = -t_{23}^T X_3$. This system is nonsingular because τ_{22} is not an eigenvalue of L .

From the third row of (3.10) we have

$$T_{11}X_1 - X_1L = -T_{13} - t_{12}x_2^T.$$

This is a Sylvester equation, which we can solve for X_1 because T_{11} and L have no eigenvalues in common (see Theorem 1.16, Chapter 1). For larger matrices the process continues, each row of the matrix yielding either a 2×2 system or a Sylvester equation for the corresponding part of X .

We now turn to the choice of X_3 . A natural choice is $X_3 = I$ — the generalization of the choice $\xi_3 = 1$ for an ordinary eigenvector. The principal advantage of this choice is that it does not require any computation to form L , which is simply X_{33} .

However, if actual eigenvectors are desired, it is better to take $X_3 = (y_3 \ z_3)$, where $x_3 = y_3 + iz_3$ is the right eigenvector of T_{33} . Arguing as in the proof of Theorem 3.1 [see (3.2)], we have

$$L = \begin{pmatrix} \mu & \nu \\ -\nu & \mu \end{pmatrix},$$

where $\mu \pm i\nu$ are the eigenvalues of T_{33} . If we then compute X as described above and partition $X = (y \ z)$, then $TX = XL$ or

$$T(y \ z) = (y \ z) \begin{pmatrix} \mu & \nu \\ -\nu & \mu \end{pmatrix}.$$

This relation implies that $y \pm iz$ are the right eigenvectors of T corresponding to the eigenvalues $\mu \pm i\nu$. Thus our choice generates the real and imaginary parts of the desired eigenvector.

This sketch leaves open many implementation details — in particular, how to solve the linear systems and how to scale to prevent overflow. Moreover, we have presented only row-oriented versions of the algorithms. It is easy to cast them into column-oriented form in the spirit of Algorithm 2.9.

The major packages adopt a different approach that is equivalent to our second choice of T_{33} . Specifically, for a complex eigenvalue λ they write out the relation $Ax = \lambda x$ and backsolve as in the real case but compute the real and imaginary parts explicitly in real arithmetic.

3.4. NOTES AND REFERENCES

Implicit shifting

The ideas in this section, from the real Schur form to the implicit double shift, are due to Francis [76]. Although we have focused on the double shift version, there is a single shift variant, which we will use to compute eigenvalues of a symmetric tridiagonal matrix. There is also a multishift variant (see below), which is implemented in the LAPACK routine xHSEQR.

The idea of an implicit shift has consequences extending beyond the computation of a real Schur form. In the next section we will show how to use implicit shifting to solve the generalized eigenvalue problem and in the next chapter will apply it to the singular value decomposition.

Processing 2×2 blocks

When a 2×2 block has converged, it should be split if it has real eigenvalues. The LAPACK routine `xLANV2` does this. It also transforms a block with complex eigenvalues into one in which the diagonal elements are equal.

The multishifted QR algorithm

There is no need to confine ourselves to two shifts in the implicitly shifted QR algorithm. For example, when we treat the symmetric eigenvalue problem, we will use single shifts. We can also use more than two shifts. Specifically, suppose that we have chosen m shifts $\kappa_1, \dots, \kappa_m$. We compute the first column c of $(A - \kappa_m I) \cdots (A - \kappa_1 I)$, which will have $m + 1$ leading nonzero components. We then determine an $(m+1) \times (m+1)$ Householder transformation Q_0 that reduces c to e_1 and then apply it to A . The matrix $Q_0 A Q_0$ has an $(m+1) \times (m+1)$ bulge below its diagonal, which is chased down out of the matrix in a generalization of Algorithm 3.2. The shifts can be computed as the eigenvalues of the trailing $m \times m$ submatrix.

This algorithm, proposed by Bai and Demmel [9] and implemented in the LAPACK routine `xHSEQR`, can take advantage of level-3 BLAS — the larger m the greater the advantage. Unfortunately, Watkins [290] has shown that if the number of shifts is great they are not transmitted accurately through the bulge chasing process.

An alternative is to note that if, say, the bulge from a 2×2 shift is chased two steps down the diagonal then another double shift can be started after it. If this process is continued, the result is a sequence of small, adjacent bulges being chased down the matrix. This algorithm and a new deflation procedure is treated by Braman, Byers, and Mathias [29]. The paper also contains an extensive bibliography on the subject.

4. THE GENERALIZED EIGENVALUE PROBLEM

The *generalized eigenvalue problem* is that of finding nontrivial solutions of the equation

$$Ax = \lambda Bx,$$

where A and B are of order n . The problem reduces to the ordinary eigenvalue problem when $B = I$ — which is why it is called a *generalized* eigenvalue problem. Although the generalization results from the trivial replacement of an identity matrix by an arbitrary matrix B , the problem has many features not shared with the ordinary eigenvalue problem. For example, a generalized eigenvalue problem can have infinite eigenvalues.

In spite of the differences between the two problems, the generalized eigenvalue problem has the equivalents of a Hessenberg form and a Schur form. Moreover, the QR algorithm can be adapted to compute the latter. The resulting algorithm is called the QZ algorithm, and the purpose of this section is to describe it.

We begin in the next subsection with the theoretical background, including Schur forms and perturbation theory. We then turn to the details of the QZ algorithm. Like the QR algorithm, the QZ algorithm begins by reducing A and B to a simplified form, called Hessenberg-triangular form, and then iterates to make both pairs triangular. We will treat the reduction to Hessenberg-triangular form in §4.2 and the QZ iteration in §4.3.

4.1. THE THEORETICAL BACKGROUND

In this subsection we will develop the theory underlying the generalized eigenvalue problem. For reasons of space, we only sketch the outlines. The reader is referred to the notes and references for more extensive treatments.

Definitions

We begin with the definition of generalized eigenvalues and eigenvectors.

Definition 4.1. Let A and B be of order n . The pair (λ, x) is an EIGENPAIR or RIGHT EIGENPAIR of the PENCIL (A, B) (also written $A - \lambda B$) if

1. $x \neq 0$,
2. $Ax = \lambda Bx$.

The scalar λ is called an EIGENVALUE of the pencil and the vector x is its EIGENVECTOR.

The pair (λ, y) is a LEFT EIGENPAIR of the pencil (A, B) if $y \neq 0$ and $y^H A = \lambda y^H B$.

As with the ordinary eigenvalue problem, eigenpairs of the generalized problem are assumed to be right eigenpairs unless otherwise qualified.

It is important to keep in mind that the geometry of generalized eigenvectors differs from that of ordinary eigenvectors. If (λ, x) is an eigenpair of A , then A maps x into a multiple of itself — i.e., the direction of x remains invariant under multiplication by A , provided we agree that the direction of the zero vector matches that of any nonzero vector. On the other hand, if (λ, x) is an eigenpair of the pencil (A, B) , the direction of x is not necessarily preserved under multiplication by A or B . Instead, the direction of both Ax and Bx are the same. This elementary fact will be useful in understanding the material to follow.

Regular pencils and infinite eigenvalues

An important difference between the generalized eigenvalue problem and the ordinary eigenvalue problem is that the matrix B in the former can be singular whereas the matrix I in the latter cannot. This difference has important consequences.

In the first place if B is singular, it is possible for any number λ to be an eigenvalue of the pencil $A - \lambda B$. For example, if

$$A = B = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

then $Ae_2 = Be_2 = 0$. Hence for any λ , $Ae_2 = \lambda Be_2$. More generally, if A and B have a common null vector x , then (λ, x) is an eigenpair of (A, B) for any λ . An even more general example is the pencil

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \lambda \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

in which A and B have no null vectors, right or left, in common.

In all these examples the determinant of $A - \lambda B$ is identically zero. This suggests the following definition.

Definition 4.2. A matrix pencil (A, B) is **REGULAR** if $\det(A - \lambda B)$ is not identically zero. We also say that the pair (A, B) is **regular**.

A regular matrix pencil can have only a finite number of eigenvalues. To see this, note that $Ax = \lambda Bx$ ($x \neq 0$) if and only if $\det(A - \lambda B) = 0$. Now $p(\lambda) = \det(A - \lambda B)$ is a polynomial of degree $m \leq n$. If the pencil is regular, then $p(\lambda)$ is not identically zero and hence has m zeros, which are eigenvalues of (A, B) .

It is possible for $p(\lambda)$ to be constant, in which case it appears that the pencil has no eigenvalues. However, this can only occur if B is singular. Now if $\lambda \neq 0$ is an eigenvalue of a pencil (A, B) , then $\mu = \lambda^{-1}$ is an eigenvalue of (B, A) and vice versa. If B is singular, then $Bx = 0$ for some $x \neq 0$, and zero is an eigenvalue of (B, A) . It is therefore natural to regard $\infty = 1/0$ as an eigenvalue of (A, B) . Thus if $p(\lambda)$ is constant, the pencil must have an infinite eigenvalue corresponding to the vector x .

The possible existence of infinite eigenvalues distinguishes the generalized eigenvalue problem from the ordinary one. It should be stressed that infinite eigenvalues are just as legitimate as their finite brethren. However, infinite eigenvalues do present mathematical and notational problems. For example, what is the multiplicity of an infinite eigenvalue? To answer this question, among others, we will appeal to a generalization of the Schur form (Theorem 4.5). But first we must describe the kind of transformations we will use to arrive at this generalized Schur form.

Equivalence transformations

We have already observed that the natural transformations for simplifying an ordinary eigenvalue problem are similarity transformations. They preserve the eigenvalues of a matrix and change the eigenvectors in a systematic manner. The corresponding transformations for the generalized eigenvalue problem are called equivalence transformations.

Definition 4.3. Let (A, B) be a matrix pencil and let U and V be nonsingular. Then the pencil $(U^H AV, U^H BV)$ is said to be **EQUIVALENT** to (A, B) . Alternatively, we say that $(U^H AV, U^H BV)$ is obtained from (A, B) by an **EQUIVALENCE TRANSFORMATION**.

The effect of a equivalence transformation on the eigenvalues and eigenvectors of a matrix pencil is described in the following theorem, whose proof is left as an exercise.

Theorem 4.4. Let (λ, x) and (λ, y) be left and right eigenpairs of the regular pencil (A, B) . If U and V are nonsingular, then $(\lambda, V^{-1}x)$ and $(\lambda, U^{-1}y)$ are eigenpairs of $(U^H AV, U^H BV)$.

Equivalence transformations also leave the polynomial $p(\lambda) = \det(A - \lambda B)$ essentially unaltered. In fact,

$$\det(U^H AV - \lambda U^H BV) = \det(U^H) \det(V) \det(A - \lambda B). \quad (4.1)$$

Since U and V are nonsingular, the equivalence transformation multiplies the $p(\lambda)$ by a nonzero constant. In particular, the roots of $p(\lambda)$ and their multiplicity are preserved by equivalence transformations.

Generalized Schur form

Just as we can reduce a matrix to simpler forms by similarity transformations, so can we simplify a pencil by equivalence transformations. For example, there is an analogue of the Jordan form for regular pencils called the Weierstrass form. Unfortunately, this form, like the Jordan form, is not numerically well determined, and the transformations that produce it can be ill conditioned (see §1.6, Chapter 1). Fortunately, if we restrict ourselves to unitary equivalences, which are well conditioned, we can reduce the matrices of a pencil to triangular form. Specifically, we have the following theorem.

Theorem 4.5 (Generalized Schur form). Let (A, B) be a regular pencil. Then there are unitary matrices U and V such that $S = U^H AV$ and $T = U^H BV$ are upper triangular.

Proof. Let v be an eigenvector of (A, B) normalized so that $\|v\|_2 = 1$, and let $(v \ V_\perp)$ be unitary. Since the (A, B) is regular, at least one of the vectors Av and Bv must be nonzero—say Av . Moreover, if $Bv \neq 0$ then it must be proportional to Av . Let $u = Av/\|Av\|_2$ and let $(u \ U_\perp)$ be unitary. Then

$$(u \ U_\perp)^H A(v \ V_\perp) = \begin{pmatrix} u^H Av & u^H AV_\perp \\ U_\perp^H Av & U_\perp^H AV_\perp \end{pmatrix} \equiv \begin{pmatrix} \sigma_{11} & s_{12}^H \\ 0 & \hat{A} \end{pmatrix}. \quad (4.2)$$

The $(2, 1)$ -block of the matrix on the right is zero because by construction Av is orthogonal to the column space of U_\perp .

Similarly,

$$(u \ U_\perp)^H B(v \ V_\perp) = \begin{pmatrix} u^H Bv & u^H BV_\perp \\ U_\perp^H Bv & U_\perp^H BV_\perp \end{pmatrix} \equiv \begin{pmatrix} \tau_{11} & t_{12}^H \\ 0 & \hat{B} \end{pmatrix}. \quad (4.3)$$

The proof is now completed by an inductive reduction of (\hat{A}, \hat{B}) to triangular form. ■

We now turn to four consequences of this theorem.

Multiplicity of eigenvalues

Let (A, B) be a regular pencil and let (\hat{A}, \hat{B}) be a generalized Schur form of (A, B) . Then

$$p(\lambda) \equiv \det(A - \lambda B) = \omega \prod_{\hat{b}_{ii} \neq 0} (\hat{a}_{ii} - \lambda \hat{b}_{ii}) \prod_{\hat{b}_{ii}=0} \hat{a}_{ii},$$

where $|\omega| = 1$ (ω is the product of the determinants of the unitary transformations that produce the generalized Schur form). Now the degree m of p is invariant under equivalence transformations [see (4.1)]. Hence exactly m of the b_{ii} are nonzero and $n-m$ are zero. The former correspond to the finite eigenvalues of (A, B) , while the latter correspond to the infinite eigenvalues. Thus any generalized Schur form will exhibit the same number of finite and infinite eigenvalues.

This argument justifies the following definitions.

Definition 4.6. Let (A, B) be a regular pencil of order n . Then

$$p_{(A,B)}(\lambda) = \det(A - \lambda B)$$

is the CHARACTERISTIC POLYNOMIAL of (A, B) . We say that the ALGEBRAIC MULTIPLICITY of a finite eigenvalue of (A, B) is its multiplicity as a zero of the characteristic equation. If m is the degree of the characteristic polynomial and $m < n$, then we say that (A, B) has an INFINITE EIGENVALUE OF ALGEBRAIC MULTIPLICITY $n-m$. An eigenvalue of algebraic multiplicity one is called a SIMPLE EIGENVALUE.

Another consequence of the above argument is that we can choose the eigenvalues of (A, B) to appear in any order in a generalized Schur form. Specifically, we can choose any eigenvalue, finite or infinite, to be in the first position. After the first step of the reduction, we have from (4.2) and (4.3) that

$$p_{(A,B)}(\lambda) = \omega(\sigma_{11} - \lambda \tau_{11}) \det(\hat{A} - \lambda \hat{B}),$$

where $|\omega| = 1$. If the second eigenvalue in the sequence is finite, then it must satisfy $\det(\hat{A} - \lambda \hat{B}) = 0$; i.e., it must be an eigenvalue of (\hat{A}, \hat{B}) . On the other hand, if the second eigenvalue is infinite, \hat{B} must be singular; for otherwise, the degree of $p_{(A,B)}$ would be too large. In either case we can find an eigenvector corresponding to the second eigenvalue. Thus we have established the following corollary.

Corollary 4.7. In Theorem 4.5, the eigenvalues of (A, B) can be made to appear in any order on the diagonals of (S, T) .

Projective representation of eigenvalues

The Schur form has enabled us to solve the mathematical problem of defining the multiplicity of generalized eigenvalues — finite or infinite. It also suggests how to put finite and infinite eigenvalues on the same notational footing.

Let (A, B) be a regular pencil reduced to Schur form. Then the eigenvalues of (A, B) are determined by the pairs $(\alpha_{ii}, \beta_{ii})$. If β_{ii} is nonzero, $\lambda = \alpha_{ii}/\beta_{ii}$ is a finite eigenvalue of A . Otherwise the eigenvalue is infinite. We can erase the distinction between the two kinds eigenvalues by representing them by the pairs $(\alpha_{ii}, \beta_{ii})$. This is equivalent to writing the generalized eigenvalue problem in the form

$$\beta_{ii}Ax = \alpha_{ii}Bx.$$

Unfortunately, this representation is not unique, since any nonzero multiple of the pair $(\alpha_{ii}, \beta_{ii})$ represents the same eigenvalue. We solve this problem by introducing the set

$$\langle \alpha_{ii}, \beta_{ii} \rangle = \{ \tau(\alpha_{ii}, \beta_{ii}) : \tau \in \mathbb{C} \}. \quad (4.4)$$

We will call this set the *projective representation of the eigenvalue*. In this nomenclature $\langle 0, 1 \rangle$ is a zero eigenvalue and $\langle 1, 0 \rangle$ is an infinite eigenvalue. An ordinary eigenvalue λ can be written in the form $\langle \lambda, 1 \rangle$.

Generalized shifting

Let (A, B) be a pencil with B nonsingular. Then the equation $Ax = \lambda Bx$ can be written in the form $B^{-1}Ax = \lambda x$. Thus when B is nonsingular, we can reduce the generalized eigenvalue problem to an ordinary eigenvalue problem — a computationally appealing procedure.

The reduction is not possible when B is singular, and it has computational difficulties when B is ill conditioned. To remedy this difficulty we can work with the *shifted pencil* $Ax = \mu(B + wA)x$, whose eigenpairs are $(\lambda/(1 + w\lambda), x)$. If we can determine w so that $B + wA$ is well conditioned, we can reduce the shifted problem to the ordinary eigenvalue problem $(B + wA)^{-1}Ax = \mu x$.

We are not restricted to shifting B alone. We can shift A and B simultaneously, as the following theorem shows.

Theorem 4.8. *Let (A, B) be a regular pencil and let*

$$W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$$

be nonsingular. Let

$$(C, D) = (w_{11}A + w_{21}B, w_{12}A + w_{22}B).$$

Then $(\langle \alpha, \beta \rangle, x)$ is an eigenpair of (A, B) if and only if

$$(\langle \gamma, \delta \rangle, x) \equiv (\langle w_{11}\alpha + w_{21}\beta, w_{12}\alpha + w_{22}\beta \rangle, x)$$

is an eigenpair of (C, D) . Moreover, these eigenpairs have the same multiplicity.

Proof. Assume without loss of generality that (A, B) is in generalized Schur form with the pair (α, β) in the initial position, so that the vector \mathbf{e}_1 is the corresponding eigenvector. Then (C, D) is also in Schur form with $(\gamma, \delta) = (\alpha, \beta)W$ in the initial position. Since W is nonsingular $(\gamma, \delta) \neq (0, 0)$. Consequently, $(\langle \gamma, \delta \rangle, \mathbf{e}_1)$ is an eigenpair of (C, D) .

The converse is shown by using the nonsingularity of W to pass from (C, D) back to (A, B) .

The statement about multiplicities follows from the fact that if an eigenvalue appears m times on the diagonal of (A, B) then by the nonsingularity of W the transformed eigenvalue appears exactly m times on the diagonal of (C, D) . ■

Left and right eigenvectors of a simple eigenvalue

We have seen that if λ is a simple eigenvalue of a matrix A then its left and right eigenvectors cannot be orthogonal [see (3.12), Chapter 1]. This allows us to calculate the eigenvalue in the form of a Rayleigh quotient $y^H A x / y^H x$. The left and right eigenvectors of a simple eigenvalue of a pencil can be orthogonal. For example, the right and left eigenvectors of the pencil

$$\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} - \lambda \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

corresponding to the eigenvalue 1 are \mathbf{e}_1 and \mathbf{e}_2 . Nonetheless, there is a generalized form of the Rayleigh quotient, as the following theorem shows.

Theorem 4.9. Let $\langle \alpha, \beta \rangle$ be a simple eigenvalue of the regular pencil (A, B) . If x and y are right and left eigenvectors corresponding to $\langle \alpha, \beta \rangle$, then

$$\langle \alpha, \beta \rangle = \langle y^H A x, y^H B x \rangle. \quad (4.5)$$

Consequently,

$$y^H A x = 0 \implies Ax = 0 \text{ and } y^H A = 0. \quad (4.6)$$

Likewise,

$$y^H B x = 0 \implies Bx = 0 \text{ and } y^H B = 0. \quad (4.7)$$

Proof. To establish (4.5) we may pass to the Schur form and assume that the pencil (A, B) has the form

$$\begin{pmatrix} \alpha & a^H \\ 0 & \hat{A} \end{pmatrix} - \lambda \begin{pmatrix} \beta & b^H \\ 0 & \hat{B} \end{pmatrix}. \quad (4.8)$$

Then x is a nonzero multiple of \mathbf{e}_1 . Now the first component of y is nonzero. For otherwise the vector \hat{y} consisting of the last $n-1$ components of y would be a left eigenvector of the pencil (\hat{A}, \hat{B}) , contradicting the simplicity of $\langle \alpha, \beta \rangle$. Hence $y^H x \neq 0$, and it follows from (4.8) that

$$\langle y^H A x, y^H B x \rangle = \langle y^H A \mathbf{e}_1, y^H B \mathbf{e}_1 \rangle = \langle \alpha y^H x, \beta y^H x \rangle = \langle \alpha, \beta \rangle.$$

To establish (4.6), we drop the Schur form and suppose that $y^H Ax = 0$. Then by (4.5) and the regularity of the pencil, $y^H Bx \neq 0$. Since Ax is proportional to Bx , we must have $Ax = 0$. Again, since $y^H A$ is proportional to $y^H B$, we must have $y^H A = 0$.

The implication (4.7) is established similarly. ■

Since $\alpha = y^H Ax$ and $\beta = y^H Bx$, we can write the ordinary form of the eigenvalue as

$$\lambda = \frac{y^H Ax}{y^H Bx}.$$

This expression is a natural generalization of the Rayleigh quotient, since B plays the role of the identity in the Rayleigh quotient for the ordinary eigenvalue problem.

This theorem will also prove important a little later when we consider the perturbation of eigenvalues of matrix pencils.

Perturbation theory

We now turn to perturbation theory for the generalized eigenvalue problem, and specifically to the perturbation of a simple eigenpair $(\langle \alpha, \beta \rangle, x)$ of a regular pencil (A, B) . Let

$$(\tilde{A}, \tilde{B}) = (A + E, B + F)$$

be the perturbed pencil. Let us agree to measure the size of this perturbation by

$$\epsilon = \sqrt{\|E\|_F^2 + \|F\|_F^2}.$$

We wish to know if there is an eigenpair $(\langle \tilde{\alpha}, \tilde{\beta} \rangle, \tilde{x})$ of (\tilde{A}, \tilde{B}) that converges to the eigenpair $(\langle \alpha, \beta \rangle, x)$ as $\epsilon \rightarrow 0$. We also wish to know how close the perturbed eigenpair is to the original as a function of ϵ .

We begin with a qualitative assessment of the effects of the perturbation (E, F) on the eigenvector x . By shifting the pencil, we can assume that B is nonsingular. Hence the problem can be reduced to the ordinary eigenvalue problem

$$(B + F)^{-1}(A + E)\tilde{x} = \tilde{\lambda}\tilde{x}.$$

Similarly, we have for the left eigenvector y

$$\tilde{y}^H(A + E)(B + F)^{-1} = \tilde{\lambda}\tilde{y}^H.$$

From the perturbation theory of inverses and Theorem 3.13, Chapter 1, we know that there exist perturbed eigenvectors \tilde{x} and \tilde{y} satisfying

$$\sin \angle(x, \tilde{x}), \sin \angle(y, \tilde{y}) = O(\epsilon). \quad (4.9)$$

If we normalize $x, \tilde{x}, y, \tilde{y}$ so that

$$\|x\|_2 = \|\tilde{x}\|_2 = \|y\|_2 = \|\tilde{y}\|_2 = 1,$$

then (4.9) implies that

$$\tilde{x} = x + O(\epsilon) \quad \text{and} \quad \tilde{y} = y + O(\epsilon). \quad (4.10)$$

For the eigenvalue we have from (4.5) that

$$\langle \tilde{\alpha}, \tilde{\beta} \rangle = \langle \tilde{y}^H \tilde{A} \tilde{x}, \tilde{y}^H \tilde{B} \tilde{x} \rangle = \langle y^H A x, y^H B x \rangle + O(\epsilon) = \langle \alpha, \beta \rangle + O(\epsilon).$$

[Here the notation $\langle \alpha, \beta \rangle + O(\epsilon)$ is an abbreviation for the more proper $\langle \alpha + O(\epsilon), \beta + O(\epsilon) \rangle$.] Thus the projective representation of the eigenvalues converges at least linearly in ϵ .

First-order expansions: Eigenvalues

We now turn to first-order expansions of a simple eigenvalue. We will use the notation established above.

Theorem 4.10. *Let x and y be simple eigenvectors of the regular pencil (A, B) , and let $\langle \alpha, \beta \rangle = \langle y^H A x, y^H B x \rangle$ be the corresponding eigenvalue. Then*

$$\langle \tilde{\alpha}, \tilde{\beta} \rangle = \langle \alpha + y^H E x, \beta + y^H F x \rangle + O(\epsilon^2). \quad (4.11)$$

Proof. The key to the proof lies in writing the perturbed eigenvectors \tilde{x} and \tilde{y} in appropriate forms. Since the pencil is regular, not both $y^H A$ and $y^H B$ can be zero. For definiteness assume that $y^H A = y^H B \neq 0$. Then by (4.6) we have $u^H x \neq 0$. This implies that if U is an orthonormal basis for the orthogonal complement of u then the matrix $(x \ U)$ is nonsingular. Hence we can write $\tilde{x} = \gamma x + U c$ for some γ and c . Now since $\tilde{x} \rightarrow x$ [see (4.10)], we must have $\gamma \rightarrow 1$. Hence renormalizing \tilde{x} by dividing by γ and setting $e = U c / \gamma$, we may write

$$\tilde{x} = x + e, \text{ where } y^H A e = y^H B e = 0, \text{ and } \|e\|_2 = O(\epsilon). \quad (4.12)$$

Similarly we may write

$$\tilde{y} = y + f, \text{ where } f^H A x = f^H B x = 0, \text{ and } \|f\|_2 = O(\epsilon). \quad (4.13)$$

Now

$$\begin{aligned} \tilde{\alpha} &= \tilde{y}^H \tilde{A} \tilde{x} \\ &= y^H A x + y^H E x + f^H A x + y^H A e + f^H A e \\ &= \alpha + y^H E x + f^H A e \quad [\text{by (4.12) and (4.13)}] \\ &= \alpha + y^H E x + O(\epsilon^2). \end{aligned}$$

Similarly, $\tilde{\beta} = \beta + y^H Fx + O(\epsilon^2)$. ■

Two observations on this theorem.

- The expression (4.11) can be written in the form

$$\langle \tilde{\alpha}, \tilde{\beta} \rangle = \langle y^H \tilde{A}x, y^H \tilde{B}x \rangle + O(\epsilon^2).$$

Hence if λ is finite,

$$\tilde{\lambda} = \frac{y^H \tilde{A}x}{y^H \tilde{B}x} + O(\epsilon^2).$$

This generalizes the Rayleigh-quotient representation of $\tilde{\lambda}$ in (3.21), Chapter 1.

- Since $\|x\|_2 = \|y\|_2 = 1$, equation (4.11) says that up to second-order terms $\tilde{\alpha}$ and $\tilde{\beta}$ lie in the intervals $[\alpha - \epsilon, \alpha + \epsilon]$ and $[\beta - \epsilon, \beta + \epsilon]$. We can convert these bounds into bounds on the ordinary form λ of the eigenvalue. However, the process is tedious and does not give much insight into what makes an eigenvalue ill conditioned. We therefore turn to a different way of measuring the accuracy of generalized eigenvalues.

The chordal metric

From (4.4) we see that the projective representation $\langle \alpha, \beta \rangle$ is the one-dimensional subspace of \mathbb{C}^2 spanned by the vector $(\alpha \ \beta)^T$. It is therefore natural to measure the distance between two eigenvalues $\langle \alpha, \beta \rangle$ and $\langle \gamma, \delta \rangle$ by the sine of the angle between them. Denoting this angle by θ , we have by the Cauchy inequality

$$\cos^2 \theta = \frac{|\alpha\gamma + \beta\delta|^2}{(|\alpha|^2 + |\beta|^2)(|\gamma|^2 + |\delta|^2)}.$$

Hence, after some manipulation

$$\sin^2 \theta = 1 - \cos^2 \theta = \frac{|\alpha\delta - \beta\gamma|^2}{(|\alpha|^2 + |\beta|^2)(|\gamma|^2 + |\delta|^2)}.$$

This justifies the following definition.

Definition 4.11. *The CHORDAL DISTANCE between $\langle \alpha, \beta \rangle$ and $\langle \gamma, \delta \rangle$ is the number*

$$\chi(\langle \alpha, \beta \rangle, \langle \gamma, \delta \rangle) = \frac{|\alpha\delta - \beta\gamma|}{\sqrt{|\alpha|^2 + |\beta|^2} \sqrt{|\gamma|^2 + |\delta|^2}}.$$

There are several comments to be made about this definition and its consequences.

- The function χ is a metric; that is, it is definite and satisfies the triangle inequality.
- If β and δ are nonzero and we set $\lambda = \alpha/\beta$ and $\mu = \gamma/\delta$, then

$$\chi(\langle \alpha, \beta \rangle, \langle \gamma, \delta \rangle) = \frac{|\lambda - \mu|}{\sqrt{1 + |\lambda|^2} \sqrt{1 + |\mu|^2}}.$$

Thus χ defines a distance between numbers in the complex plane. By abuse of notation we will denote this distance by $\chi(\lambda, \mu)$.

By a limiting argument, we can write

$$\chi(\lambda, \infty) = \frac{1}{\sqrt{1 + |\lambda|^2}}.$$

Thus the chordal distance includes the point at infinity. In particular, $\chi(0, \infty) = 1$.

- In Theorem 4.8, we saw how to shift a pencil and its eigenvalues by a nonsingular matrix W of order two. Specifically, the eigenvalue $\langle \alpha, \beta \rangle$ was shifted into $\langle (\alpha, \beta)W \rangle$. In general, the chordal metric is not invariant under such shifts. But if W is a nonzero multiple of a unitary matrix, the angle between two projective points is preserved, and the chordal distance between them remains unchanged.

- If $|\lambda|, |\mu| \leq 1$, then

$$\frac{1}{2}|\lambda - \mu| \leq \chi(\lambda, \mu) \leq |\lambda - \mu|. \quad (4.14)$$

Hence for eigenvalues that are not too large, the chordal metric behaves like the ordinary distance between two points in the complex plane.

- The name chordal distance comes from the fact that $\chi(\lambda, \mu)$ is half the length of the chord between the projections of λ and μ onto the Riemann sphere.

The condition of an eigenvalue

Turning now to our first-order expansion, we have

$$\langle \tilde{\alpha}, \tilde{\beta} \rangle \cong \langle \alpha + y^H Ex, \beta + y^H Fx \rangle,$$

from which we get

$$\chi(\langle \alpha, \beta \rangle, \langle \tilde{\alpha}, \tilde{\beta} \rangle) \cong \frac{|\alpha y^H Fx - \beta y^H Ex|}{|\alpha|^2 + |\beta|^2}.$$

But the numerator of this expression is

$$\left| \langle \alpha, \beta \rangle \begin{pmatrix} y^H Fx \\ y^H Ex \end{pmatrix} \right| \leq \epsilon \|x\|_2 \|y\|_2 \sqrt{|\alpha|^2 + |\beta|^2}.$$

Hence

$$\chi(\langle \alpha, \beta \rangle, \langle \tilde{\alpha}, \tilde{\beta} \rangle) \lesssim \frac{\|x\|_2 \|y\|_2}{\sqrt{|\alpha|^2 + |\beta|^2}} \epsilon.$$

We summarize our results in the following theorem, in which we recapitulate a bit.

Theorem 4.12. Let λ be a simple eigenvalue (possibly infinite) of the pencil (A, B) and let x and y be its right and left eigenvectors. Let the projective representation of λ be $\langle \alpha, \beta \rangle$, where

$$\alpha = y^H A x \quad \text{and} \quad \beta = y^H B x. \quad (4.15)$$

Let $\tilde{A} = A + E$ and $\tilde{B} = B + F$, and set

$$\epsilon = \sqrt{\|E\|_F^2 + \|F\|_F^2}.$$

Then for ϵ sufficiently small, there is an eigenvalue $\tilde{\lambda}$ the pencil (\tilde{A}, \tilde{B}) satisfying

$$\chi(\lambda, \tilde{\lambda}) \leq \nu \epsilon + O(\epsilon^2),$$

where

$$\nu = \frac{\|x\|_2 \|y\|_2}{\sqrt{|\alpha|^2 + |\beta|^2}}.$$

There are five comments to be made on this theorem.

- The number ν mediates the effects of the error ϵ on the perturbation of the eigenvalue and is therefore a condition number for the problem. By (4.15) it is independent of the scaling of x and y .
- If $\|x\|_2 = \|y\|_2 = 1$, the condition number ν is large precisely when α and β are both small. This agrees with common-sense reasoning from the first-order perturbation expansion

$$\langle \tilde{\alpha}, \tilde{\beta} \rangle \cong \langle \alpha + y^H E x, \beta + y^H F x \rangle.$$

If both α and β are small, they are sensitive to the perturbation E and F , which means that the subspace $\langle \tilde{\alpha}, \tilde{\beta} \rangle$ is ill determined.

- On the other hand, if one of α or β is large, then ν is not large and the eigenvalue is well conditioned. In particular, we may have $\beta = 0$, so that the eigenvalue λ is infinite. In this case, although λ moves infinitely far in the complex plane, it remains in a small neighborhood of infinity on the Riemann sphere. Such are the regularizing properties of the chordal metric.
- The bound does not reduce to the ordinary first-order bound

$$|\tilde{\lambda} - \lambda| \lesssim \sec \angle(x, y) \|E\|_2$$

when $B = I$ and $F = 0$. However, if we normalize A so that $\|A\|_2 = 1$ and normalize x and y so that $y^H x = 1$, then $|\lambda| \leq 1$ and $\|x\|_2 \|y\|_2 = \sec \angle(x, y)$. Hence,

$$\frac{\sec \angle(x, y)}{\sqrt{2}} \leq \nu \leq \sec \angle(x, y).$$

Moreover, by (4.14),

$$\frac{1}{2}|\tilde{\lambda} - \lambda| \lesssim \chi(\tilde{\lambda}, \lambda) \lesssim |\lambda - \mu|.$$

Hence, for the case $\|A\|_2 = 1$ the results of ordinary perturbation theory and our generalized theory agree up to constants near one.

- Unfortunately, ill-conditioned eigenvalues can restrict the range of perturbations for which the first-order expansion of a well-conditioned eigenvalue remains valid, as the following example shows.

Example 4.13. Consider the pencil

$$(A, B) = \left[\begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 2\epsilon \end{pmatrix} \right],$$

where ϵ is small. If we perturb it to get the pencil

$$(\tilde{A}, \tilde{B}) = \left[\begin{pmatrix} 1 & \epsilon \\ \epsilon & 2\epsilon \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix} \right],$$

then

$$AB^{-1} = \begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix}.$$

Thus the eigenvalues of the perturbed pencil are $1 \pm \sqrt{\epsilon}$. In particular, if $\epsilon = 10^{-10}$, the “well-conditioned” eigenvalue 1 has changed by 10^{-5} .

The example shows that perturbations can cause an ill-conditioned eigenvalue to wander around, bumping into other eigenvalues and destroying their accuracy. However, this problem should not be taken too seriously. It is obvious from the example that only a very special perturbation can undo a well-conditioned eigenvalue. Such perturbations are unlikely to arise from essentially random sources, such as rounding or measurement error.

Perturbation expansions: Eigenvectors

To derive a perturbation expansion for a simple eigenvector x of regular eigenpair (A, B) , we appeal to the first step of the reduction to Schur form [see (4.2)]. Specifically, let $(x \ X)$ and $(u \ U)$ be unitary matrices such that

$$(u \ U)^H(A, B)(x \ X) = \left[\begin{pmatrix} \alpha & a^H \\ 0 & \hat{A} \end{pmatrix}, \begin{pmatrix} \beta & a^H \\ 0 & \hat{B} \end{pmatrix} \right]. \quad (4.16)$$

The fact that we can choose U so $U^H A x = U^H B x = 0$ is a consequence of the fact that x is an eigenvector. Conversely, if such a U exists, then $(u \ U)^H(A, B)(x \ X)$ must be of the form (4.16), and x must be an eigenvector.

Now let $(\tilde{A}, \tilde{B}) = (A + E, B + F)$. We seek the perturbed eigenvector in the form $\tilde{x} = (x + Xp)$, where p is to be determined. We also seek the corresponding matrix \tilde{U} in the form $\tilde{U} = (U + uq^H)(I + qq^H)^{-\frac{1}{2}}$, where q is also to be determined. Since we know that \tilde{x} converges to x as $\epsilon = \sqrt{\|E\|_F^2 + \|F\|_F^2} \rightarrow 0$, we may assume that p and q are small. In particular, the factor $(I + qq^H)^{-\frac{1}{2}}$, which makes \tilde{U} orthonormal, differs from the identity by second-order terms in q , and we will drop it in what follows.

For \tilde{x} to be an eigenvector of (\tilde{A}, \tilde{B}) , we must have $\tilde{U}^H \tilde{A} \tilde{x} = \tilde{U}^H \tilde{B} \tilde{x} = 0$. But

$$\begin{aligned}\tilde{U}^H \tilde{A} \tilde{x} &= (U + uq^H)^H (A + E)(x + Xp) \\ &\cong U^H Ax + U^H Ex + qu^H Ax + U^H AXp \\ &= U^H Ex + \alpha q + \hat{A}p,\end{aligned}$$

the last equality following from (4.16). Similarly, $\tilde{U}^H \tilde{B} \tilde{x} \cong U^H Fx + \beta q + \hat{B}p$. Hence p and q must satisfy the equations

$$\begin{aligned}\hat{A}p + \alpha q &\cong -U^H Ex, \\ \hat{B}p + \beta q &\cong -U^H Fx.\end{aligned}$$

We may eliminate the auxiliary vector q from these equations by multiplying the first by β and the second by α and subtracting to get

$$(\beta \hat{A} - \alpha \hat{B})p \cong -\beta U^H Ex + \alpha U^H Fx. \quad (4.17)$$

Now the matrix $\beta \hat{A} - \alpha \hat{B}$ is nonsingular. For otherwise (α, β) would be an eigenvalue of (\hat{A}, \hat{B}) , contradicting the simplicity of the eigenvalue (α, β) . Thus

$$\tilde{x} = x + X(\beta \hat{A} - \alpha \hat{B})^{-1}(-\beta U^H Ex + \alpha U^H Fx) + O(\epsilon^2), \quad (4.18)$$

where

$$\epsilon = \sqrt{\|E\|_F^2 + \|F\|_F^2}.$$

The condition of an eigenvector

To convert (4.18) to a first order-bound, note that we can multiply (4.17) by a constant to make $\max\{|\alpha|, |\beta|\} = 1$. Moreover, by Lemma 3.12, Chapter 1, up to second-order terms $\sin \angle(x, \tilde{x}) \cong \|X^H \tilde{x}\|_2$. Finally, since X and U are orthonormal, $\|X^H U\|_2 \leq 1$. Hence on taking norms we get:

In the above notation suppose that $\max\{|\alpha|, |\beta|\} = 1$ and set

$$\delta[(\alpha, \beta), (\hat{A}, \hat{B})] = \|(\beta \hat{A} - \alpha \hat{B})^{-1}\|_2^{-1}.$$

Then

$$\sin \angle(x, \tilde{x}) \leq \frac{\epsilon}{\delta[(\alpha, \beta), (\hat{A}, \hat{B})]} + O(\epsilon^2). \quad (4.19)$$

The number $\delta[(\alpha, \beta), (\hat{A}, \hat{B})]$ is analogous to the number $\text{sep}(\lambda, M)$ for the ordinary eigenvalue problem [see (3.16), Chapter 1]. It is nonzero if $\langle \alpha, \beta \rangle$ is simple, and it approaches zero as $\langle \alpha, \beta \rangle$ approaches an eigenvalue of (\hat{A}, \hat{B}) . Most important, as (4.19) shows, its reciprocal is a condition number for the eigenvector corresponding to $\langle \alpha, \beta \rangle$.

4.2. REAL SCHUR AND HESSENBERG-TRIANGULAR FORMS

We now turn to the computational aspects of the generalized eigenvalue problem. As indicated in the introduction to this section, the object of our computation is a Schur form, which we will approach via an intermediate Hessenberg-triangular form. The purpose of this subsection is to describe the real generalized Schur form that is our computational goal and to present the reduction to Hessenberg-triangular form.

Generalized real Schur form

Let (A, B) be a real, regular pencil. Then (A, B) can be reduced to the equivalent of the real Schur form (Theorem 3.1); i.e., there is an orthogonal equivalence transformation that reduces the pencil to a quasi-triangular form in which the 2×2 blocks contain complex conjugate eigenvalues. Specifically, we have the following theorem.

Theorem 4.14. *Let (A, B) be a real regular pencil. Then there are orthogonal matrices U and V such that*

$$S = U^T A V = \begin{pmatrix} S_{11} & S_{12} & S_{13} & \cdots & S_{1k} \\ 0 & S_{22} & S_{23} & \cdots & S_{2k} \\ 0 & 0 & S_{33} & \cdots & S_{3k} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & S_{kk} \end{pmatrix}$$

and

$$T = U^T B U = \begin{pmatrix} T_{11} & T_{12} & T_{13} & \cdots & T_{1k} \\ 0 & T_{22} & T_{23} & \cdots & T_{2k} \\ 0 & 0 & T_{33} & \cdots & T_{3k} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & T_{kk} \end{pmatrix}.$$

The diagonal blocks of S and T are of order either one or two. The pencils corresponding to the blocks of order one contain the real eigenvalues of (A, B) . The pencils corresponding to the blocks of order two contain a pair of complex conjugate eigenvalues of (A, B) . The blocks can be made to appear in any order.

The proof of this theorem is along the lines of the reduction to generalized Schur form, the chief difference being that we use two-dimensional subspaces to deflate the complex eigenvalues. Rather than present a detailed description of this deflation, we sketch the procedure.

1. Using Definition 4.6 and Theorem 4.8, argue that complex eigenvalues of a real pencil occur in conjugate pairs and that the real and imaginary parts of their eigenvectors are independent.
2. Let $x = y + iz$ be an eigenvector corresponding to a complex eigenvalue and let $X = (y \ z)$. Show that for some matrix L , we have $AX = BXL$.
3. Let $(V \ V_\perp)$ be orthogonal with $\mathcal{R}(V) = \mathcal{R}(X)$. Let $(U \ U_\perp)$ be orthogonal with $\mathcal{R}(U) = \mathcal{R}(AV) \cup \mathcal{R}(BV)$.
4. Show that

$$(U \ U_\perp)^H (A, B) (V \ V_\perp) = \left[\begin{pmatrix} S & G \\ 0 & \hat{A} \end{pmatrix}, \begin{pmatrix} T & H \\ 0 & \hat{B} \end{pmatrix} \right],$$

where S and T are 2×2 .

By repeated applications of this deflation of complex eigenvalues along with the ordinary deflation of real eigenvalues, we can prove Theorem 4.14.

Hessenberg-triangular form

The first step in the solution of the ordinary eigenvalue problem by the QR algorithm is a reduction of the matrix in question to Hessenberg form. In this section, we describe a reduction of a pencil (A, B) to a form in which A is upper Hessenberg and B is triangular. Note that if A is upper Hessenberg and B is lower triangular, then AB^{-1} is upper Hessenberg. Thus the reduction to Hessenberg-triangular form corresponds to the reduction to Hessenberg form of the matrix AB^{-1} .

The process begins by determining an orthogonal matrix Q such that $Q^T B$ is upper triangular. The matrix Q can be determined as a product of Householder transformations (Algorithm I:4.1.2). The transformation Q^T is also applied to A .

We will now use plane rotations to reduce A to Hessenberg form while preserving the upper triangularity of B . The reduction proceeds by columns. Figure 4.1 illustrates the reduction of the first column. Zeros are introduced in A beginning at the bottom of the first column. The elimination of the $(5, 1)$ -element of A by premultiplication by the rotation Q_{45} in the $(4, 5)$ -plane introduces a nonzero into the $(5, 4)$ -element of B . This nonzero element is then annihilated by postmultiplication by a rotation Z_{54} in the $(5, 4)$ -plane. The annihilation of the $(4, 1)$ - and $(3, 1)$ -elements of A proceeds similarly.

If this process is repeated on columns 2 through $n-2$ of the pencil, the result is Algorithm 4.1. Here are some comments.

- We have implemented the algorithm for real matrices. However, with minor modifications it works for complex matrices.
- The computation of Q is special. Since Q is to contain the transpose of the accumulated transformations, we must postmultiply by the transformations, even though they premultiply A and B .

$$\begin{array}{c}
 \left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ \hat{a} & a & a & a & a \end{pmatrix} \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right] Q_{45}(A,B) \Rightarrow \\
 \left[\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} \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & \hat{b} & b \end{pmatrix} \right] (A,B)Z_{54} \Rightarrow \\
 \left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ \hat{a} & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix} \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right] Q_{34}(A,B) \Rightarrow \\
 \left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix} \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & \hat{b} & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right] (A,B)Z_{43} \Rightarrow \\
 \left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ \hat{a} & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix} \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right] Q_{23}(A,B) \Rightarrow \\
 \left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix} \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & \hat{b} & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right] (A,B)Z_{32} \Rightarrow \\
 \left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix} \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right]
 \end{array}$$

Figure 4.1: Reduction to Hessenberg-triangular form

This algorithm takes a matrix pencil of order n and reduces it to Hessenberg-triangular form by orthogonal equivalences. The left transformations are accumulated in the matrix Q , the right transformations in the matrix Z .

```

1. HessTriForm(A, B, Q, Z)
    Reduce B to triangular form
2.  $Q = I$ 
3. for  $k = 1$  to  $n-1$ 
4.   housegen(B[k:n, k], u, temp)
5.    $v^T = u^T * B[k:n, k:n]$ 
6.    $B[k:n, k:n] = B[k:n, k:n] - u * v^T$ 
7.    $v^T = u^T * A[k:n, k:n]$ 
8.    $A[k:n, :] = A[k:n, :] - u * v^T$ 
9.    $v = Q[:, k:n] * u$ 
10.   $Q[:, k:n] = Q[:, k:n] - v * u^T$ 
11. end for  $k$ 
    Reduce A to Hessenberg form
12.  $Z = I$ 
13. for  $k = 1$  to  $n-2$ 
14.   for  $j = n-1$  to  $k+1$  by -1
15.     rotgen(A[j, k], A[j+1, k], c, s)
16.     rotapp(c, s, A[j, k+1:n], A[j+1, k+1:n])
17.     rotapp(c, s, B[j, j:n], B[j+1, j:n])
18.     rotapp(c, s, Q[:, j], Q[:, j+1])
19.     rotgen(B[j+1, j+1], B[j+1, j], c, s)
20.     rotapp(c, s, B[1:j, j+1], B[1:j, j])
21.     rotapp(c, s, A[:, j+1], A[:, j])
22.     rotapp(c, s, Z[:, j+1], Z[:, j])
23.   end for  $j$ 
24. end for  $k$ 
25. end hess-triform
```

Algorithm 4.1: Reduction to Hessenberg-triangular form

- The operation count for the reduction of B is $2\frac{2}{3}n^3$ flam. The reduction of A takes $2\frac{5}{6}n^3$ firot. Thus for real matrices, the algorithm requires

$$8\frac{1}{3}n^3 \text{ fladd} + 13\frac{2}{3}n^3 \text{ flmlt.}$$

- If only eigenvalues of the pencil are desired, we need not accumulate the transformations.
- The algorithm is backward stable in the usual sense.

4.3. THE DOUBLY SHIFTED QZ ALGORITHM

Like the QR algorithm, the doubly shifted QZ algorithm is an iterative reduction of a real Hessenberg-triangular pencil to real generalized Schur form. Since many of the details — such as back searching — are similar to those in the QR algorithm, we will not attempt to assemble the pieces into a single algorithm. Instead we begin with the general strategy of the QZ algorithm and then treat the details.

Overview

Let (A, B) be in Hessenberg-triangular form and assume that B is nonsingular. Then the matrix $C = AB^{-1}$ is Hessenberg and is a candidate for an implicit double QR step. The idea underlying the QZ algorithm is to mimic this double QR step by manipulations involving only A and B . In outline the algorithm goes as follows. [Before proceeding, the reader may want to review the outline (3.6) of the doubly shifted QR algorithm.]

- Let H be the Householder transformation that starts a double shift QR step on the matrix AB^{-1} .
- Determine orthogonal matrices Q and Z with $Q\mathbf{e}_1 = \mathbf{e}_1$ such that $(\hat{A}, \hat{B}) = Q^T(HA, HB)Z$ is in Hessenberg-triangular form.

To see why this algorithm can be expected to work, let $\hat{C} = \hat{A}\hat{B}^{-1}$. Then

$$\hat{C} = Q^T H^T C H Q.$$

Moreover, since $Q\mathbf{e}_1 = \mathbf{e}_1$, the first column of HQ is the same as that of H . It follows that \hat{C} is the result of performing an implicit double QR step on C . In consequence, we can expect at least one of the subdiagonal elements $c_{n,n-1}$ and $c_{n-1,n-2}$ to converge to zero. But because (A, B) is in Hessenberg-triangular form,

$$a_{n,n-1} = c_{n,n-1}b_{n-1,n-1} \quad \text{and} \quad a_{n-1,n-2} = c_{n-1,n-2}b_{n-2,n-2}.$$

Hence if $b_{n-1,n-1}$ and $b_{n-2,n-2}$ do not approach zero, at least one of the subdiagonal elements $a_{n,n-1}$ and $a_{n-1,n-2}$ must approach zero. On the other hand, if either $b_{n-1,n-1}$ or $b_{n-2,n-2}$ approaches zero, the process converges to an infinite eigenvalue, which can be deflated.

If repeated QZ steps force $a_{n,n-1}$ to zero, then the problem deflates with a real eigenvalue. If, on the other hand, $a_{n-1,n-2}$ goes to zero, a 2×2 block, which may contain real or complex eigenvalues, is isolated. In either case, the iteration can be continued with a smaller matrix.

Getting started

We now turn to the details of the QZ algorithm. The first item is to compute the Householder transformation to start the QZ step. In principle, this is just a matter of computing the appropriate elements of C and using Algorithm 3.1. In practice, the formulas can be simplified. In particular, if we set $m = n-1$, the three components of the vector generating the Householder transformation are given by

$$\begin{aligned} p_1 &= \left[\left(\frac{a_{mm}}{b_{mm}} - \frac{a_{11}}{b_{11}} \right) \left(\frac{a_{nn}}{b_{nn}} - \frac{a_{11}}{b_{11}} \right) - \left(\frac{a_{mn}}{b_{mn}} \right) \left(\frac{a_{nm}}{b_{nm}} \right) \right. \\ &\quad \left. + \left(\frac{a_{mn}}{b_{mn}} \right) \left(\frac{a_{nm}}{b_{nm}} \right) \left(\frac{a_{11}}{b_{11}} \right) \right] \cdot \left(\frac{b_{11}}{a_{21}} \right) \\ &\quad + \left(\frac{a_{12}}{b_{22}} \right) - \left(\frac{a_{11}}{b_{11}} \right) \left(\frac{b_{11}}{b_{22}} \right), \\ p_2 &= \left(\frac{a_{22}}{b_{22}} - \frac{a_{11}}{b_{11}} \right) - \left(\frac{a_{21}}{b_{11}} \right) \left(\frac{b_{12}}{b_{22}} \right) \\ &\quad - \left(\frac{a_{mm}}{b_{mm}} - \frac{a_{11}}{b_{11}} \right) - \left(\frac{a_{nn}}{b_{nn}} - \frac{a_{11}}{b_{11}} \right) + \left(\frac{a_{nm}}{b_{mn}} \right) \left(\frac{b_{mn}}{b_{nn}} \right), \\ p_3 &= \frac{a_{32}}{b_{22}}. \end{aligned}$$

In EISPACK and LAPACK these formulas are further simplified to reduce the number of divisions.

The formulas are not well defined if any of the diagonal elements b_{11} , b_{22} , b_{mm} , or b_{mn} is zero. For the case of zero b_{11} , the problem can be deflated (see p. 153). If the others diagonals are too small, they are replaced by $\epsilon_M \|B\|$, where B is a suitable norm.

The above formulas do not use the Francis double shift computed from the matrix C . Instead they shift by the eigenvalues of the the trailing 2×2 pencil of (A, B) : namely,

$$\begin{pmatrix} a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{nn} \end{pmatrix} - \lambda \begin{pmatrix} b_{n-1,n-1} & b_{n-1,n} \\ b_{n,n-1} & b_{nn} \end{pmatrix}.$$

This approach is simpler and is just as effective.

The QZ step

Having now determined a 3×3 Householder transformation to start the QZ step, we apply it to the pencil (A, B) and reduce the result back to Hessenberg-triangular form. As usual we will illustrate the process using Wilkinson diagrams. Our original pencil

has the form

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

If we premultiply this pencil by the Householder transformation computed above, we get a pencil of the form

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ b^2 & b & b & b & b \\ b^1 & b^1 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right]. \quad (4.20)$$

The next step reduces B back to triangular form. First postmultiply by a Householder transformation to annihilate the elements with superscripts 1. We follow with a plane rotation that annihilates the element with superscript 2. This produces a pencil of the form

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a^1 & a & a & a & a \\ a^1 & a & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

We now premultiply by a Householder transformation to annihilate the elements of A with superscripts 1. This results in a pencil of the form

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & b^2 & b & b & b \\ 0 & b^1 & b^1 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

Once again we reduce B back to triangular form using a Householder transformation and a plane rotation:

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & a^1 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

When we repeat the procedure this time, we need only use a plane rotation to zero out the $(5, 3)$ -element of A . The result is a matrix that is almost in Hessenberg-triangular form:

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & b^1 & b \end{pmatrix} \right].$$

A postmultiplication by a plane rotation annihilates the $(5, 5)$ -element of B without affecting the Hessenberg form of A .

Since the QZ algorithm, like the QR algorithm, works with increasingly smaller matrices as the problem deflates, the sweep will be applied only to the working matrix. Algorithm 4.2 implements a sweep between rows $i1$ and $i2$. In studying this algorithm, keep in mind that $k-1$ points to the column of A from which the current Householder transformation is obtained (except for the first, which is provided by the input to the algorithm). Also note that the Householder transformation that postmultiplies (A, B) reduces the three elements of B to a multiple of e_3^T — not e_1 as *housegen* assumes. We therefore use the *cross matrix*

$$F = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

in statements 10 and 12 to reverse the order of the components.

There are four comments to be made about this algorithm.

- An operation count shows that the sweep requires

$$12n(i2 - i1) \text{ flam} + 2(i2^2 - i1^2) \text{ flrot.} \quad (4.21)$$

Thus a QZ sweep is a little over twice as expensive as a doubly shifted QR sweep.

- The extra rotations in (4.21) are due to the necessity of eliminating b^2 in (4.20). An alternative is to premultiply by a Householder transformation to eliminate the elements b^1 in the pencil

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ b^1 & b & b & b & b \\ b^1 & b & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

Given a real pencil (A, B) in Hessenberg-triangular form, and a vector u that generates a starting Householder transformation, *qz2step* implements a QZ sweep between rows $i1$ and $i2$.

```

1.  qz2step(A, B, u, i1, i2, Q, Z)
2.  for k = i1 to i2-2
3.    if (k ≠ i1)
4.      housegen(A[k:k+2, k-1], u, ν)
5.      A[k, k-1] = ν; A[k+1:k+2, k-1] = 0
6.    end if
7.    vT = uT*A[k:k+2, k:n]; A[k:k+2, k:n] = A[k:k+2, k:n] - uvT
8.    vT = uT*B[k:k+2, k:n];
      B[k:k+2, k:n] = B[k:k+2, k:n] - uvT
9.    v = Q[:, k:k+2]*u; Q[:, k:k+2] = Q[:, k:k+2] - vuT
10.   housegen(B[k+2, k:k+2]*f, uT, ν)
11.   B[k+2, k+2] = ν; B[k+2, k:k+1] = 0
12.   uT = uT*f
13.   v = B[1:k+1, k:k+2]*u;
      B[1:k+1, k:k+2] = B[1:k+1, k:k+2] - vuT
14.   kk = min{k+3, i2}
15.   v = A[1:kk, k:k+2]*u; A[1:kk, k:k+2] = A[1:kk, k:k+2] - vuT
16.   v = Z[:, k:k+2]*u; Z[:, k:k+2] = Z[:, k:k+2] - vuT
17.   rotgen(B[k+1, k+1], B[k+1, k], c, s)
18.   rotapp(c, s, B[1:k, k+1], B[1:k, k])
19.   rotapp(c, s, A[1:kk, k+1], A[1:kk, k])
20.   rotapp(c, s, Z[:, k+1], Z[:, k])
21. end for k
22. rotgen(A[i2-1, i2-2], A[i2, i2-2], c, s)
23. rotapp(c, s, A[i2-1, i2-1:i2], (A[i2, i2-1:i2]))
24. rotapp(c, s, B[i2-1, i2-1:i2], (B[i2, i2-1:i2]))
25. rotapp(c, s, Q[:, i2-1], Q[:, i2])
26. rotgen(B[i2, i2], B[i2, i2-1], c, s)
27. rotapp(c, s, B[1:i2-1, i2], B[1:i2-1, i2-1])
28. rotapp(c, s, A[1:i2, i2], A[1:i2, i2-1])
29. rotapp(c, s, Z[:, i2], Z[:, i2-1])
30. end qz2step

```

Algorithm 4.2: The doubly shifted QZ step

This gives an operation count of

$$12n(i2 - i1) \text{ flam},$$

which is exactly twice the count for a double QR sweep. For more see the notes and references.

- If only eigenvalues are desired, the sweep can be performed only on the sub-pencil $(A[i1 : i2, i1 : i2], B[i1 : i2, i1 : i2])$ and the transformations need not be accumulated.
- The algorithm is stable in the usual sense.

4.4. IMPORTANT MISCELLANEA

Anyone attempting to code a QZ-style algorithm from the description in the last two sections will quickly find how sketchy it is. The purpose of this subsection is to fill the gaps—at least to the extent they can be filled in a book of this scope. We will treat the following topics: balancing, back searching, infinite eigenvalues, 2×2 problems, and eigenvectors.

Balancing

The natural class of transformations for balancing the pencil (A, B) are the diagonal equivalences. Thus we should like to find diagonal matrices D_R and D_C with positive diagonals so that the nonzero elements of $D_R A D_C$ and $D_R B D_C$ are as nearly equal in magnitude as possible. If we write

$$D_R = \text{diag}(r_1, \dots, r_n) \quad \text{and} \quad D_C = \text{diag}(c_1, \dots, c_n),$$

then we wish to choose positive numbers r_1, \dots, r_n and c_1, \dots, c_n so that all the elements $r_i c_j |a_{ij}|$ and $r_i c_j |b_{ij}|$ of $D_R A D_C$ and $D_R B D_C$ are nearly equal.

This problem is not well determined, since multiplying the pencil (A, B) by a scalar does not change the balance. But we can make it well determined by specifying a value around which the scaled elements must cluster. If we take that value to be one, then our problem is to choose positive numbers r_1, \dots, r_n and c_1, \dots, c_n so that

$$r_i c_j |a_{ij}|, r_i c_j |b_{ij}| \cong 1. \tag{4.22}$$

The problem (4.22) is nonlinear in the unknowns r_i and c_j . However, if we set

$$\rho_i = \log r_i \quad \text{and} \quad \gamma_j = \log c_j,$$

then the problem is equivalent to finding quantities ρ_i and γ_j so that

$$\log |a_{ij}| + \gamma_i + \rho_j \cong 0 \quad \text{and} \quad \log |b_{ij}| + \gamma_i + \rho_j \cong 0.$$

If we solve this problem in the least squares sense, we end up with the problem

$$\sum_{a_{ij} \neq 0} (\log |a_{ij}| + \gamma_i + \rho_j)^2 + \sum_{b_{ij} \neq 0} (\log |b_{ij}| + \gamma_i + \rho_j)^2 = \min. \tag{4.23}$$

Without going into details, the normal equations for (4.23) are very simple. Although the system is too large to solve directly, it can be solved by an iterative technique called the method of conjugate gradients. Each iteration requires $O(n)$ operations, and the method converges so swiftly that the determination of the scaling factors is also an $O(n)$ process (remember that we do not have to determine the scaling factors very accurately).

The procedure sketched above is not invariant under change of scale of A and B — i.e., it will produce different results for the pair $(\sigma A, \tau B)$ as σ and τ vary. Consequently, it is recommended that A and B be normalized so that their norms are one in some easily calculated, balanced norm. This scaling also helps in controlling underflow and overflow in the QZ step.

Back searching

The QZ step implemented in Algorithm 4.2 is performed between rows $i1$ and $i2$ of A and B , where the elements $A[i1-1, i1]$ and $A[i2-1, i2]$ have been deemed negligible. The integers $i1$ and $i2$ are determined by a back-searching algorithm analogous to Algorithm 2.6.

The chief problem is to decide when an element is negligible. The usual procedure is to assume that the problem is reasonably balanced and declare an element a_{ij} negligible if

$$|a_{ij}| \leq \epsilon_M \|A\|,$$

in some convenient norm. A similar criterion is used for the elements of B .

Infinite eigenvalues

If, in the course of the QZ iteration, a negligible element occurs on the diagonal of B , the pencil has an infinite eigenvalue. There are many ways of treating infinite eigenvalues. The following is a simple and efficient deflation procedure.

Consider the Wilkinson diagram

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & \hat{b} & b & b \\ 0 & 0 & 0 & 0 & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right],$$

in which the fourth diagonal element of B is zero. We can postmultiply this pencil by a rotation in the $(4, 3)$ -plane to annihilate the hatted element in B . The result is the pencil

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & \hat{a} & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

We may now premultiply a rotation in the $(4, 5)$ -plane to return A to Hessenberg form:

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & \hat{b} & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

We can now repeat the process by postmultiplying by a rotation in the $(3, 2)$ -plane to get

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & \hat{a} & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right]$$

and then premultiplying by a rotation in the $(3, 4)$ -plane to get

$$\left[\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} b & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

One more step of post- and premultiplication gives

$$\left[\begin{pmatrix} a & a & a & a & a \\ \hat{a} & a & a & a & a \\ 0 & a & a & a & a \\ 0 & 0 & a & a & a \\ 0 & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \\ 0 & 0 & 0 & 0 & b \end{pmatrix} \right].$$

Finally premultiplication by a rotation in the $(1, 2)$ -plane gives

$$\left[\begin{pmatrix} a & | & a & a & a & a \\ 0 & | & a & a & a & a \\ 0 & | & a & a & a & a \\ 0 & | & 0 & a & a & a \\ 0 & | & 0 & 0 & a & a \end{pmatrix}, \begin{pmatrix} 0 & | & b & b & b & b \\ 0 & | & b & b & b & b \\ 0 & | & 0 & b & b & b \\ 0 & | & 0 & 0 & b & b \\ 0 & | & 0 & 0 & 0 & b \end{pmatrix} \right].$$

As the lines in the above diagram show, the infinite eigenvalue has been deflated from the problem.

We can also deflate an infinite eigenvalue if there are two small consecutive diagonal elements. Specifically, consider the matrix

$$\begin{pmatrix} b & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & \epsilon_1 & c \\ 0 & 0 & 0 & 0 & \epsilon_2 \end{pmatrix}.$$

If we apply a plane rotation in the $(5, 4)$ -plane to annihilate ϵ_1 , then the $(5, 4)$ -element becomes

$$-\frac{\epsilon_1 \epsilon_2}{\sqrt{\epsilon_1^2 + c^2}}.$$

Consequently, if $\epsilon_1 \epsilon_2 / c$ is negligible, we can set the $(5, 4)$ -element to zero and deflate the zero $(4, 4)$ -element as described above.

The 2×2 problem

When the QZ iteration deflates a 2×2 pencil, it may have real eigenvalues. In that case, we will naturally want to deflate it further, so that the real eigenvalues appear explicitly on the diagonal of the Schur form. This amounts to computing the Schur form of a pencil of order two. This, it turns out, is not an easy problem when the effects of rounding error are taken into account. The details are too involved to give here, and we only warn the reader not to go it alone. Copy the EISPACK or LAPACK code.

Eigenvectors

Eigenvectors can be computed by a back substitution algorithm analogous to the ones for the ordinary eigenvalue problem. Unlike the generalized eigenproblem of order two, this algorithm contains no significant numerical pitfalls. However, its derivation is lengthy, and once again we refer the reader to the EISPACK or LAPACK code.

4.5. NOTES AND REFERENCES

The algebraic background

The generalized eigenvalue problem (regarded as a pair of bilinear forms) goes back at least to Weierstrass [293, 1868], who established the equivalent of the Jordan form for regular matrix pencils. Jordan [137, 1874] later gave a new proof for singular pencils. Kronecker [152, 1890] extended the result to rectangular pencils. For modern treatments see [80, 141, 269, 281, 304].

The generalized Schur form is due to Stewart [251].

Perturbation theory

For a general treatment of the perturbation theory of the generalized eigenvalue problem along with a bibliography see [269]. The use of the chordal metric in generalized

eigenvalue problems is due to Stewart [255], who is happy to acknowledge a broad hint from W. Kahan. For a chordless perturbation theory and backward error bounds see [112].

The QZ algorithm

The QZ algorithm, including the preliminary reduction to Hessenberg-triangular form, is due to Moler and Stewart [178]. The double shift strategy, of course, works when the eigenvalues of the trailing pencil are real. However, Ward [286] has observed that the algorithm performs better if real shifts are treated by a single shift strategy. This option has been incorporated into the EISPACK and LAPACK codes.

The balancing algorithm is due to Ward [287], who builds on earlier work of Curtis and Reid [50]. Ward [286] and Watkins [291] consider the problem of infinite eigenvalues. In particular, both the reduction to Hessenberg–triangular form and the steps of the QZ algorithm move zero elements on the diagonal of B toward the top. Moreover, Watkins shows that the presence of small or zero elements on the diagonal of B does not interfere with the transmission of the shift. Thus an alternative to deflating infinite eigenvalues from the middle of the pencil is to wait until they reach the top, where they can be deflated by a single rotation.

For the treatment of 2×2 blocks see the EISPACK code `qzval` or the LAPACK code `SLAG2`.

3

THE SYMMETRIC EIGENVALUE PROBLEM

We have seen that symmetric (or Hermitian) matrices have special mathematical properties. In particular, their eigenvalues are real and their eigenvectors can be chosen to form an orthonormal basis for \mathbb{R}^n . They also have special computational properties. Because of their symmetry they can be stored in about half the memory required for a general matrix. Moreover, we can frequently use the symmetry of the problem to reduce the operation count in certain algorithms. For example, the Cholesky algorithm for triangularizing a symmetric matrix requires half the operations that are required by its nonsymmetric counterpart, Gaussian elimination (see Algorithm I:2.2.1).

The first part of this chapter is devoted to efficient algorithms for the symmetric eigenvalue problem. In the first section we will show how to adapt the QR algorithm to solve the symmetric eigenvalue problem. The procedure begins as usual with a reduction to a simpler form — in this case tridiagonal form — followed by a QR iteration on the reduced form. We will treat this reduction in §1.1 and the tridiagonal QR iteration itself in §1.2.

In §2 we will consider some alternative methods for the symmetric eigenvalue problem: a method for updating the symmetric eigendecomposition, a divide-and-conquer algorithm, and methods for finding eigenvalues and eigenvectors of band matrices.

We next turn to the singular value decomposition. It is included in this chapter because the squares of the singular values of a real matrix X are the eigenvalues of the symmetric matrix $A = X^T X$ and the eigenvectors of A are the left singular vectors of X . Thus, in principle, the algorithms of the first section of this chapter can be used to solve the singular value problem. In practice, this approach has numerical difficulties (see §3.2), and it is better to solve the singular value problem on its own terms. Fortunately, the QR algorithm for symmetric matrices has an analogue for the singular value problem.

Finally, in a brief coda, we treat the symmetric positive definite generalized eigenvalue problem $Ax = \lambda Bx$, where A is symmetric and B is positive definite. This problem has appealing analogies to the ordinary symmetric eigenproblem, but it is mathematically and numerically less tractable.

In the ordinary eigenvalue problem, real and complex matrices are treated in es-

sentially different ways. For the symmetric eigenvalue problem, the singular value problem, and the definite generalized eigenvalue problem, the complex case differs from the real case only in the fine details. For this reason we will assume in this chapter that:

A and B are real symmetric matrices of order n.

1. THE QR ALGORITHM

In this section we will show how the QR algorithm can be adapted to find the eigenvalues and eigenvectors of a symmetric matrix. The general outline of the procedure is the same as for the nonsymmetric eigenvalue problem — reduce the matrix to a compact form and perform QR steps, accumulating the transformations if eigenvectors are desired. Because orthogonal similarities preserve symmetry, the process simplifies considerably. First, the Hessenberg form becomes a tridiagonal form [see (1.1)]. Second, the final Schur form becomes diagonal. Finally, the columns of the accumulated transformations are the eigenvectors of A . Thus the algorithm computes the spectral decomposition

$$V^T A V = \Lambda,$$

where V is an orthogonal matrix whose columns are eigenvectors of A and Λ is a diagonal matrix whose diagonal elements are the corresponding eigenvalues.

1.1. REDUCTION TO TRIDIAGONAL FORM

If a symmetric matrix A is reduced to Hessenberg form by orthogonal similarity transformations, all the elements below its first subdiagonal are zero and hence by symmetry so are all the elements above the first superdiagonal. Thus the matrix assumes a *tridiagonal form* illustrated by the following Wilkinson diagram:

$$\begin{pmatrix} X & X & 0 & 0 & 0 & 0 \\ X & X & X & 0 & 0 & 0 \\ 0 & X & X & X & 0 & 0 \\ 0 & 0 & X & X & X & 0 \\ 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & X & X \end{pmatrix}. \quad (1.1)$$

In this volume almost all tridiagonal matrices will turn out to be symmetric or Hermitian. Hence:

Unless otherwise stated a tridiagonal matrix is assumed to be symmetric or, if complex, Hermitian.

The purpose of this subsection is to describe the details of the reduction of a symmetric matrix to tridiagonal form. But first we must consider how symmetric matrices may be economically represented on a computer.

The storage of symmetric matrices

Since the elements a_{ij} and a_{ji} of a symmetric matrix are equal, a symmetric matrix is completely represented by the $n(n+1)/2$ elements on and above its diagonal. There are two ways of taking advantage of this fact to save storage.

First, one can store the matrix in, say, the upper half of a two-dimensional array of real floating-point numbers and code one's algorithms so that only those elements are altered. This approach has the advantage that it permits the use of array references like $A[i, j]$ while leaving the rest of the array free to hold other quantities. The disadvantage is that the rest of the array is seldom used in practice because its $n(n-1)/2$ elements are insufficient to hold another symmetric or triangular matrix of order n .

Second, one can put the diagonal and superdiagonal elements of a symmetric matrix into a one-dimensional array — a device known as *packed storage*. The two usual ways of doing this are to store them in column order and in row order. The following diagram illustrates a 4×4 symmetric matrix stored in an array a in column order.

$$\begin{aligned} a_{11} &\leftrightarrow a[1] & a_{12} &\leftrightarrow a[2] & a_{13} &\leftrightarrow a[4] & a_{14} &\leftrightarrow a[7] \\ a_{22} &\leftrightarrow a[3] & a_{23} &\leftrightarrow a[5] & a_{24} &\leftrightarrow a[8] \\ a_{33} &\leftrightarrow a[6] & a_{34} &\leftrightarrow a[9] \\ a_{44} &\leftrightarrow a[10] \end{aligned}$$

Packed storage offers two advantages. First, unlike the natural representation, it uses all the storage allocated to the matrix. Second, on nonoptimizing compilers it may run more efficiently, since the code references only a one-dimensional array. The price to be paid for these advantages is that the resulting code is hard to read (although practice helps).

Because our algorithms are easier to understand in the natural representation, we will assume in this volume that the symmetric matrix in question is contained in the upper part of a two-dimensional array.

Reduction to tridiagonal form

The algorithm for reduction to tridiagonal form is basically the same as the algorithm for reduction to Hessenberg form described beginning on page 83. We partition A in the form

$$A = \begin{pmatrix} \alpha_{11} & a_{21}^T \\ a_{21} & A_{22} \end{pmatrix}$$

and determine a Householder transformation H such that $Ha_{21} = \gamma \mathbf{e}_1$. Then

$$\text{diag}(1, H)A \text{ diag}(1, H) = \begin{pmatrix} \alpha_{11} & \gamma \mathbf{e}_1^T \\ \gamma \mathbf{e}_1 & HA_{22}H \end{pmatrix}.$$

This matrix has the Wilkinson diagram ($n = 6$)

$$\begin{pmatrix} X & X & 0 & 0 & 0 & 0 \\ X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \end{pmatrix},$$

from which it is seen that the transformation has introduced zeros into the appropriate parts of A . The reduction continues by recursively reducing $HA_{22}H$ to tridiagonal form.

The computation of $HA_{22}H$ can be simplified by taking advantage of symmetry. If we denote the vector defining H by r , then

$$\begin{aligned} HA_{22}H &= (I - rr^T)A_{22}(I - rr^T) \\ &= A - rr^T A_{22} - A_{22}rr^T + (r^T A_{22}r)rr^T. \end{aligned}$$

This formula can be simplified by setting

$$s = A_{22}r - \frac{r^T A_{22}r}{2}r, \quad (1.2)$$

so that

$$HA_{22}H = A - rs^T - sr^T.$$

Before we can code the algorithm, we must decide how to represent the final tridiagonal matrix. The usual convention is to store the diagonal of the matrix in a one-dimensional array, say d , and the off-diagonal elements in another one-dimensional array, say e . Thus the final tridiagonal matrix has the form

$$\begin{pmatrix} d_1 & e_1 & & & & \\ e_1 & d_2 & e_2 & & & \\ & e_2 & d_3 & e_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & e_{n-2} & d_{n-1} & e_{n-1} \\ & & & & e_{n-1} & d_n \end{pmatrix}. \quad (1.3)$$

Algorithm 1.1 implements the tridiagonalization described above. Here are some comments.

- The computation of the product of $A[k+1, k+1]$ and the Householder generating vector must be done at the scalar level, since in our storage scheme traversing a row

Given a real symmetric matrix A of order n , *TriReduce* reduces A to tridiagonal form. The diagonal elements of the tridiagonal matrix are stored in the array d ; the off-diagonal elements, in the array e . The transformations are accumulated in the array V .

```

1.  TriReduce( $A, d, e, V$ )
2.    for  $k = 1$  to  $n-2$ 
3.       $d(k) = A[k, k]$ 
4.      housegen( $A[k, k+1:n], r[k+1:n], e[k+1]$ )
5.       $V[k+1:n, k] = r[k+1:n]$ 
6.       $w = 0$ 
7.      for  $i = k+1$  to  $n$ 
8.         $s[i] = 0$ 
9.        for  $j = k+1$  to  $i-1$ 
10.        $s[i] = s[i] + A[j, i]*r[j]$ 
11.     end for  $j$ 
12.     for  $j = i$  to  $n$ 
13.        $s[i] = s[i] + A[i, j]*r[j]$ 
14.     end for  $j$ 
15.      $w = w + s[i]*r[i]$ 
16.   end for  $i$ 
17.    $s[k+1:n] = s[k+1:n] - (w/2)*r[k+1:n]$ 
18.   for  $j = k+1$  to  $n$ 
19.     for  $i = 1$  to  $j$ 
20.        $A[i, j] = A[i, j] - r[i]*s[j] - s[i]*r[j]$ 
21.     end for  $i$ 
22.   end for  $j$ 
23. end for  $k$ ;
24.  $d[n-1] = A[n-1, n-1]$ 
25.  $e[n-1] = A[n-1, n]$ 
26.  $d[n] = A[n, n]$ 
27.  $V[1:n-2, n-1:n] = 0; V[n-1:n, n-1:n] = I$ 
28. for  $k = n-2$  to  $1$  by  $-1$ 
29.    $r = V[k+1:n, k]$ 
30.    $v^T = r^T V[k+1:n, k+1:n]$ 
31.    $V[k+1:n, k+1:n] = V[k+1:n, k+1:n] - r*v^T$ 
32.    $V[:, k] = e_k$ 
33. end for  $k$ 
34. end TriReduce
```

Algorithm 1.1: Reduction to tridiagonal form

of A must be done by proceeding down the corresponding column to the diagonal and then across the row. To keep the indexing consistent we store the Householder vector in $r[k+1, n]$.

- As in Algorithm 2.2, Chapter 2, we economize by saving the generating vectors for the Householder transformations and then accumulating them in a separate loop.
- The algorithm is cheaper than the reduction of a general matrix to Hessenberg form. Specifically:

The operation count for Algorithm 1.1 is

$$\frac{2}{3}n^3 \text{ flam for the reduction of } A$$

$$\frac{2}{3}n^3 \text{ flam for the accumulation of } V$$

This should be compared with an operation count of $\frac{5}{3}n^3$ flam for the reduction of A in Algorithm 2.2, Chapter 2.

- In spite of the fact that the final tridiagonal form requires only two one-dimensional arrays to store, its computation requires the upper half of a two-dimensional array. This particular implementation uses the array containing the original matrix A , destroying A in the process.
- The algorithm is backward stable in the usual sense (see Definition 2.4, Chapter 2).
- The computation of s , and $r^T A_{22}r$ in (1.2) is done in statements 8–17. The computation of $A_{22}r$ requires two loops because only the upper part of the array A is referenced. This computation is neither column nor row oriented, and it is an interesting exercise to recast it so that it has uniform orientation.
- The first column of the matrix V is e_1 . This fact will prove important in deriving the implicit shift QR algorithm for symmetric tridiagonal matrices.

Making Hermitian tridiagonal matrices real

Algorithm 1.1 can easily be converted to compute the tridiagonal form of a complex Hermitian matrix. In this form the array d will necessarily contain real numbers; the array e , however, can contain complex numbers. This means that subsequent manipulations of the tridiagonal form will use complex rotations, which is expensive (see the comment on page 91). Fortunately, we can make the elements of e real by a sequence of diagonal similarity transformations.

The process is sufficiently well illustrated for the case $n = 3$. The tridiagonal matrix T has the form

$$T = \begin{pmatrix} d_1 & e_1 & 0 \\ \bar{e}_1 & d_2 & e_2 \\ 0 & \bar{e}_2 & d_3 \end{pmatrix}.$$

The first step is to make e_1 real. Let $D_1 = \text{diag}(1, \nu_1, 1)$. Then

$$D_1^{-1} T D_1 = \begin{pmatrix} d_1 & \nu_1 e_1 & 0 \\ \nu_1^{-1} \bar{e}_1 & d_2 & \nu_1^{-1} e_2 \\ 0 & \nu_1 \bar{e}_2 & d_3 \end{pmatrix}.$$

If $e_1 = 0$, it is real, and we do not need to do anything. In this case we may take $\nu_1 = 1$. Otherwise we take $\nu_1 = \bar{e}_1/|e_1|$, so that

$$D_1^{-1} T D_1 = \begin{pmatrix} d_1 & |e_1| & 0 \\ |e_1| & d_2 & \bar{\nu}_1 e_2 \\ 0 & \nu_1 \bar{e}_2 & d_3 \end{pmatrix} \equiv \begin{pmatrix} d_1 & |e_1| & 0 \\ |e_1| & d_2 & e'_2 \\ 0 & \bar{e}'_2 & d_3 \end{pmatrix}.$$

Similarly if we define $D_2 = \text{diag}(1, 1, \nu_2)$, where

$$\nu_2 = \begin{cases} 1 & \text{if } e'_2 = 0, \\ \bar{e}'_2/|e'_2| & \text{if } e'_2 \neq 0, \end{cases}$$

then

$$D_2^{-1} D_1^{-1} T D_1 D_2 = \begin{pmatrix} d_1 & |e_1| & 0 \\ |e_1| & d_2 & |e_2| \\ 0 & |e_2| & d_3 \end{pmatrix}.$$

Thus the original matrix has been reduced to real tridiagonal form.

Algorithm 1.2 sketches this scheme. It is incomplete in an important respect. The array e is necessarily an array of type complex, whereas at the end we require an array of real floating-point numbers. This problem can be overcome by adjusting the e_k (and the columns of V) as they are generated in the course of the reduction to tridiagonal form. The details are left as an exercise.

1.2. THE SYMMETRIC TRIDIAGONAL QR ALGORITHM

Since a tridiagonal matrix T is necessarily upper Hessenberg and the basic QR step preserves upper Hessenberg form, the result of performing a QR step on T is upper Hessenberg. Since the QR step also preserves symmetry, if T is symmetric and tridiagonal, the result of performing a QR step on T is symmetric and tridiagonal. Thus the QR algorithm preserves symmetric tridiagonal form. When this fact is taken into account the result is a sleek, speedy algorithm for computing the eigenvalues and optionally the eigenvectors of a symmetric tridiagonal matrix.

As above, we will assume that the tridiagonal matrix T in question is represented by an array d containing the elements of the diagonal of T and an array e containing the elements of the superdiagonal [see (1.3)].

This algorithm takes a Hermitian tridiagonal matrix represented by the arrays d and e and transforms it to a real tridiagonal matrix. The transformations are accumulated in a matrix V .

```

1.  for  $k = 1$  to  $n-1$ 
2.     $a = |e_k|$ 
3.    if ( $a \neq 0$ )
4.       $\nu = \bar{e}_k/a$ 
5.       $e_k = a$ 
6.      if ( $k \neq n-1$ )
7.         $e_{k+1} = \bar{\nu}*e_{k+1}$ 
8.      end if
9.       $V[1:n, k+1] = \nu*V[1:n, k+1]$ 
10.     end if
11.   end for  $k$ 
```

Algorithm 1.2: Hermitian tridiagonal to real tridiagonal form

The implicitly shifted QR step

The symmetric tridiagonal QR algorithm has explicitly and implicitly shifted forms, of which the latter is the most commonly implemented. In broad outline the tridiagonal QR step goes as follows.

- Given a shift κ determine a plane rotation P_{12} such that the $(2, 1)$ -element of $P_{12}^T(T - \kappa I)$ is zero.
- Determine an orthogonal matrix Q whose first column is a multiple of e_1 such that $\hat{T} = Q^T P_{12}^T A P_{12} Q$ is tridiagonal.

As with the implicit double shift, the reasoning following (3.6), Chapter 2, shows that \hat{T} is the matrix that would result from an explicit QR step with shift κ .

An implementation of this sketch goes as follows. Since P_{12} is a rotation in the $(1, 2)$ -plane, the matrix $P_{12}^T A P_{12}$ has the form

$$\begin{pmatrix} d & e & \hat{f} \\ e & d & e \\ \hat{f} & e & d & e \\ & e & d & e \\ & e & d \end{pmatrix}.$$

$$\begin{array}{c}
 \left(\begin{array}{ccccc} d & e & & & \\ e & d & e & & \end{array} \right) \xrightarrow{P_{12}} \left(\begin{array}{ccccc} d & e & \hat{f} & & \\ e & d & e & & \\ \hat{f} & e & d & e & \\ e & d & e & & \\ e & d & e & & \end{array} \right) \xrightarrow{P_{23}} \left(\begin{array}{ccccc} d & e & & & \\ e & d & e & \hat{f} & \\ e & d & e & e & \\ \hat{f} & e & d & e & \\ e & d & e & e & \end{array} \right) \\
 \xrightarrow{P_{34}} \left(\begin{array}{ccccc} d & e & & & \\ e & d & e & & \\ e & d & e & \hat{f} & \\ e & d & e & e & \\ \hat{f} & e & d & e & \end{array} \right) \xrightarrow{P_{45}} \left(\begin{array}{ccccc} d & e & & & \\ e & d & e & & \end{array} \right)
 \end{array}$$

Figure 1.1: Implicit tridiagonal QR step

We now choose a rotation P_{23} in the $(2, 3)$ -plane to annihilate f . On applying the similarity transformation, we obtain a matrix of the form

$$\left(\begin{array}{ccccc} d & e & & & \\ e & d & e & \hat{f} & \\ e & d & e & & \\ \hat{f} & e & d & e & \\ e & d & e & & \end{array} \right).$$

We see that f has been chased one row and column down the matrix. This process can be repeated until the element f drops off the end. Figure 1.1 illustrates the entire procedure.

In the implementation of this algorithm we must, as usual, take into account the fact that the problem may have deflated. Consequently, Algorithm 1.3 implements a QR step between rows $i1$ and $i2$ of T . The application of the rotations is tricky. As one enters the loop on i , the matrix has the form

$$\left(\begin{array}{cccc} \ddots & \ddots & \ddots & \\ & \hat{e}_{i-1} & g & e_i \\ & e_i & d_{i+1} & e_{i+1} \\ & \ddots & \ddots & \ddots \end{array} \right),$$

where a hat indicates a final value. The element g is an intermediate value of d_i after the rotation $P_{i-1,i}$ has been applied. The first two statements of the loop complete the

Given a symmetric tridiagonal matrix whose diagonal is contained in the array d and whose superdiagonal is contained in e , this algorithm performs one QR step between rows $i1$ and $i2$. The transformations are accumulated in V .

```

1.  triqr( $d, e, \kappa, i1, i2, V$ )
2.   $g = d[i1] - \kappa$ 
3.   $s = 1; c = 1; p = 0;$ 
4.  for  $i = i1$  to  $i2-1$ 
5.     $f = s*e[i]$ 
6.     $b = c*e[i]$ 
7.    rotgen( $g, f, c, s$ )
8.    if ( $i \neq i1$ )  $e[i-1] = g$ ; fi
9.     $u = d[i] - p$ 
10.    $v = (d[i+1] - u)*s + 2*c*b$ 
11.    $p = s*v$ 
12.    $d[i] = u+p$ 
13.    $g = c*v-b$ 
14.   rotapp( $c, s, V[:, i], V[:, i+1]$ )
15. end for  $i$ 
16.  $d[i2] = d[i2] - p$ 
17.  $e[i2-1] = g$ 
18. end triqr
```

Algorithm 1.3: The symmetric tridiagonal QR step

application of $P_{i-1,i}$ to bring the matrix in the form

$$\begin{pmatrix} \ddots & \ddots & \ddots & & \\ \hat{e}_{i-1} & g & b & f & \\ & b & d_{i+1} & e_{i+1} & \\ & f & \ddots & \ddots & \ddots \end{pmatrix}.$$

Finally the subsequent statements bring the matrix into the form

$$\begin{pmatrix} \ddots & \ddots & \ddots & & \\ \hat{e}_{i-1} & \hat{d}_i & \hat{e}_i & & \\ \hat{e}_i & g & e_{i+1} & & \\ & \ddots & \ddots & \ddots & \end{pmatrix},$$

which advances the loop by one step.

It is worth noting that the derivation of this sequence requires the application of the identity $c^2 + s^2 = 1$, so that in the presence of rounding error the statements do not produce the same results as the direct application of the plane rotations $P_{i,i+1}$. Such “simplifications” based on exact mathematical identities can turn a stable algorithm into an unstable one. Fortunately, this is not true of Algorithm 1.3, which is stable in the usual sense.

It is easy to see that if $m = i2 - i1$, then the transformation of T requires $6m$ fladd and $5m$ flmt plus m calls to *rotgen*. The accumulation of the transformations in V requires mn flrot. Thus the computation of eigenvectors is the dominant part of the calculation. If only eigenvalues are required, statement 14 can be eliminated with great saving in computations.

Choice of shift

Two shifts are commonly used with Algorithm 1.3. The first is the *Rayleigh quotient shift*, which is simply the diagonal element d_{i2} . The second is the *Wilkinson shift*, which is the eigenvalue of the matrix

$$\begin{pmatrix} d_{i2-1} & e_{i2} \\ e_{i2} & d_{i2} \end{pmatrix}$$

that is nearest d_{i2} . The Wilkinson shift is globally convergent and is generally the one that is used.

Local convergence

The local convergence analysis beginning on page 73 applies to the symmetric problem. Specifically, if, say, d_n is converging to a simple eigenvalue, then (2.14), Chapter 2, implies that the values of e_{n-1} converge cubically to zero, provided the Rayleigh quotient shift is used. In fact, because the multiple eigenvalues of a defective matrix

are nondefective the convergence is cubic even for multiple eigenvalues. The Wilkinson shift usually gives cubic convergence, although in theory the convergence can be quadratic. This possibility is so remote that it is not considered a reason to prefer the Rayleigh quotient shift over the Wilkinson shift.

Deflation

If e_i is zero, the tridiagonal matrix T deflates into two tridiagonal matrices, one ending with d_i and the other beginning with d_{i+1} . If e_i is negligible, we can set it to zero to deflate the problem. Thus we must consider what criterion we shall use to decide when a subdiagonal element of a tridiagonal matrix is effectively zero.

We have already treated this problem in §2.4, Chapter 2, where we proposed two possible solutions, one for balanced matrices and one for graded matrices. Here we will propose yet another solution adapted to tridiagonal matrices. We will proceed informally by deriving a criterion for deflating a graded matrix of order two.

Consider the matrix

$$T = \begin{pmatrix} d & e \\ e & \rho d \end{pmatrix}, \quad (1.4)$$

where $\rho < 1$ represents a grading ratio. When e is zero, the matrix has eigenvalues d and ρd . We will now see how e perturbs the eigenvalue ρd when e is small.

The characteristic equation for T is

$$\lambda^2 - (1 + \rho)d\lambda + \rho d^2 + g,$$

where $g = e^2$. Differentiating this equation implicitly with respect to g gives

$$2\lambda \dot{\lambda}_g - (1 + \rho)d\dot{\lambda}_g + 1 = 0,$$

where $\dot{\lambda}_g$ is the derivative of λ with respect to g . Since $\lambda = \rho d$, we have

$$\dot{\lambda}_g = -\frac{1}{(1 - \rho)d} \cong d^{-1},$$

the latter approximation holding when ρ is small. Hence when e is small, the perturbation in the eigenvalue ρd is approximately $-e^2/a$. If we wish the relative error in ρd to be less than the rounding unit, we must have

$$\frac{|e|^2}{|a||\rho a|} \lesssim \epsilon_M$$

or

$$|e| \leq \sqrt{|a||\rho a|}\sqrt{\epsilon_M}.$$

Note that the quantity $\sqrt{|a||\rho a|}$ is the geometric mean of the two diagonal elements of T . This suggests that we regard a subdiagonal of a graded tridiagonal matrix to be negligible if

$$|e_i| \leq \sqrt{|d_i d_{i+1}|} \sqrt{\epsilon_M}. \quad (1.5)$$

This criterion is effective for graded matrices, but it is too liberal for ordinary matrices. We therefore replace it with

$$|e_i| \leq \sqrt{|d_i d_{i+1}|} \epsilon_M. \quad (1.6)$$

Note that if d_i and d_{i+1} are of a size, then their geometric mean is about the same size, and the criterion reduces comparing e_i to the rounding unit scaled by the size of the local diagonals. On the other hand, if the matrix is graded as in (1.4), then the criterion is stricter than (1.5), which tends to preserve the smaller eigenvalues.

Because $\sqrt{|d_i d_{i+1}|} \leq \|T\|_F$, the Hoffman–Wielandt theorem (Theorem 3.10, Chapter 1) insures that the criterion (1.6) always preserves the absolute accuracy of the eigenvalues.

Graded matrices

It is important to remember that when one is using the tridiagonal QR algorithm to solve graded problems, the grading should be downward [see (2.36), Chapter 2]. If the grading is upward we can flip the matrix about its cross diagonal to reverse the grading. Alternatively, there is a variant of the QR algorithm, called the QL algorithm, which performs well on matrices that are graded upward. For more see the notes and references.

Variations

This brings to an end our discussion of the QR algorithm for symmetric matrices. But it by no means exhausts the subject. One has only to write down a naive version of the tridiagonal QR step and compare it with Algorithm 1.3 to realize that the formulas can be manipulated in many different ways. Not surprisingly, people have proposed many versions of the basic tridiagonal algorithm, each with their advantages—real or imagined. Of these variants perhaps the most important is the square root free PWK method, which is especially valuable when only the eigenvalues of a symmetric tridiagonal matrix are needed. See the notes and references.

1.3. NOTES AND REFERENCES

General references

The title of this chapter is the same as that of Beresford Parlett's masterful survey of the field [206], which is required reading for anyone interested in the subject. Wilkinson's *Algebraic Eigenvalue Problem* [300] contains much valuable material, as does the Handbook series of algorithms [305].

Reduction to tridiagonal form

Givens [83, 1954] was the first to use orthogonal transformations to reduce a symmetric matrix to tridiagonal form. He used plane rotations, which in consequence are often called Givens rotations, and the method is called Givens' method. He went on to find the eigenvalues of the tridiagonal matrix by Sturm sequences and the inverse power method (see §2.3). For an error analysis of Givens' method see [300, §5.26].

As mentioned in §2.6, Chapter 2, Householder [123] observed that a general matrix could be reduced to Hessenberg form by Householder transformations and by implication that a symmetric matrix could be reduced to tridiagonal form. The actual algorithm given here is due to Wilkinson [297]. For an error analysis of Householder's method see [300, §5.35].

The tridiagonal QR algorithm

Although Francis proposed his implicitly shifted QR algorithm for Hessenberg matrices, the extension to symmetric tridiagonal matrices is obvious. For a detailed treatment of the algorithm see [206, Ch. 8].

Parlett [206, Ch. 4] gives a proof of the convergence of the QR algorithm with Rayleigh quotient shift that he attributes to Kahan. The result is not quite global convergence, although the method performs well in practice. In 1968 Wilkinson [302] showed that the QR algorithm with Wilkinson shift (not his name for it) is globally convergent, and this has been the preferred shift ever since. However, Jiang and Zhang [135] have proposed a way of alternating between the Rayleigh quotient and Wilkinson shifts in such a way that the algorithm is globally and cubically convergent.

We have derived the criterion (1.6) for setting an element to zero informally by considering a 2×2 matrix. For positive definite matrices, the criterion can be rigorously justified [59, Theorem 2.3].

If a symmetric tridiagonal matrix is graded upward, the QR algorithm will generally fail to compute the smaller eigenvalues accurately. An alternative is the QL algorithm, which is based on the sequence defined by

1. $A_k = \kappa_k I = Q_k L_k$,
2. $A_{k+1} = L_k Q_k + \kappa_k I$.

The QL step moves northwest from the bottom of the matrix to the top, and it is the first off-diagonal element that converges rapidly to zero. If the tridiagonal matrix was obtained by Householder tridiagonalization, that algorithm must also proceed from the bottom up, rather than top down as in Algorithm 1.1.

Perhaps the most complete set of options is given by the LAPACK codes `xSYTRD` and `xSTEQR`. The former reduces a symmetric matrix to tridiagonal form and can be coaxed into performing the reduction either top down or bottom up. The latter automatically chooses to use either the QR or QL algorithm based on the relative sizes of the diagonals at the beginning and ends of the current blocks.

For a brief summary of the various implementations of the symmetric tridiagonal

QR algorithm see [206, §§8.14–15], which also contains a complete exposition of the PWK (Pal–Walker–Kahan) algorithm. For an implementation see the LAPACK routine `xSTERF`.

2. A CLUTCH OF ALGORITHMS

The QR algorithm is not the only way to compute the eigensystem of a symmetric matrix. In this section we will consider some important alternative algorithms. We will begin this section with the problem of computing the eigensystem of a diagonal matrix plus a rank-one modification. This algorithm can in turn be used to implement a divide-and-conquer algorithm for the symmetric tridiagonal problem, which is treated in §2.2. Finally, in §2.3 we will show how to compute selected eigenvalues and eigenvectors of symmetric band matrices.

2.1. RANK-ONE UPDATING

In this subsection we will be concerned with the following problem. Let the symmetric matrix A have the spectral decomposition

$$A = V\Lambda V^T \quad (2.1)$$

[see (1.10), Chapter 1]. Let

$$\hat{A} = A + \sigma xx^T, \quad (2.2)$$

where $\sigma = \pm 1$. Find the spectral decomposition $\hat{A} = \hat{V}\hat{\Lambda}\hat{V}^T$ of \hat{A} .

One way of solving this problem is to form \hat{A} and use the QR algorithm to compute its eigensystem. However, it might be hoped that we can use the spectral decomposition (2.1), which we already know, to simplify the computation — a process called *updating*. With some decompositions — e.g., the Cholesky decomposition — updating can result in a striking reduction in computation, typically from $O(n^3)$ to $O(n^2)$. Unfortunately, updating the spectral decomposition requires $O(n^3)$ operations. Nonetheless, the updating algorithm is less expensive than computing the decomposition from scratch, and the central part of the computation — the diagonal updating problem — is useful in its own right.

Algorithm 2.1 outlines the basic steps of the updating algorithm. We will treat these steps each in turn.

Reduction to standard form

We first turn to the problem of reducing $\hat{A} = A + \sigma xx^T$ to the standard form $D + zz^T$. We proceed in stages. At each stage we underline the currently reduced matrix

First, since $\sigma = \pm 1$, we may write the updating problem in the form

$$\hat{A} = \sigma(\sigma A + \sigma^2 xx^T) = \sigma(\underline{\sigma A} + \underline{x}x^T).$$

Let $A = V\Lambda V^T$ be the spectral decomposition of the symmetric matrix A . This algorithm sketches how to compute the spectral decomposition of $\hat{A} = A + \sigma xx^T$, where $\sigma = \pm 1$.

1. Reduce the problem to that of updating the eigensystem of $D + zz^T$, where D is diagonal with nondecreasing diagonal elements, and z is a vector with nonnegative components.
2. Deflate the problem to get rid of all very small components of z and to make the diagonal elements of D reasonably distinct.
3. Find all the remaining eigenvalues of $D + zz^T$. This is done by solving a rational equation whose roots are the required eigenvalues.
4. Use the eigenvalues to compute the eigenvectors of $D + zz^T$.
5. Transform the eigensystem of $D + zz^T$ back to the original eigensystem of \hat{A} .

Algorithm 2.1: Rank-one update of the spectral decomposition

This makes the sign of xx^T positive.

Second, from the spectral decomposition (2.1), we have

$$\hat{A} = \sigma(\sigma V\Lambda V^T + xx^H) = \sigma V(\sigma\Lambda + V^T xxV)V^T \equiv \sigma V(\underline{\sigma\Lambda + yy^T})V^T.$$

The matrix to be updated is now $\sigma\Lambda$, which is diagonal.

Third, let S be a diagonal matrix whose elements are ± 1 , such that Sy is nonnegative. Then

$$\hat{A} = \sigma(VS)(\underline{\sigma\Lambda + S^T yy^T S})(VS)^T.$$

This makes the elements of the vector defining the update nonnegative.

Fourth, let P be a permutation such that the diagonals of $\sigma P^T \Lambda P$ appear in non-decreasing order. Then

$$\hat{A} = \sigma(VSP)(\underline{\sigma P^T S \Lambda S P + P^T S y y^T S P})(VSP)^T.$$

If we now set

$$D = \sigma P^T S \Lambda S P = \text{diag}(d_1, \dots, d_n) \quad \text{and} \quad z = P^T S y,$$

then

$$\hat{A} = \sigma(VSP)(\underline{D + zz^T})(VSP)^T.$$

Thus if we can compute the spectral decomposition

$$D + zz^T = Q\check{\Lambda}Q^T,$$

then with $\hat{\Lambda} = \sigma \check{\Lambda}$ the decomposition

$$\hat{A} = (VSPQ)\hat{\Lambda}(VSPQ)^T \quad (2.3)$$

is the spectral decomposition of A .

Here are some comments on the reduction.

- The way $\sigma = \pm 1$ keeps popping in and out of the formulas may be confusing. Algorithmically, what the equations say is that if $\sigma = -1$, update $-\hat{A}$ and then change the sign back at the end.
- To complete the updating, we must compute $VSPQ$ (this is essentially step five in Algorithm 2.1). The computation of VS is simply a column scaling. The computation of $(VS)P$ is a permutation of the columns of VS , about which more below. The computation of $(VSP)Q$ is a matrix-matrix multiplication and takes about n^3 flam. This has the consequence that our updating algorithm does not gain in order over the naive scheme of computing the decomposition of \hat{A} from scratch. But still it is faster.
- Unless the order n of the problem is small, care should be taken to choose an efficient algorithm to sort the diagonals of D in order to avoid a lengthy series of column interchanges in the matrix VS .
- We will have occasions to exchange diagonals of D later in the algorithm. Any permutation of the diagonals must be reflected in a corresponding permutation of the columns of VS and the components of z . When we speak of such permutations, we will tacitly assume that VS and z have also been adjusted.
- In an actual implementation it may be better not to permute the columns of VS , the diagonals of D , and the components of z until the very end. Instead one might construct a data structure that keeps track of where these quantities should be and reference them through that structure.

Deflation: Small components of z

If the i th component of z is zero, so that z has the form

$$z^T = (z_a^T \ 0 \ z_b^T),$$

then $D + zz^T$ can be partitioned in the form

$$D + zz^T = \begin{pmatrix} D_a + z_a z_a^T & 0 & z_a z_b^T \\ 0 & d_i & 0 \\ z_b z_a^T & 0 & D_b + z_b z_b^T \end{pmatrix}. \quad (2.4)$$

In this case the element d_i is an eigenvalue of the updated matrix. By a permutation we can move this eigenvalue to the end of the matrix and bring it back at the end of the update.

In practice, we will seldom meet with components of z that are exactly zero. Instead we will find that some component z_i is small, and we must decide when we can regard it as zero. Now in this case we can write

$$D + zz^T = \begin{pmatrix} D_a + z_a z_a^T & 0 & z_a z_b^T \\ 0 & d_i & 0 \\ z_b z_a^T & 0 & D_b + z_b z_b^T \end{pmatrix} + \begin{pmatrix} 0 & z_i z_a & 0 \\ z_i z_1 & z_i^2 & z_i z_b \\ 0 & z_i z_b & 0 \end{pmatrix} \equiv B + E.$$

Thus the effect of replacing z_i by zero is to replace $D + zz^T$ by $D + zz^T - E$. Thus a reasonable criterion is to require z_i to be so small that

$$\|E\|_2 \leq \gamma \|D + zz^H\|_2 \epsilon_M,$$

where ϵ_M is the rounding unit, and γ is some modest constant (more about that later). By Theorem 3.8, Chapter 1, if this criterion is satisfied, setting z_i to zero will move each eigenvalue by no more than $\|D + zz^H\|_2 \epsilon_M$. Unless the original matrix has special structure, this is the kind of perturbation we should expect from rounding the elements of A .

Now by computing the Frobenius norm of E , we get the bound

$$\|E\|_2 \leq \sqrt{2}|z_i|\|z\|_2.$$

Moreover we can estimate

$$\|D + zz^T\|_2 \cong \max |d_j| + \|z\|_2^2.$$

Hence our criterion becomes

$$|z_i|\|z\|_2 \leq \gamma(\max |d_j| + \|z\|_2^2) \epsilon_M. \quad (2.5)$$

Here we have absorbed the constant $\sqrt{2}$ into γ .

Deflation: Nearly equal d_i

It turns out that if two diagonals of D are sufficiently near one another, we can use a plane rotation to zero out one of the corresponding components of z , thereby deflating the problem. The process is sufficiently well illustrated by the 2×2 problem

$$\begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} (z_1 \ z_2).$$

Let c and s be the cosine and sine of a plane rotation that annihilates the component z_2 of z :

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \tilde{z}_1 \\ 0 \end{pmatrix},$$

where $\tilde{z}_1 = \sqrt{z_1^2 + z_2^2}$. Then

$$\begin{aligned} & \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \left[\begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} (z_1 \ z_2) \right] \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \\ &= \begin{pmatrix} d_1 c^2 + d_2 s^2 & (d_2 - d_1)cs \\ (d_2 - d_1)cs & d_1 s^2 + d_2 c^2 \end{pmatrix} + \begin{pmatrix} \tilde{z}_1 \\ 0 \end{pmatrix} (\hat{z}_1 \ 0). \end{aligned}$$

Consequently, if $(d_2 - d_1)cs$ is sufficiently small we can ignore it, and the problem deflates. As above, we deem this quantity sufficiently small if

$$(d_2 - d_1)cs \leq \gamma(\max |d_j| + \|z\|_2^2)\epsilon_M. \quad (2.6)$$

In the general case, because the diagonal elements of d are nondecreasing it is only necessary to test adjacent diagonal elements. Of course, once a diagonal has been deflated, the diagonal that remains must be tested again against its new neighbor.

When to deflate

There is an important tradeoff in our criteria for deflation. If it is too stringent, we will seldom deflate and miss the substantial benefits of deflation. On the other hand, if the criterion is too liberal, the accuracy of our final results will deteriorate.

There is yet another tradeoff. The calculations of the eigenvalues and eigenvectors to be described later require that the z_i be nonzero and that the d_i be strictly increasing. Thus the deflation procedure can be seen as a preprocessing step to insure that the computation as a whole runs smoothly. This favors a liberal criterion over a conservative one. Saving accuracy at one stage is a poor bargain if it is lost at another.

Since γ controls the stringency of the criterion, the above discussion suggests that it should have a reasonable large value. A strict rounding-error analysis suggests that γ ought to grow with n . However, this is too liberal because the rounding-error analysis is necessarily conservative. The conventional wisdom is that γ should be a modest constant, say 10.

The secular equation

We turn now to the computation of the eigenvalues of $D + zz^H$. We begin with a theorem that shows where the eigenvalues lie and gives an equation they must satisfy.

Theorem 2.1. *If $d_1 < d_2 < \dots < d_n$ and the components of z are nonzero, the eigenvalues of $D + zz^T$ satisfy the SECULAR EQUATION*

$$f(\lambda) \equiv 1 - \sum_{i=1}^n \frac{z_i^2}{\lambda - d_i}. \quad (2.7)$$

Moreover, the solutions $\hat{\lambda}_1 < \dots < \hat{\lambda}_n$ of (2.7) satisfy

$$d_1 < \hat{\lambda}_1 < d_2 < \hat{\lambda}_2 < \dots < d_n < \hat{\lambda}_n. \quad (2.8)$$

Proof. To derive the secular equation we will employ the useful fact that

$$\det(I - uv^T) = 1 - v^T u. \quad (2.9)$$

This can be proved by observing that any vector orthogonal to v is an eigenvector of $I - uv^T$ corresponding to one, while the vector v is an eigenvector corresponding to $1 - v^T u$. The result now follows from the fact that the determinant of a matrix is the product of the eigenvalues of the matrix.

The characteristic polynomial of $D + zz^T$ is

$$\begin{aligned} p(\lambda) &= \det(\lambda I - D - zz^T) \\ &= \det(\lambda I - D) \det[I - (\lambda I - D)^{-1}zz^T] \\ &= \prod_i (\lambda - d_i) \left[1 - \sum_i \frac{z_i^2}{\lambda - d_i} \right], \end{aligned} \quad (2.10)$$

the last equality following from (2.9).

Thus the characteristic polynomial is the product of $\prod_i (\lambda - d_i)$ and $f(\lambda)$ defined by (2.7). Let us consider the zeros of the latter. As λ approaches d_1 from above, $f(\lambda)$ must become negative. As λ approaches d_2 from below, $f(\lambda)$ must become positive. Hence f has a zero $\hat{\lambda}_1$ in (d_1, d_2) . In the same way we can show that f has zeros $\hat{\lambda}_i \in (d_i, d_{i+1})$ ($i = 1, \dots, n-1$). Finally, as λ approach d_n from above, $f(\lambda)$ becomes negative, while as $\lambda \rightarrow \infty$, $f(\lambda) \rightarrow 1$. Hence f has a zero $\hat{\lambda}_n$ satisfying $\hat{\lambda}_n > d_n$.

The proof is completed by observing that the n zeros $\hat{\lambda}_i$ of f are also zeros of the characteristic polynomial p and hence are the eigenvalues of $D + zz^T$. ■

Solving the secular equation

Theorem 2.1 shows that we can find the updated eigenvalues $\hat{\lambda}_i$ of \hat{A} (up to the sign σ) by solving the secular equation (2.7). Moreover, the interlacing inequality (2.8) shows that each root $\hat{\lambda}_i$ is simple and located in (d_i, d_{i+1}) [or in (d_n, ∞) for $\hat{\lambda}_n$]. Thus in principle we could use an off-the-shelf root finder to compute the $\hat{\lambda}_i$.

However, to determine the eigenvectors the eigenvalues must be calculated to high accuracy. Thus our root-finding algorithm must be specifically designed to work with the secular equation. The detailed description of such an algorithm is beyond the scope of this book, and here we will merely sketch the important features. But first a word on computer arithmetic.

Throughout these volumes we assume that floating-point computations are done in standard floating-point arithmetic. Specifically, we assume that, excluding overflows and underflows, each floating-point operation returns the correctly rounded value of the exact operation. This assumption is critical for the algorithms that follow. For more see the notes and references.

For definiteness we will consider the problem of computing $\hat{\lambda}_k$, where $k < n$. The first step is to determine whether λ_k is nearer d_k or d_{k+1} . This can be done by

evaluating $f[(d_k + d_{k+1})/2]$. If the value is negative, $\hat{\lambda}_k$ is nearer d_i ; if it is positive, $\hat{\lambda}_k$ is nearer d_{i+1} . For definiteness we will suppose that $\hat{\lambda}_k$ is nearer d_k .

The second step is to shift the secular equation so that the origin is d_k . Specifically, let

$$\lambda = d_k + \tau$$

and define $g(\tau) = f(d_k + \tau)$. Then

$$\begin{aligned} g(\tau) &= 1 - \sum_i \frac{z_i^2}{d_k + \tau - d_i} \\ &= 1 - \sum_{i < k} \frac{z_i^2}{(d_k - d_i) + \tau} - \sum_{i \geq k} \frac{z_i^2}{(d_k - d_i) + \tau} \\ &\equiv 1 - r(\tau) - s(\tau). \end{aligned}$$

This shift has the advantage that we are now solving for the correction τ_k to d_k that gives $\hat{\lambda}_k$. Moreover, the functions $r(\tau)$ and $s(\tau)$ can be evaluated to high relative accuracy whenever τ is between 0 and $(d_{k+1} - d_k)/2$.

In principle any good zero-finding technique applied to g will suffice to find the correction τ_k . However, because $g(\tau)$ has poles at zero and $d_{k+1} - d_k$, the most effective schemes are based on approximating $g(\tau)$ by a rational function $\tilde{g}(\tau)$ and finding the root of \tilde{g} as the next approximation. For more, see the notes and references.

Finally, one must decide when the iteration for τ_k has converged. Because of the simplicity of the function $g(\tau)$, as it is calculated one can also calculate a bound on the error in its computed value. Our stopping criterion is:

$$\text{Stop when the computed value of } g(\tau) \text{ at the current iterate is less than its error bound.} \quad (2.11)$$

It can be shown that this criterion returns eigenvalues that are sufficiently accurate to be used in the computation of the eigenvectors, to which we now turn.

Computing eigenvectors

Let $\hat{\lambda}_j$ be an eigenvalue of $D + zz^T$. Then the corresponding eigenvector q must satisfy the equation $(D + zz^T)q_j = \hat{\lambda}_j q_j$ or equivalently

$$(\hat{\lambda}_j I - D)q_j = (z^T q_j)z.$$

Thus q_j must lie along the direction of $(\hat{\lambda}_j I - D)^{-1}z$, and hence the normalized eigenvector is given by

$$q_j = \frac{(\hat{\lambda}_j I - D)^{-1}z}{\|(\hat{\lambda}_j I - D)^{-1}z\|_2}.$$

Unfortunately, we do not know the eigenvalues of $D + zz^T$. Instead we have approximations $\tilde{\lambda}_j$ obtained by solving the secular equation. Now the i th component of q_j is given by

$$\varsigma_{ij} = \eta_j \frac{z_i}{(\tilde{\lambda}_j - d_i)}, \quad (2.12)$$

where η_j is the normalizing constant. If $\tilde{\lambda}_j$ is very near d_i , as it will be when d_i and d_{i+1} are very near each other, replacing $\tilde{\lambda}_j$ by $\tilde{\lambda}_j$ in (2.12) will make a large perturbation in q_{ij} , unless $\tilde{\lambda}_j$ is a very accurate approximation to $\tilde{\lambda}_j$. In particular, eigenvectors computed in this way may be inaccurate and, worse yet, be far from orthogonal to one another.

One cure for this problem is to compute the eigenvalues $\tilde{\lambda}_j$ to high precision (it can be shown that double precision is sufficient). However, there is an alternative that does not require extended precision. It is based on the following theorem.

Theorem 2.2. *Let the quantities $\tilde{\lambda}_i$ satisfy the interlacing conditions*

$$d_1 < \tilde{\lambda}_1 < d_2 < \tilde{\lambda}_2 < \dots < d_n < \tilde{\lambda}_n. \quad (2.13)$$

Then the quantities

$$\tilde{z}_i = \text{sign } \tilde{z}_i \sqrt{\frac{\prod_j (d_i - \tilde{\lambda}_j)}{\prod_{j \neq i} (d_i - d_j)}} \quad (2.14)$$

are well defined. If we set $\tilde{z} = (\tilde{z}_1 \ \dots \ \tilde{z}_n)^T$, then the eigenvalues of $D - \tilde{z}\tilde{z}^T$ are $\tilde{\lambda}_1, \dots, \tilde{\lambda}_n$.

Proof. We will first derive the formula (2.14) under the assumption that the required \tilde{z}_i exist, and then show that these values have the required property.

We compute the characteristic polynomial of $D + \tilde{z}\tilde{z}^T$ in two ways:

$$\begin{aligned} \det(\lambda I - D - \tilde{z}\tilde{z}^T) &= \prod_j (\lambda - \tilde{\lambda}_j) \\ &= \prod_j (\lambda - d_j) \left[1 - \sum_j \frac{\tilde{z}_j^2}{\lambda - d_j} \right] \end{aligned} \quad (2.15)$$

[see (2.10)]. Setting $\lambda = d_i$, and solving for \tilde{z}_i^2 , we find that

$$\tilde{z}_i^2 = \frac{\prod_j (d_i - \tilde{\lambda}_j)}{\prod_{j \neq i} (d_i - d_j)}.$$

The interlacing inequalities imply that the right-hand side of this equation is positive. Hence the \tilde{z}_i are well defined and given by (2.14).

To show that the eigenvalues of $D + \tilde{z}\tilde{z}^T$ are the $\tilde{\lambda}_j$, note that by the way the \tilde{z}_i were defined the two expressions for the characteristic polynomial of $D + \tilde{z}\tilde{z}^T$ in (2.15) are equal at the n distinct points d_j . Since both expressions are monic polynomials of degree n , they must be identical. In particular if we evaluate both expressions at the $\tilde{\lambda}_j$, we discover that they are zeros of the second expression — i.e., the $\tilde{\lambda}_j$ satisfy the secular equation for $D + \tilde{z}\tilde{z}^T$. ■

Equation (2.14) does not quite specify \tilde{z}_i , since the right-hand side contains the factor $\text{sign}(\tilde{z}_i)$. As we shall see in a moment, in our application it is sufficient to replace $\text{sign}(\tilde{z}_i)$ with $\text{sign}(z_i)$.

To apply the theorem, suppose that we compute the \tilde{z}_i from (2.14), with the appropriate signs, and then use them to compute the components of the eigenvectors from (2.12). Now in standard floating-point arithmetic, the \tilde{z}_i can be computed to high relative accuracy. Moreover, in the expression

$$\tilde{z}_{ij} = \eta_j \frac{\tilde{z}_i}{(\tilde{\lambda}_j - d_i)},$$

the $\tilde{\lambda}_j$ are the *exact* eigenvalues of $D + \tilde{z}\tilde{z}^T$. Hence the components \tilde{z}_{ij} of the computed eigenvectors \tilde{q}_j will be to high relative accuracy the components of the exact eigenvectors of $D + \tilde{z}\tilde{z}^T$. In particular, the matrix Q of these eigenvectors will be orthogonal to working accuracy.

There remains the question of the stability of this algorithm; after all, we have not found the eigenvectors of $D + zz^T$ but those of $D + \tilde{z}\tilde{z}^T$. However, it can be shown that if we use the stopping criterion (2.11) in finding the approximate eigenvalues $\tilde{\lambda}_i$, then

$$|\tilde{z}_i - z_i| \leq \mu \|z_i\|_2 \epsilon_M, \quad (2.16)$$

where μ is a slowly growing function of n . Thus the computation of the eigenvectors of $D + zz^T$ is backward stable.

The inequality (2.16) also justifies our replacing $\text{sign}(\tilde{z}_i)$ by $\text{sign}(z_i)$. For if \tilde{z}_i and z_i have opposite signs, they must be within a small multiple of the rounding unit, and hence insignificant.

We summarize the computation of the eigenvectors in Algorithm 2.2. Here are two comments on this program.

- We have already noted that the algorithm is backward stable. However, this does not imply that the computed eigenvectors are accurate. They will have small residuals [see (2.35), Chapter 2], and, as we have pointed out, they will be orthogonal almost to working accuracy. But by Theorem 3.13, Chapter 1, the vectors corresponding to poorly separated eigenvalues will be inaccurate.
- The computation of the z_i requires approximately $2n^2$ flam and n^2 fdiv. The computation of the vectors requires approximately n^2 fladd and n^2 fdiv for the computation of the components of Q and n^2 flam and n^2 fdiv for the normalization (the divisions in the normalization can be replaced by multiplications). Thus the computation

Given sufficiently accurate approximations $\tilde{\lambda}_j$ [see (2.11)] to the eigenvalues of $D + zz^T$ that satisfy the interlacing inequalities (2.13), this algorithm returns approximations to the eigenvectors of $D + zz^T$ in the array Q .

Compute the modified update vector

1. **for** $i = 1$ **to** n
2. $\tilde{z}_i = \tilde{\lambda}_n - d_i$
3. **for** $j = 1$ **to** $i-1$
4. $\tilde{z}_i = \tilde{z}_i * (\tilde{\lambda}_j - d_i) / (d_j - d_i)$
5. **end for** j
6. **for** $j = i$ **to** $n-1$
7. $\tilde{z}_i = \tilde{z}_i * (\tilde{\lambda}_j - d_i) / (d_{j+1} - d_i)$
8. **end for** j
9. $\tilde{z}_i = \text{sqrt}(\tilde{z}_i) * \text{sign}(z_i)$
10. **end**

Compute the eigenvectors

11. **for** $j = 1$ **to** n
12. **for** $i = 1$ **to** n
13. $Q[i, j] = \tilde{z}_i / (\tilde{\lambda}_j - d_i)$
14. **end for** i
15. $Q[:, j] = Q[:, j] / \|Q[:, j]\|_2$
16. **end for** j

Algorithm 2.2: Eigenvectors of a rank-one update



of the eigenvectors is an $O(n^2)$ process. However, it must be remembered that the diagonal updating is only a part of the larger Algorithm 2.1. Once the cost of computing $VSPQ$ in (2.3) is taken into account, the count rises to $O(n^3)$.

Concluding comments

Our sketch of the algorithm for spectral updating is far from a working program. The data structures to handle permutations and deflation are nontrivial, and the routine to solve the secular equation is intricate. Thus the code for any implementation will be long and complicated compared with the code for forming \hat{A} directly and applying the QR algorithm. Is it worth it?

The answer is yes, at least for sufficiently large matrices. We have seen that the $O(n^3)$ overhead in the QR algorithm is the accumulation of the transformation, particularly the expensive accumulation of the plane rotations generated while solving the tridiagonal system. The updating algorithm has only one $O(n^3)$ overhead: a single matrix-matrix multiplication to update the eigenvectors V . Thus for n large enough the updating algorithm must be faster.

However, there is a second aspect. The deflation also saves time, and in some applications there are many deflations. We will return to this point in the next subsection.

The algorithm is not without its drawbacks. Extra storage is required for the eigenvectors of the diagonalized problem $D + zz^T$. The algorithm does not work well for graded matrices. In particular, the deflation criteria—e.g., (2.6)—are couched in terms of the norms of D and z , which allows small but meaningful elements in a graded matrix to be declared negligible.

2.2. A DIVIDE-AND-CONQUER ALGORITHM

In this subsection we will present a divide-and-conquer algorithm for solving the symmetric tridiagonal eigenvalue problem. We begin with some generalities about the divide-and-conquer approach.

Generalities

The notion of a divide-and-conquer algorithm can be described abstractly as follows. Let $P(n)$ denote a computational problem of size n , and suppose that we can divide $P(n)$ into two problems $P_1(\frac{n}{2})$ and $P_2(\frac{n}{2})$ of size $\frac{n}{2}$ in such a way that the solution of P can be recovered from those of P_1 and P_2 at a cost of $C(n)$. Now if P_1 and P_2 are problems of the same type as P , we can repeat this dividing procedure on P_1 and P_2 , then on the subproblems of P_1 and P_2 , and so on. This recursion will stop after the size of the subproblems becomes so small that their solutions are trivial — after no more than about $\log_2 n$ steps.

We can estimate the cost of a divide-and-conquer algorithm by counting the costs at each level of the recurrence. For convenience, assume that $n = 2^m$. At the top level of the recurrence we solve one problem at a cost of $C(n)$. At the second level, we solve two problems each at a cost of $C(\frac{n}{2})$ for a total cost of $2C(\frac{n}{2})$. At the third

level, we solve four problems at a cost of $C(\frac{n}{4})$ for a total cost of $4C(\frac{n}{4})$ — and so on. Summing up these costs, we get for the total cost of the divide-and-conquer algorithm

$$C_{\text{total}} = C(n) + 2^1 C(2^{-1}n) + 2^2 C(2^{-2}n) + \cdots + 2^m C(2^{-m}n). \quad (2.17)$$

Of course we must know the function $C(n)$ to evaluate this expression. We will return to it later, after we have derived our algorithm.

Derivation of the algorithm

Let T be a symmetric tridiagonal matrix of order n , and let T be partitioned in the form

$$T = \left(\begin{array}{cc|c} d_1 & e_1 & & \\ e_1 & d_2 & e_2 & \\ \ddots & \ddots & \ddots & \\ & e_{s-1} & d_s & e_s \\ \hline & e_s & d_{s+1} & e_{s+1} \\ & & \ddots & \ddots \\ & & e_{n-2} & d_{n-1} & e_{n-1} \\ & & e_{n-1} & d_n & \end{array} \right). \quad (2.18)$$

(Here s stands for *splitting index*.) We can also write this partition in the form

$$T = \begin{pmatrix} T_1 & e_s e_s e_1^T \\ e_s e_1 e_s^T & T_2 \end{pmatrix}.$$

Now let

$$\hat{T}_1 = T_1 - e_s e_s e_s^T \quad \text{and} \quad \hat{T}_2 = T_2 - e_s e_1 e_1^T,$$

so that \hat{T}_1 is T_1 with the element d_s replaced by $d_s - e_s$ and \hat{T}_2 is T_2 with the element d_{s+1} replaced by $d_{s+1} - e_s$. Then we can write

$$T = \begin{pmatrix} \hat{T}_1 & 0 \\ 0 & \hat{T}_2 \end{pmatrix} + e_s \begin{pmatrix} e_s \\ e_1 \end{pmatrix} (e_s^T \ e_1^T).$$

Thus we have expressed T as the sum of a block diagonal matrix whose blocks are tridiagonal and a matrix of rank one. The idea behind our divide-and-conquer algorithm is to compute the eigensystems of \hat{T}_1 and \hat{T}_2 (which can be done recursively) and then combine them to give the eigensystem of T .

To show how to combine the eigensystems, let

$$\hat{T}_1 = \hat{V}_1 \hat{\Lambda}_1 \hat{V}_1^T \quad \text{and} \quad \hat{T}_2 = \hat{V}_2 \hat{\Lambda}_2 \hat{V}_2^T$$

be spectral decompositions of \hat{T}_1 and \hat{T}_2 . Then

$$\begin{aligned} T &= \begin{pmatrix} \hat{V}_1 & 0 \\ 0 & \hat{V}_2 \end{pmatrix} \left[\begin{pmatrix} \hat{\Lambda}_1 & 0 \\ 0 & \hat{\Lambda}_2 \end{pmatrix} + e_s \begin{pmatrix} \hat{V}_1^T \mathbf{e}_s \\ \hat{V}_2^T \mathbf{e}_1 \end{pmatrix} (\mathbf{e}_s^T \hat{V}_1 \quad \mathbf{e}_1^T \hat{V}_2) \right] \begin{pmatrix} \hat{V}_1^T & 0 \\ 0 & \hat{V}_2^T \end{pmatrix} \\ &= \begin{pmatrix} \hat{V}_1 & 0 \\ 0 & \hat{V}_2 \end{pmatrix} \left[\begin{pmatrix} \hat{\Lambda}_1 & 0 \\ 0 & \hat{\Lambda}_2 \end{pmatrix} + e_s \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} (z_1 \ z_2) \right] \begin{pmatrix} \hat{V}_1^T & 0 \\ 0 & \hat{V}_2^T \end{pmatrix} \\ &\equiv \hat{V}(D + e_s z z^T) \hat{V}^T. \end{aligned}$$

The matrix $D + z z^T$ is a rank-one modification of a diagonal matrix, whose spectral decomposition

$$D + z z^T = Q \Lambda Q^T$$

can be computed by the algorithm of the last section. It follows that

$$T = (\hat{V}Q)\Lambda(\hat{V}Q)^T$$

is the spectral decomposition of T .

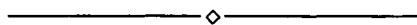
We are now in a position to apply the divide-and-conquer strategy to compute the eigensystem of a symmetric tridiagonal matrix. Algorithm 2.3 implements this scheme. Here are some comments.

- For ease of exposition we cast the algorithm in terms of full matrices. In practice, we would represent T in some compact form — e.g., as in (2.18).
- We have already observed that the method for updating a spectral decomposition has a great deal of overhead, so that for small matrices the QR approach is more competitive. Consequently, when the submatrices in the recursion fall below a certain size, specified here by the parameter *mindac*, we abandon the divide-and-conquer approach and compute the decompositions directly via the QR algorithm. Ideally, the value of *mindac* should be the break-even point where the two algorithms require about the same time to execute. Actually, for large matrices the ill consequences of using the wrong value are not very great, as we shall see. In LAPACK it is set at 25.
- The recursive approach taken here is natural for this algorithm. However, a direct, bottom-up approach is possible. For example, one might initially partition the matrix into blocks of size just less than *mindac*. Then one can work upward combining them in pairs. The code is not pretty.
- Because the algorithm *diagupdate* is backward stable, it can be shown that *cuppen* is also stable.
- We have noted that the updating algorithm *diagupdate* may not work well with graded matrices. Algorithm 2.3 inherits this drawback.
- In statements 16 and 17 we have taken advantage of the fact that V is block diagonal in forming VQ . This reduces the work from n^3 flam to $\frac{1}{2}n^3$ flam.

Given a symmetric tridiagonal matrix T , *Cuppen* computes its spectral decomposition $V\Lambda V^T$. It uses the routine *diagupdate*(D, σ, z, Λ, Q) that returns the spectral decomposition $D + zz^T = Q\Lambda Q^T$. If the size of the matrix is less than the parameter *mindac* (min divide-and-conquer) the QR algorithm is used to compute its eigensystem.

1. *Cuppen*(T, Λ, V)
2. $n =$ the order of T
3. **if** ($n < \text{mindac}$)
4. Compute Λ and V using the QR algorithm
5. **else**
6. $s = \lfloor n/2 \rfloor$
7. $\hat{T}_1 = T[1:s, 1:s]$
8. $\hat{T}_1[s, s] = \hat{T}[s, s] - T[s, s+1]$
9. *Cuppen*($\hat{T}_1, \hat{\Lambda}_1, \hat{V}_1$)
10. $\hat{T}_2 = T[s+1:n, s+1:n]$
11. $\hat{T}_2[1, 1] = \hat{T}_2[1, 1] - T[s, s+1]$
12. *Cuppen*($\hat{T}_2, \hat{\Lambda}_2, \hat{V}_2$)
13. $D = \text{diag}(\hat{\Lambda}_1, \hat{\Lambda}_2)$
14. $z = (\hat{V}_1[s, :] \hat{V}_2[1, :])^T$
15. *diagupdate*($D, T[s, s+1], z, \Lambda, Q$)
16. $V[1:s, :] = \hat{V}_1 * Q[1:s, :]$
17. $V[s+1:n, :] = \hat{V}_2 * Q[s+1:n, :]$
18. **end if**
19. **end cuppen**

Algorithm 2.3: Divide-and-conquer eigensystem of a symmetric tridiagonal matrix



Complexity

To apply the formula

$$C_{\text{total}} = C(n) + 2^1 C(2^{-1}n) + 2^2 C(2^{-2}n) + \cdots + 2^m C(2^{-m}n)$$

[see (2.17)] for the total cost of Algorithm 2.3, we must first determine the function $C(n)$. This is essentially the cost of *Cuppen* exclusive of its cost of the recursive calls to itself. An examination of the algorithm shows two potentially significant sources of work: the call to *diagupdate* in statement 15 and the matrix multiplication in statements 16 and 17. We have already noted that the updating requires only $O(n^2)$ operations, while the multiplication requires $\frac{1}{2}n^3$ flam. Thus, the computation is dominated by the matrix multiplication, and we may take

$$C(n) = \frac{n^3}{2}.$$

It is now an easy matter to get a sharp upper bound on C_{total} . Specifically,

$$\begin{aligned} C_{\text{total}} &= \frac{1}{2} \left[n^3 + 2 \frac{n^3}{2^3} + 2^2 \frac{n^3}{4^3} + 2^3 \frac{n^3}{8^3} + \cdots \right] \\ &= \frac{n^3}{2} \left[1 + \frac{1}{4} + \frac{1}{4^2} + \cdots \right] \\ &< \frac{n^3}{2(1 - 1/4)} \\ &= \frac{2}{3}n^3. \end{aligned} \tag{2.19}$$

Thus:

Algorithm 2.3 requires about $\frac{2}{3}n^3$ flam.

We see that our divide-and-conquer algorithm has not really conquered in the sense that it has reduced the order of the work. It is still $O(n^3)$. But if it has not conquered, it has certainly prospered. The order constant $\frac{2}{3}$ is quite respectable — comparable to Gaussian elimination and far better than the QR algorithm.

In addition to the low-order constant, the algorithm benefits from deflations in *diagupdate*, which effectively make part of Q diagonal. With some classes of matrices the number of deflations grows with n , so that the divide-and-conquer approach becomes dramatically faster than the QR approach.

Of the total of $\frac{2}{3}n^3$ flam work $\frac{3}{4}$ (i.e., $\frac{1}{2}n^3$ flam) is done in the very first stage of the recursion, leaving $\frac{1}{4}$ of the work for the remaining stages. The second stage does $\frac{3}{4}$ of this work, leaving only $\frac{1}{16}$ of the total. In general after the k th stage only $\frac{1}{4^k}$ of the total work remains to be done. Thus after a two or three stages, the remaining work is insignificant compared to the work done by the higher stages. This has the consequence that for large matrices, the choice of the point at which one shifts to the QR algorithm (parameterized by *mindaq*) is not very critical. So little work is being done at the lower stages that it does not matter if we use an inefficient algorithm there.

2.3. EIGENVALUES AND EIGENVECTORS OF BAND MATRICES

We begin with a definition.

Definition 2.3. Let A be symmetric of order n . Then A is a **BAND MATRIX OF BAND WIDTH $2p + 1$** if

$$|i - j| > p \implies a_{ij} = 0.$$

Thus if $p = 1$, then A is a tridiagonal matrix. If $p = 2$, then A is *pentadiagonal* and has the form

$$\begin{pmatrix} X & X & X \\ X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X \end{pmatrix}.$$

Band matrices are economical to store. For example, a symmetric pentadiagonal matrix can be represented by $3n - 3$ floating-point numbers. This means that it is possible to store band matrices that are so large that the matrix of their eigenvectors, which is of order n , cannot be stored. Fortunately, in many applications involving band matrices we are interested in only a few eigenpairs, often a few of the largest or smallest. The computation of a selected set of eigenpairs of a large symmetric band matrix is the subject of this subsection.

In outline our algorithm goes as follows.

1. Reduce A to tridiagonal form T
2. Find the selected eigenvalues of T
3. Find the selected eigenvectors by the inverse power method applied to A

Note that we compute the eigenvalues from T but the eigenvectors from A . Another possibility is to store the transformations that reduce A to T , find the eigenvectors of T , and transform back. Unfortunately, if n is large, the transformations will require too much storage. Thus we are forced to find the eigenvectors of A directly from A .

For the most part our algorithms will be cast in terms of absolute tolerances relative to the norm of A . This means that small eigenvalues may be determined inaccurately. How to compute small but well-determined eigenpairs of a general band matrix is at present an unsolved problem.

A special case of our algorithm is when the matrix A is already tridiagonal. In this case it turns out that there are better algorithms than the ones proposed here. However, their complexity puts them beyond the scope of this book, and the reader is referred to the notes and references.

The storage of band matrices

Although we will present our algorithms for band matrices as if they were stored in full arrays, in practice we must come to grips with the fact that such storage is not economical. Specifically, a symmetric band matrix of band width $2p+1$ has approximately $(p+1)n$ nonzero elements on and above its diagonal. If $p \ll n$ it would be a waste of memory to store it naturally in an array of order n . Instead such a matrix is usually stored in an array just large enough to accommodate its nonzero elements. There are many ways of doing this. The current standard is the LAPACK scheme—actually two schemes.

The first scheme stores a band matrix in a $(p+1) \times n$ array as illustrated below for a pentadiagonal matrix.

$$\begin{array}{ccccccccc}
 a_{11} & a_{12} & a_{13} & & & & & & \\
 a_{12} & a_{22} & a_{23} & a_{24} & & * & * & a_{13} & a_{24} & a_{35} \\
 a_{13} & a_{23} & a_{33} & a_{34} & a_{35} & \longleftrightarrow & * & a_{12} & a_{23} & a_{34} & a_{45} \\
 a_{24} & a_{34} & a_{44} & a_{45} & & & a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\
 a_{35} & a_{45} & a_{55} & & & & & & &
 \end{array}$$

The asterisks denote unused elements.

More generally if B is the array in which the matrix is stored, then

$$a_{ij} \longleftrightarrow B[p+1+i-j, j], \quad i \leq j.$$

The other scheme stores the elements according to the formula

$$a_{ij} \longleftrightarrow B[1+i-j, j], \quad i \geq j.$$

Thus one scheme stores the upper part of A , whereas the other stores the lower part. When A is complex and Hermitian, this is not just a rearrangement of the same numbers, since the corresponding upper and lower elements are not equal but are conjugates of one another.

Tridiagonalization of a symmetric band matrix

We will now show how to reduce a symmetric band matrix to tridiagonal form. One possibility is simply to apply Algorithm 1.1 for tridiagonalizing a dense matrix to the full band matrix, zeros and all. However, it is easy to see that even if one takes into account the zero elements in the band matrix, the band width grows as the reduction proceeds, so that eventually one is working with a full dense matrix.

Here we consider another alternative that uses plane rotations and preserves the band size. We will illustrate it for a matrix of band width seven ($p = 3$). We begin by choosing a rotation in the $(3, 4)$ -plane to annihilate the $(4, 1)$ -element (and by symmetry the $(1, 4)$ -element) of A . The application of this rotation introduce new nonzeros outside the band in the $(7, 3)$ - and $(3, 7)$ -elements. This step of the reduction is illustrated below:

We now choose a rotation in the $(6, 7)$ -plane to annihilate the elements outside the band. This places new nonzeros in the $(10, 6)$ - and $(6, 10)$ -elements. However, when we eliminate this element, no new elements are generated, as illustrated below:

$$\left(\begin{array}{ccccccccc} x & x & x & 0 & & & & & \\ x & x & x & x & x & & & & \\ x & x & x & x & x & x & & & \\ 0 & x & x & x & x & x & x & & \\ & x & x & x & x & x & x & x & \\ & & x & x & x & x & x & x & x \\ & & & x & x & x & x & x & x \\ & & & & x & x & x & x & x \\ & & & & & x & x & x & x \\ & & & & & & x & x & x \\ & & & & & & & x & x \\ & & & & & & & & x \end{array} \right) \quad \left(\begin{array}{ccccccccc} x & x & \hat{x} & 0 & & & & & \\ x & x & x & x & x & x & x & & \\ \hat{x} & x & x & x & x & x & x & x & \\ 0 & x & x & x & x & x & x & x & \\ & x & x & x & x & x & x & x & \\ & & x & x & x & x & x & x & x \\ & & & x & x & x & x & x & x \\ & & & & x & x & x & x & x \\ & & & & & x & x & x & x \\ & & & & & & x & x & x \\ & & & & & & & x & x \\ & & & & & & & & x \end{array} \right)$$

Thus we have annihilated the $(4, 1)$ - and $(1, 4)$ -elements while preserving the band structure.

We are now ready to work on the $(3, 1)$ -element. A rotation in the $(2, 3)$ -plane annihilates this element, but introduces nonzeros in the $(6, 2)$ - and $(6, 2)$ -elements. These elements can be chased out of the matrix as illustrated below:

Thus we have reduced the first row and column to tridiagonal form.

The reduction now continues by reducing the second row and column, then the third row and column, and so on. The actual code is tedious and not very informative, and we will not give it here.

To get an operation count for this algorithm, we will begin by counting plane rotations. The key observation is that when an element is being chased out of the matrix it moves down the matrix in steps of length p . This can be inferred from our example and easily confirmed in general. Hence when the first row and column are being processed, each zero introduced requires roughly n/p rotations to chase the element outside the band to the end of the matrix. Since we must zero out $p-1$ elements in the first row and column, the total number of rotations is roughly $n(p-1)/p$. Similarly it requires $(n-1)(p-1)/p$ rotations to process the second row and column, $(n-2)(p-1)/p$ to process the third, and so on. Hence the total number of rotations for this algorithm is about

$$\sum_{i=1}^n (n-i) \frac{p-1}{p} \cong \frac{n^2}{2} \frac{p-1}{p}.$$

Remembering that we only operate on the upper or lower part of the matrix, we see that each rotation is applied to $2(p+1)$ pairs of elements (here we count the adjustment of the 2×2 block on the diagonal as two applications). Thus the total number of applications of rotations to pairs of elements is

$$n^2 \left(p + \frac{1}{p} \right).$$

We can convert this to multiplications and additions as usual.

Thus the reduction is truly an $O(n^2)$ process. On the other hand, since the reduction generates $O(n^2)$ rotations we cannot afford to save or accumulate the transformations when n is very large.

An important implication of these counts is that if we can afford n^2 units of storage to accumulate transformations we are better off using the standard Algorithm 1.1. It is true that the reduction itself requires $O(n^3)$ operations as opposed to $O(n^2)$ for the banded reduction. But for both algorithms the accumulation of the transformations requires $O(n^3)$ operations, and the extra work in accumulating plane rotations, as opposed to Householder transformations, tips the scales against the band algorithm.

The algorithm is stable in the usual sense. The computed tridiagonal matrix is orthogonally similar to $A + E$, where $\|E\|_2/\|A\|_2$ is of the order of ϵ_M . However, in this case E does not share the banded structure of A . But it is symmetric, and hence the eigenvalues of A are moved by no more than quantities of order $\|A\|_2 \epsilon_M$ (Theorem 3.8, Chapter 1).

Inertia

We now turn to the problem of computing some of the eigenvalues of a symmetric tridiagonal matrix. The basic tool is a method for counting how many eigenvalues lie

to the left of a point on the real line. We begin with the mathematical foundations of this method.

Let A be symmetric. The number of eigenvalues lying above the point λ on the real line is the same as the number of eigenvalues of $A - \lambda I$ lying above zero. Thus to count the location of the eigenvalues of a symmetric matrix with respect to an arbitrary point, it is sufficient to be able to count their locations with respect to zero. This suggests the following definition.

Definition 2.4. Let A be symmetric. Then $\text{inertia}(A)$ is the triplet (ν, ζ, π) , where ν is the number of negative eigenvalues of A , ζ is the number of zero eigenvalues of A , and π is the number of positive eigenvalues of A .

Although inertia is defined by the eigenvalues of a matrix, it is much easier to compute than the eigenvalues themselves. The reason is that inertia is invariant under *congruence transformations*, i.e., transformations of the form $U^T A U$, which are more general than similarity transformations. Specifically, we have the following theorem.

Theorem 2.5 (Sylvester, Jacobi). Let A be symmetric and U be nonsingular. Then

$$\text{inertia}(A) = \text{inertia}(U^T A U).$$

Proof. The proof is by contradiction. Suppose that A has more positive eigenvalues than $U^T A U$. Let \mathcal{X} be the space spanned by the eigenvectors corresponding to the positive eigenvalues of A . Then

$$x \in \mathcal{X} \implies x^T A x > 0.$$

Now let \mathcal{Y} be the space spanned by the vectors corresponding to the nonpositive eigenvalues of $U^T A U$, and let $\mathcal{Z} = U\mathcal{Y}$. Then

$$z \in \mathcal{Z} \implies z^T A z \leq 0.$$

It follows that $\mathcal{X} \cap \mathcal{Z} = \{0\}$. But the dimension of \mathcal{X} is the number of positive eigenvalues of A and the dimension of \mathcal{Z} is the number of nonpositive eigenvalues of $U^T A U$. By hypothesis, the sum of these dimensions is greater than the order of A . Hence they have a nonzero vector in common — a contradiction.

Similarly, we can show that the number of negative eigenvalues of A is the same as the number of negative eigenvalues of $U^T A U$ and therefore that the number of zero eigenvalues is the same for both matrices. ■

Inertia and the LU decomposition

Turning now to the computation of inertia, we begin with a constructive theorem for determining inertia.

Theorem 2.6. Let A be symmetric and suppose that the leading principle submatrices of A are nonsingular. Then there is a unique unit lower triangular matrix L and a unique upper triangular matrix U such that

$$A = LU.$$

Moreover if $D_U = \text{diag}(u_{11}, u_{22}, \dots, u_{nn})$, then $\text{inertia}(A) = \text{inertia}(D_U)$.

Proof. It is well known that if the leading principal submatrices of A are nonsingular then A has a unique factorization $A = LU$ into a unit lower triangular matrix L and an upper triangular matrix U (Theorem I:3.1.3). Since A is nonsingular, the diagonal elements of U are nonzero. Hence we can write $A = LD_U(D_U^{-1}U)$. By symmetry $(D_U^{-1}U)^T(LDU)$ is an LU decomposition of A , and hence by the uniqueness of LU decompositions, $D_U^{-1}U = L^T$, so that $A = LD_UL^T$. Hence A and D_U are congruent and have the same inertia. ■

It follows that if we have an LU decomposition of A , then we can read off the inertia of A by looking at the diagonal elements of U . Now the LU decomposition of a matrix can be computed by Gaussian elimination (see Chapter I:3). Hence, in principle, we can compute the inertia of A by performing Gaussian elimination and looking at the diagonal elements of U (which, in fact, are just the pivot elements in the elimination). Unfortunately, Gaussian elimination requires pivoting for stability, and the usual pivoting procedures do not result in a U-factor that has anything to do with inertia. But if A is tridiagonal, we can compute the diagonals of U in a stable manner. Let us see how.

The inertia of a tridiagonal matrix

Consider a symmetric tridiagonal matrix T , written in the usual form:

$$T = \begin{pmatrix} d_1 & e_1 & & & \\ e_1 & d_2 & e_2 & & \\ & e_2 & d_3 & e_3 & \\ & & \ddots & \ddots & \ddots \\ & & & e_{n-2} & d_{n-1} & e_{n-1} \\ & & & & e_{n-1} & d_n \end{pmatrix}. \quad (2.20)$$

Since we will later be concerned with the inertia of $T - \lambda I$ as a function of λ , we will work with the matrix

$$T(\lambda) = T - \lambda I \begin{pmatrix} d_1 - \lambda & e_1 & & & \\ e_1 & d_2 - \lambda & e_2 & & \\ & e_2 & d_3 - \lambda & e_3 & \\ & & \ddots & \ddots & \ddots \\ & & & e_{n-2} & d_{n-1} - \lambda & e_{n-1} \\ & & & & e_{n-1} & d_n - \lambda \end{pmatrix}.$$

Given a tridiagonal matrix T in the form (2.20) and a constant λ the function *LeftCount* returns a count of the eigenvalues less than λ . The parameter $umin > 0$ is a minimum value on the absolute value of the current $u_i(\lambda)$ defined by (2.21).

```

1.  LeftCount( $d, e, \lambda$ )
2.  LeftCount = 0
3.   $u = d[1] - \lambda$ 
4.  if ( $u < 0$ ) LeftCount = LeftCount + 1 fi
5.  for  $i = 2$  to  $n$ 
6.    if ( $|u| < umin$ )
7.      if ( $u \geq 0$ )
8.           $u = umin$ 
9.      else
10.         $u = -umin$ 
11.    end if
12.  end if
13.   $u = d[i] - \lambda + e[i-1]^2/u$ 
14.  if ( $u < 0$ ) LeftCount = LeftCount + 1 fi
15. end for  $i$ 
16. end LeftCount
```

Algorithm 2.4: Counting left eigenvalues

If we perform Gaussian elimination on this matrix we get the following recursion for the diagonals of the U-factor:

$$\begin{aligned} u_1(\lambda) &= d_1 - \lambda, \\ u_i(\lambda) &= d_i - \lambda - \frac{e_{i-1}^2}{u_{i-1}(\lambda)}, \quad i = 2, \dots, n. \end{aligned} \tag{2.21}$$

The generation of $u_i(\lambda)$ requires a division by $u_{i-1}(\lambda)$, and we must specify what happens when $u_{i-1}(\lambda)$ is zero. It turns out that we can replace $u_{i-1}(\lambda)$ by a suitably small number. The choice of such a number is a matter of some delicacy and depends on how the matrix is scaled. We will return to this point later.

Algorithm 2.4 counts the eigenvalues to the left of λ . Here are some comments.

- The algorithm requires $2n$ fladd n flmlt and n fldiv. In applications where the count must be repeated for many different values of λ , it will pay to modify *LeftCount* to accept a precomputed array of the numbers e_i^2 .
- The method is stable. In particular, it can be shown that the the computed sequence $u_i(\lambda)$ is an exact sequence of a perturbed matrix $\tilde{T}(\lambda)$ whose elements satisfy

$$|\tilde{d}_i - d_i| \leq 3|d_i - \lambda|\epsilon_M + umin$$

and

$$|\tilde{e}_i - e_i| \leq 3|e_i|\epsilon_M.$$

This result implies that the count will be correct unless λ is extremely close to an eigenvalue of T — so close that in the presence of rounding error the two can scarcely be distinguished.

These results do not imply that unpivoted Gaussian elimination applied to a tridiagonal matrix is stable. The stability concerns only the u_i , not the LU decomposition as a whole.

- The choice of $umin$ depends on the scaling of T and how accurately we wish to determine the count. One possibility is to make the tolerance relative to e_i , so that at the i th step we take

$$umin = |e_{i-1}|\epsilon_M.$$

This choice assumes that the matrix has been scaled so that $e_i/umin$ will not overflow.

- In replacing u with $\pm umin$, we choose the sign so that it is consistent with our previous adjustment of *LeftCount*

Calculation of a selected eigenvalue

Suppose that the eigenvalues of T are ordered so that

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

and we wish to compute the eigenvalue λ_m . We can do so by combining *LeftCount* with a technique called interval bisection.

We begin by using Gershgorin's theorem to calculate a lower bound a on λ_1 and an upper bound b of λ_n , so that $LeftCount(a) = 0$ and $LeftCount(b) = n$. Set $c = (a + b)/2$ and $k = LeftCount(c)$. There are two cases.

First, if $k \geq m$, then $\lambda_m \leq c$. Consequently, $\lambda_m \in [a, c]$. Second, if $k < m$, then $\lambda_m \geq c$. Consequently, $\lambda_m \in [c, b]$. In either case, we have isolated λ_m in an interval that is half the width of the original. If we rename this interval $[a, b]$, we can repeat the procedure until the interval width is less than a prescribed tolerance. This procedure is called *interval bisection*.

Algorithm 2.5 uses this procedure to find λ_m . Here are some comments.

- Except perhaps at the end, the size of the interval $[a, b]$ halves at each step. Consequently, after k iterations, the difference between the current a and b is bounded by $|b - a| \leq 2^{-k}|b_0 - a_0|$, where a_0 and b_0 denote the initial values. If we demand that this bound be less than eps , then the method must converge in about $\log_2(|b_0 - a_0|/eps)$ iterations.

Let the tridiagonal matrix T be represented in the form (2.20) and let the eigenvalues of T satisfy

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

The function *FindLambda* finds an approximation to λ_m . It uses the function *LeftCount* (Algorithm 2.4). The quantity *eps* is a generic convergence criterion.

```

1.   FindLambda( $d, e, m$ )
      Find upper and lower bounds on  $\lambda_m$ 
2.    $a = d(1)|e(1)|; b = d(1) + |e(1)|$ 
3.   for  $i = 2$  to  $n-1$ 
4.        $a = \min\{a, d(i) - |e(i-1)| - |e(i)|\}$ 
5.        $b = \max\{b, d(i) + |e(i-1)| + |e(i)|\}$ 
6.   end for  $i$ 
7.    $a = \min\{a, d(n) - |e(n-1)|\}$ 
8.    $b = \max\{b, d(n) + |e(n-1)|\}$ 
      Find  $\lambda_m$  by interval bisection
9.   while ( $|b-a| > \text{eps}$ )
10.     $c = (a+b)/2$ 
11.    if ( $c = a$  or  $c = b$ ) return  $c$  fi
12.     $k = \text{LeftCount}(d, e, c)$ 
13.    if ( $k < m$ )
14.         $a = c$ 
15.    else
16.         $b = c$ 
17.    end if
18.   end while
19.   return  $(a+b)/2$ 
20. end FindLambda
```

Algorithm 2.5: Finding a specified eigenvalue

- Statement 11 takes care of an interesting anomaly. If eps is too small, it is possible to get endpoints a and b that differ only in a single bit. In that case, c must be equal to one of a and b , and without statement 11 the algorithm would cycle indefinitely.
- The exact choice of eps depends on the accuracy required by the user. The computation of eigenvectors requires very accurate eigenvalues, and in this case a choice like

$$\text{eps} = 4(|a| + |b|)\epsilon_M$$

will result in low relative error for all nonzero eigenvalues.

- If the eigenvalue being sought is simple, then eventually it will be isolated in the interval $[a, b]$. At this point it may pay to shift over to an iteration that converges more quickly—e.g., the secant method. Although *LeftCount* does not produce a function whose zero is the desired eigenvalue, the product of the u 's it generates is the value of the characteristic polynomial at c .
-

The function *FindLambda* can obviously be used to find a sequence of several consecutive eigenvalues. It can also be used to find all the eigenvalues in an interval. But this is inefficient for two reasons. First the computation of the Gerschgorin bounds is repeated with each call to *FindLambda*. Second, in the process of finding one eigenvalue, one generates bounds on the other eigenvalues, bounds that can be used to speed up the bisection process. Therefore, when more than one eigenvalue is to be found, special code adapted to the task at hand should be used.

Selected eigenvectors

We now turn to the problem of computing the eigenvectors corresponding to a set of eigenvalues of a symmetric matrix A . For definiteness we will assume that these eigenvalues are

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$$

and that they are approximated by the quantities

$$\mu_1 \leq \mu_2 \leq \cdots \leq \mu_m.$$

We will make no assumptions about how the μ_k were obtained; but because we are going to use inverse iteration to compute their corresponding eigenvectors, we will assume that they are accurate to almost full working precision.

The first thing we must decide is what we can actually compute. If the eigenvalue λ_k is poorly separated from the other eigenvalues of A , then the eigenvector corresponding to λ_k will be sensitive to perturbations in A , and we will not be able to calculate it accurately [see (3.28), Chapter 1]. Instead we will attempt to find an approximation x_k such that the normalized residual

$$\frac{\|(A - \mu_k I)x_k\|_2}{\|A\|_2\|x\|_2}$$

is small. The justification for this criterion may be found in Theorem 1.3, Chapter 2, which says that x_k is the exact eigenvector of $A + E$, where $\|E\|_2/\|A\|_2$ is equal to the scaled residual norm. For the implications of this result, see the discussion following the theorem.

A nice feature of this criterion is that it is easy to test. Specifically, if we take u normalized so that $\|u\|_2 = 1$ as a starting vector for the inverse power step and let v be the solution of the equation $(A - \mu_k I)v = u$, then

$$\frac{\|(A - \mu_k I)v\|_2}{\|A\|_2\|v\|_2} = \frac{1}{\|A\|_2\|v\|_2}. \quad (2.22)$$

Thus if the quantity $\|u\|_2/\|A\|_2\|v\|_2$ is small enough, say near the rounding unit, then we can accept $x_k = v/\|v\|_2$ as a normalized eigenvector.

There remains the question of whether the inverse power method is capable of producing a vector with a sufficiently small residual. Once again, let us start with a vector u normalized so that $\|u\|_2 = 1$. Let y_k be the normalized eigenvector of A corresponding to λ_k . Then if $v = (A - \mu_k I)^{-1}u$, we have

$$\|v\|_2 \geq |y_k^T v| = \frac{|y_k^T u|}{|\lambda_k - \mu_k|}.$$

It follows from (2.22) that

$$\frac{\|(A - \mu_k I)v\|_2}{\|A\|_2\|v\|_2} \leq \frac{|\lambda_k - \mu_k|}{|y_k^T u|\|A\|_2}. \quad (2.23)$$

Now suppose that μ_k is a good relative approximation to λ_k , so that

$$|\lambda_k - \mu_k| \leq \gamma |\lambda_k| \epsilon_M$$

for some modest constant γ . Then since $|\lambda_k| \leq \|A\|_2$, we have from (2.23) that

$$\frac{\|(A - \mu_k I)v\|_2}{\|A\|_2\|v\|_2} \leq \frac{\gamma \epsilon_M}{|y_k^T u|}. \quad (2.24)$$

Thus the effectiveness of the inverse power method depends on the size of the component $y_k^T u$ of u along y_k . If u is chosen at random, we can expect this component to be of the order $1/\sqrt{n}$ [see (1.9), Chapter 2], and consequently we can estimate the bound (2.24) by

$$\frac{\|(A - \mu_k I)v\|_2}{\|A\|_2\|v\|_2} \stackrel{\text{est}}{\leq} \sqrt{n} \gamma \epsilon_M. \quad (2.25)$$

Thus if $n = 10,000$, the estimate is only 100 times larger than $\gamma \epsilon_M$. However, note that (2.25) is likely to be an overestimate since it ignores the contribution of the components of u along the $n-1$ remaining eigenvectors of A .

If it turns out that the scaled residual is not satisfactorily small, we can always try again with $u = v/\|v\|_2$. Since this u has had its components associated with the eigenvalues near μ_k enhanced, we can expect this second iteration to do the job. For safety one might allow additional iterations.

In addition to having small scaled residual norms, we would like our computed eigenvectors to be orthogonal. Unfortunately, it is easy to lose orthogonality when two of the λ_k are close, as the following example shows.

Example 2.7. A matrix A was constructed by generating a random orthogonal matrix Q and setting $A = Q\Lambda Q^T$, where

$$\Lambda = \text{diag}(1, 1 + 10^{-8}, 2).$$

Thus A has two very close eigenvalues near one and an isolated eigenvalue at two.

Let us approximate the first two eigenvalues by

$$\begin{aligned}\mu_1 &= 1 + 10^{-15}, \\ \mu_2 &= 1 + 10^{-8} + 10^{-15},\end{aligned}$$

and perform inverse iteration with each μ_k to approximate the corresponding eigenvectors. The residual norms for the approximations x_1 and x_2 are about $1 \cdot 10^{-15}$ and $2 \cdot 10^{-15}$, which is extremely good. However, when we compute the inner product $x_1^T x_2$, we find that

$$x_1^T x_2 = -1 \cdot 10^{-7}.$$

The vectors x_1 and x_2 have lost half the orthogonality we expect from eigenvectors of a symmetric matrix.

A little perturbation theory will help us understand what is going on here. The eigenvalues λ_1 and λ_2 are very close— 10^{-8} apart. Since the residual norms are of order 10^{-15} , the computed vectors are exact vectors of A perturbed by matrices of order 10^{-15} . By (3.28), Chapter 1, these perturbations will move the computed eigenvectors by about 10^{-8} . Since the errors in the eigenvectors are independent of one another, there is a corresponding deterioration in the orthogonality between the two vectors.

The cure for this problem is to force x_2 to be orthogonal to x_1 . Specifically, if $\|x_1\|_2 = 1$, the vector

$$\hat{x}_2 = \frac{(I - x_1 x_1^T)x_2}{\|(I - x_1 x_1^T)x_2\|_2}$$

is easily seen to have 2-norm one and to be orthogonal to x_2 . If, in our example, we perform this orthogonalization, we find that $x_1^T \hat{x}_2 = 7 \cdot 10^{-18}$, so that x_1 and \hat{x}_2 are orthogonal to working accuracy. Moreover, the residual norm for \hat{x}_2 is $2 \cdot 10^{-15}$, so that

the quality of \hat{x}_2 as an eigenvector has not been compromised by the orthogonalization process.

The orthogonalization process can be adapted to handle a sequence of close eigenvalues. We divide the μ_k into clusters according to some tolerance. As we pass through a cluster we use the Gram–Schmidt algorithm to orthogonalize each vector against the previously computed vectors in the cluster. Since the details of the orthogonalization are tricky (for the full story see §I:4.1.4), we leave the process to a routine

$$\text{gsreorthog}(X, u, v, r, \rho) \quad (2.26)$$

that orthogonalizes a vector u against the columns of X returning the result in v (which can be identified with u). Specifically, the routine returns a normalized vector v that is orthogonal to the columns of X and satisfies

$$\rho v = u - Xr.$$

Unfortunately, the orthogonalization complicates our convergence criterion. To see why, suppose that we have a group $\mu_{k-g+1}, \dots, \mu_k$ of g approximate eigenvalues and have determined normalized approximate eigenvectors $x_{k-g+1}, \dots, x_{k-1}$, with x_k to be determined. We will also assume that we have bounds

$$\|Ax_i - \mu_i x_i\|_2 \leq \eta_i$$

on the residual norms of the x_i .

Now suppose that we have computed v by the inverse power method with shift μ_k applied to normalized vector u . If a stable method is used to calculate v , it satisfies

$$(A + E - \mu_k I)v = u,$$

where E is of the order of the rounding unit. Now the approximate eigenvector x_k produced by *gsreorthog* satisfies

$$\rho x_k = v - Xr.$$

Hence

$$(A - \mu_k)(\rho x_k + Xr) = u - Ev.$$

It follows that

$$\begin{aligned} \rho(A - \mu_k I)x_k &= u - Ev - (A - \mu_k I)Xr \\ &= u - Ev - \sum_{i=k-g+1}^{k-1} r_i[(A - \mu_i I)x_i - (\mu_i - \mu_k)x_i]. \end{aligned}$$

Hence

$$\|Ax_i - \mu_k x_i\|_2 \leq \frac{1 - \|E\|_2\|v\|_2 + \sum_{i=k-g+1}^{k-1} |r_i|(\eta_i + |\mu_i - \mu_k|)}{|\rho|} \equiv \eta_k \quad (2.27)$$

Let

$$\mu_1 \leq \mu_2 \leq \cdots \leq \mu_m$$

be a consecutive set of eigenvalues of a symmetric matrix A with relative accuracy approximating the rounding unit ϵ_M . This algorithm returns in the columns of the array X approximations to the eigenvectors corresponding to the μ_k . The algorithm uses the routine *gsreorthog* to orthogonalize a vector against a set of vectors [see (2.26)].

```

1.  groupsep =  $\|A\|_\infty * 10^{-3}$ 
2.  tol =  $100 * \sqrt{n} * \|A\|_\infty * \epsilon_M$ 
3.  maxit = 5; g = 1
4.  for k = 1 to m
5.    if ( $k > 1$ )
6.      if ( $\mu_k \leq \mu_{k-1} + groupsep$ ) g = g+1 else g = 1 fi
7.    end if
8.    it = 1
9.    u = a vector of random normal deviates
10.   if ( $g > 1$ ) gsreorthog( $X[:, k-g+1, k-1]$ , u, u, r, rho)
11.     else u = u/ $\|u\|_2$ 
12.   end if
13.   it = 1
14.   while (true)
15.     Solve the system  $(A - \mu_k I)v = u$ 
16.     if ( $g > 1$ ) gsreorthog( $X[:, k-g+1, k-1]$ , v, u, r, rho)
17.       else rho =  $\|v\|_2$ ; u = v/rho
18.     end if
19.      $\eta_k = \|u\|_2 + \|v\|_2 * \sqrt{(n)} * \|A\|_\infty * \epsilon_M$ 
20.     for i = k-g+1 to k-1
21.        $\eta_k = \eta_k + (\eta_i + |\mu_i - \mu_k|) * |r_{i-k+g}|$ 
22.     end for i
23.     it = it + 1
24.     if (it > maxit) error fi
25.   end while
26. end for k

```

Algorithm 2.6: Computation of selected eigenvectors



is a bound on the residual norm for x_k .

Algorithm 2.6 pulls all these threads together. There are several comments to be made about it.

- An absolute criterion is used to group eigenvalues for orthogonalization (see statements 1 and 6). If we use too liberal a grouping criterion, we will perform a lot of unnecessary orthogonalizations. On the other hand, if the criterion for belonging to a group is too small, the distance between successive groups can be small, resulting in loss of orthogonality between groups. (The routine *gsreorthog* insures that the vectors within a group are orthogonal to working accuracy.)
 - The `while` loop can be exited with a new vector or with an error return when the number of iterations exceeds *maxit*.
 - The starting vector for the inverse power method is a random normal vector. It is orthogonalized against the other members of the group to purge it of components that will contribute to loss of orthogonality.
 - The algorithm uses the residual bound (2.27) to test for convergence. Limited experiments show that the bound tracks the actual residual norm with reasonable fidelity.
 - We have not specified how to solve the system in statement 15. The matrix $A - \mu_k I$ will in general be indefinite, and some form of pivoting is required for stability. Unfortunately, diagonal pivoting, which preserves symmetry, destroys band structure. Thus the natural choice is Gaussian elimination with partial pivoting. This method does not respect symmetry. But the *L* and *U* factors it computes are banded, with *L* of bandwidth $p+1$ and *U* of bandwidth $2p+1$. Thus the storage requirements for the algorithm are four times the amount required to represent the original symmetric matrix. For more see §I:3.2.4.
-

It should be stressed that Algorithm 2.6 can be expected to work only because we have aimed low. It is intended for use on balanced matrices eigenvalues and uses absolute criteria to determine when quantities can be ignored (e.g., see statements 1 and 2). It will not work on graded matrices, and it is not clear that it can be modified to do so. Moreover, it can and does fail on groups of eigenvalues that are separated by up to $10^3 \epsilon_M$. For example, the algorithm is not a reliable way of computing a complete set of eigenvectors of a small perturbation of the identity matrix.

Even as it stands Algorithm 2.6 involves a number of ad hoc decisions—e.g., the clustering criterion, the choice of a starting vector, the convergence criterion, when to orthogonalize. There is a certain common sense to our choices, but the same can be said of other choices. See the notes and references.

2.4. NOTES AND REFERENCES

Updating the spectral decomposition

This updating algorithm has a long history, beginning with a survey of eigenvalue updating problems by Golub [87, 1973]. Bunch, Nielsen, and Sorensen [30, 1978] formulated the algorithm treated here and noted that there were problems with maintaining orthogonality among the eigenvectors. Kahan, in unpublished communications, suggested that the algorithm can be made to work provided extended precision is used at certain critical points and noted that for IEEE standard arithmetic the extended precision arithmetic could be simulated. Sorensen and Tang [247, 1991] showed that more was required and gave an algorithm that worked with simulated extended precision on all IEEE compliant machines. Finally, Gu and Eisenstat [100, 1994] noted that by recomputing the z_i so that the $\tilde{\lambda}_i$ are exact eigenvalues of the corresponding matrix the orthogonality could be maintained in the eigenvectors without resorting to higher precision arithmetic. The stopping criterion in the secular equation solver insures that the procedure is backward stable. For an analysis of methods for solving the secular equation see [164].

An implementation of this method can be found in the LAPACK code xLAED1.

The divide-and-conquer algorithm

The divide-and-conquer algorithm is due to Cuppen [49]. It was later modified by Dongarra and Sorensen [66]. For an error analysis see [13]. The algorithm has been implemented in the LAPACK routine SLAED0.

Gu and Eisenstat [101] propose a different divide-and-conquer algorithm in which the critical step is computing the spectral decomposition of an arrowhead matrix — that is, a matrix of the form

$$\begin{pmatrix} X & X & X & X & X & X \\ X & X & 0 & 0 & 0 & 0 \\ X & 0 & X & 0 & 0 & 0 \\ X & 0 & 0 & X & 0 & 0 \\ X & 0 & 0 & 0 & X & 0 \\ X & 0 & 0 & 0 & 0 & X \end{pmatrix}.$$

Reduction to tridiagonal form

The technique of reducing a band matrix to tridiagonal form is due to Schwarz [235].

Inertia and bisection

The fact that the inertia of a matrix is preserved by congruence was discovered by Sylvester [272, 1853] and independently by Jacobi [129, 1857, posthumous].

Givens [83, 1954], who was the first to find eigenvalues of tridiagonal matrices by bisection, did not use Algorithm 2.4 to count eigenvalues. Instead he used closely

related Sturm sequences. Specifically, If $T - \lambda I$ has the LU decomposition LU , then the product $p_k(\lambda) = u_{11} \cdots u_{kk}$ is the determinant of the leading principal minor of $T - \lambda I$. Clearly, the $p_k(\lambda)$ change sign when the u_k do. The polynomials $p_k(\lambda)$ are called a Sturm sequence (see [122, §2.4] for definitions and properties). Unfortunately, they are prone to overflow and underflow.

The Handbook program *bisect* [15] appears to be the first to use the diagonal elements of U to count eigenvalues, although the authors justified their use by an appeal to Sturm sequences. This program also saves information from previous counts to speed up the convergence of the bisection procedure. There are bisection codes in EISPACK and LAPACK.

Eigenvectors by the inverse power method

Algorithm 2.6 is included to illustrate the problems of implementing a power method. For working code see the Handbook routine *tristurm* [305, II/18] or the LAPACK routine *SSTEIN*.

In his Ph.D. thesis Dhillon [65] has described an $O(n^2)$ to compute all the eigenvalues and eigenvectors of a tridiagonal matrix. The thesis also contains a review and critique of previous attempts to use the inverse power method to compute eigenvectors and an extensive bibliography.

Jacobi's method

Jacobi's method for the solution of the symmetric eigenvalue problem uses rotations to annihilate off-diagonal elements of the matrix in question. Specifically, given a current iterate A_k and a pivot element $a_{i_k, j_k}^{(k)}$, a rotation R_k in the (i_k, j_k) -plane is determined so that the (i_k, j_k) -element of $A_{k+1} = R_k^T A_k R_k$ is zero. If we let τ_k^2 be the sum of squares of the diagonal elements of A_k and σ_k^2 be the sum of squares of the off-diagonal elements, then

$$\|A\|_F^2 = \|A_k\|_F^2 = \tau_k^2 + \sigma_k^2$$

and

$$\tau_{k+1}^2 = \tau_k^2 + a_{i_k j_k}^{(k)}.$$

Hence, if we choose (i_k, j_k) so that $a_{i_k j_k}^{(k)}$ is maximal, then it is easy to show that $\tau_k^2 \rightarrow \|A\|_F^2$. In other words, A_k approaches a diagonal matrix.

Jacobi [127, 1845] first proposed this method as a technique for reducing the normal equations from least squares problems to diagonally dominant form, after which it could be readily solved by what is now called the point Jacobi iteration [285, §3.3.1]. In 1846 [128], he applied the same idea to reduce the symmetric eigenproblem to a diagonally dominant one, after which the eigenvectors could be found by an iteration based on a perturbation expansion. Thus, Jacobi regarded the algorithm now named for him as a preprocessing step, not as a complete algorithm in itself.

According to Goldstine and Horowitz [84], the method was rediscovered by Goldstine, von Neumann, and Murray in 1949 (also see [85]). The subsequent literature was dominated by two aspects of the convergence of the method. First, a search for a maximal element is too expensive on a digital computer. Consequently, it was proposed to sweep through the elements of the matrix in a systematic order. This cyclic Jacobi method considerably complicates the convergence proofs. Second, if the matrix has no multiple eigenvalues, its asymptotic convergence is at least quadratic. The situation with multiple eigenvalues and clusters of eigenvalues is considerably more complicated. For more on these topics see [75, 107, 110, 171, 298]. An interesting alternative to the cyclic method is the threshold method [212], in which all off-diagonal elements greater than a given threshold are annihilated until there are none left above the threshold, after which the process is repeated with a smaller threshold.

Like other algorithms that proceed by rotations, Jacobi's method works well with graded matrices and matrices with small, well-determined eigenvalues — at least when the matrix is positive definite. For more on this topic see [59, 172].

Until it was superseded by the QR algorithm, Jacobi's method was the algorithm of choice for dense symmetric matrix. (For an elegant implementation by Rutishauser — always a name to be reckoned with — see the Handbook code [226].) But since then it has not fared well. It has great natural parallelism, and at one time it and its variants were considered candidates for systolic arrays — a class of computers that never realized their promise. But no one can say for sure what the future holds for this bewitching algorithm.

3. THE SINGULAR VALUE DECOMPOSITION

In this section we will treat methods for computing the singular values and vectors of a general $n \times p$ matrix X . For definiteness we will assume that $n \geq p$ and that X is real. The singular value decomposition is closely allied to the spectral decomposition. In particular, there is an analogue of the QR algorithm to solve the singular value problem. The main purpose of this section is to describe this algorithm.

We will begin with the mathematical background, including the perturbation of singular values and vectors. In §3.2 we will show how the singular value decomposition can be reduced to the solution of a symmetric eigenvalue problem. This method, as we shall see, has limitations that render it unsuitable as a general-purpose routine, though it may be the algorithm of choice in special cases. In §3.3 we will begin our development of the QR algorithm by showing how to reduce a general matrix to bidiagonal form. The QR algorithm itself is treated in §3.4.

3.1. BACKGROUND

In this subsection we will present the background necessary for this chapter. Much of this material has been treated in detail in §I:1.4.3 of this series, and many results will be presented without proof.

Existence and nomenclature

The following theorem gives the particulars of the *singular value decomposition*.

Theorem 3.1. Let $X \in \mathbb{R}^{n \times p}$, where $n \geq p$. Then there are orthogonal matrices U and V such that

$$U^T X V = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}, \quad (3.1)$$

where

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_p)$$

with

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0. \quad (3.2)$$

Here are some comments on this theorem.

- The numbers σ_i are called the *singular values* of X . The nondecreasing order in (3.2) is conventional and will be assumed unless otherwise stated. We will use the notation $\sigma_i(X)$ to denote the i th singular value of X .
- The columns of

$$U = (u_1 \ u_2 \ \dots \ u_n) \quad \text{and} \quad V = (v_1 \ v_2 \ \dots \ v_p)$$

are called *left and right singular vectors of X* . They satisfy

$$X v_i = \sigma_i u_i, \quad i = 1, 2, \dots, p,$$

and

$$X^T u_i = \sigma_i v_i, \quad i = 1, 2, \dots, p.$$

Moreover,

$$X^T u_i = 0, \quad i = p+1, p+2, \dots, n.$$

- It often happens that $n \gg p$, in which case maintaining the $n \times n$ matrix U can be burdensome. As an alternative we can set

$$U_p = (u_1 \ u_2 \ \dots \ u_p)$$

and write

$$X = U_p \Sigma V^T. \quad (3.3)$$

This form of the decomposition is sometimes called the *singular value factorization*.

- Since the 2-norm is unitarily invariant, $\|X\|_2 = \|\Sigma\|_2$. From this it is easily established that:

The 2-norm of a matrix is its largest singular value.

From the definition of the 2-norm it follows that

$$\sigma_1 = \max_{\|v\|_2=1} \|Xv\|_2 = \|X\|_2.$$

- Later it will be convenient to have a notation for the smallest singular value of X . We will denote it by

$$\inf(X) \stackrel{\text{def}}{=} \sigma_p(X).$$

- Since the Frobenius norm is unitarily invariant, it follows that

$$\|X\|_F^2 = \|U^T X V\|_F^2 = \|\Sigma\|_F^2 = \sum_{i=1}^p \sigma_i^2.$$

In other words:

The square of the Frobenius norm of a matrix is the sum of squares of its singular values.

Uniqueness

The singular value decomposition is essentially unique. The singular values themselves are unique. The right singular vectors corresponding to a simple singular value (one distinct from the others) are unique up to a factor of absolute value one. The right singular vectors corresponding to a multiple singular vector are not unique, but the space they span is. Once the right singular vectors have been specified, the left singular vectors corresponding to nonzero singular values are uniquely determined by the relation $Xv_i = \sigma_i u_i$.

Relation to the spectral decomposition

From (3.1) and the fact that U is orthogonal, it follows that

$$V^T X^T X V = \Sigma^2.$$

This shows that:

The squares of the singular values X are the eigenvalues of the CROSS-PRODUCT MATRIX $X^T X$. The right singular vectors are the eigenvectors of $X^T X$. (3.4)

An analogous result holds for XX^T . However, when $n > p$, the matrix XX^T has $n-p$ additional zero eigenvalues. Sometimes they cannot be ignored [see (3.17)], and we will call them *honorary singular values*.

Characterization and perturbation of singular values

Given the relation of the singular value and spectral decompositions it is not surprising that the min-max characterization for eigenvalues (Theorem 3.5, Chapter 1) has an analogue for singular values.

Theorem 3.2. *Let the singular values of X be ordered in descending order:*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p.$$

Then

$$\sigma_k(X) = \min_{\dim(W)=p-k+1} \max_{\substack{w \in W \\ \|w\|_2=1}} \|Xw\|_2, \quad k = 1, 2, \dots, p,$$

and

$$\sigma_k(X) = \max_{\dim(W)=k} \min_{\substack{w \in W \\ \|w\|_2=1}} \|Xw\|_2, \quad k = 1, 2, \dots, p.$$

An immediate consequence of this characterization is the following perturbation theorem for singular values.

Theorem 3.3. *Let the singular values of X be arranged in descending order:*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p.$$

Let the singular values of $\tilde{X} = X + E$ also be ordered in descending order:

$$\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_p.$$

Then

$$|\tilde{\sigma}_i - \sigma_i| \leq \|E\|_2, \quad i = 1, 2, \dots, p.$$

Thus singular values, like the eigenvalues of a symmetric matrix, are perfectly conditioned. However, this does not mean that perturbations of small singular values have high relative accuracy. See Example 3.9, Chapter 1.

First-order perturbation expansions

The perturbation of singular vectors is best approached via first-order perturbation theory. Let $\sigma_1 \neq 0$ be a simple singular value of the $n \times p$ matrix X with normalized left and right singular vectors u_1 and v_1 . (Here we have discarded the conventional downward ordering of the singular values.) Assuming that $n \geq p$, let us partition the matrices of left and right singular vectors in the forms

$$U = (u_1 \ U_2 \ U_3) \quad \text{and} \quad V = (v_1 \ V_2),$$

where U_2 and V_2 have $p-1$ columns. Then

$$(u_1 \ U_2 \ U_3)^T X (v_1 \ V_2) = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{pmatrix}, \quad (3.5)$$

where $\Sigma_2 = \text{diag}(\sigma_2, \dots, \sigma_p)$ contains the singular values of X other than σ_1 .

Now let $\tilde{X} = X + E$, where E is presumed small. Then we can write

$$(u_1 \ U_2 \ U_3)^T \tilde{X} (v_1 \ V_2) = \begin{pmatrix} \sigma_1 + \varphi_1 & f_{12}^T \\ f_{21} & \Sigma_2 + F_{22} \\ F_{31} & F_{32} \end{pmatrix}, \quad (3.6)$$

where $\varphi_1 = u_1^T E u_1$, $e_{12}^T = u_1^T E V_2$, etc.

The singular vectors corresponding to σ_1 in the matrix $U^T X V$ are both e_1 (of appropriate dimensions). We will seek the singular value of $U^T \tilde{X} V$ in the form $\sigma_1 + \theta_1$, where θ_1 is assumed to go to zero along with E . We will seek the corresponding left and right singular vectors of $U^T \tilde{X} V$ in the forms $(1 \ g_2^T \ g_3^T)^T$ and $(1 \ h_2^T)^T$, where the g_2 , g_3 , and h_2 are also assumed to go to zero along with E .

Now consider the relation

$$\begin{pmatrix} \sigma_1 + \varphi_1 & f_{12}^T \\ f_{21} & \Sigma_2 + F_{22} \\ F_{31} & F_{32} \end{pmatrix} \begin{pmatrix} 1 \\ h_2 \end{pmatrix} = (\sigma_1 + \theta_1) \begin{pmatrix} 1 \\ g_2 \\ g_3 \end{pmatrix}. \quad (3.7)$$

From the first row of this relation, we have

$$\sigma_1 + \varphi_1 + f_{12}^T h_2 = \sigma_1 + \theta_1.$$

Since both f_{12}^T and h_2 go to zero with E , their product must eventually become negligible, and we can write

$$\theta_1 \cong \varphi_1.$$

From the third row of (3.7) we have

$$f_{31} + F_{32} h_2 = (\sigma_1 + \theta_1) g_3.$$

Ignoring higher-order terms, as above, we get

$$g_3 \cong \sigma_1^{-1} f_{31}. \quad (3.8)$$

The second row of (3.7) gives us

$$f_{21} + (\Sigma_2 + F_{22}) h_2 = (\sigma_1 + \theta_1) g_2,$$

or

$$\sigma_1 g_2 - \Sigma_2 h_2 \cong f_{21}. \quad (3.9)$$

This expression is not enough to determine g_2 and h_2 . But from the second row of the relation

$$\begin{pmatrix} \sigma_1 + \varphi_1 & f_{21}^T & f_{31}^T \\ f_{12} & \Sigma + F_{22}^T & F_{32}^T \end{pmatrix} \begin{pmatrix} 1 \\ g_2 \\ g_3 \end{pmatrix} = (\sigma_1 + \theta_1) \begin{pmatrix} 1 \\ h_2 \end{pmatrix}$$

we get

$$\sigma_1 h_2 - \Sigma_2 g_2 \cong f_{12}. \quad (3.10)$$

The approximations (3.9) and (3.10) can be readily solved for g_2 and h_2 to give the approximations

$$g_2 \cong (\sigma_1^2 I - \Sigma_2^2)^{-1}(\sigma_1 f_{21} - \Sigma_2 f_{12})$$

and

$$h_2 \cong (\sigma_1^2 I - \Sigma_2^2)^{-1}(\sigma_1 f_{12} - \Sigma_2 f_{21}). \quad (3.11)$$

A simple singular value and its normalized singular vectors are differentiable functions of the elements of its matrix. Hence the quantities θ_1 , g_2 , g_3 , and h_2 are all $O(\|E\|)$, and the products we have ignored in deriving our approximations are all $O(\|E\|^2)$. If we transform these approximations back to the original system, we have the following theorem.

Theorem 3.4. *Let $\sigma_1 > 0$ be a simple singular value of $X \in \mathbb{R}^{n \times p}$ ($n \geq p$) with normalized left and right singular vectors u_1 and v_1 . Let the matrices of left and right singular vectors be partitioned in the form*

$$U = (u_1 \ U_2 \ U_3) \quad \text{and} \quad V = (v_1 \ V_2),$$

where U_2 and V_2 have $p-1$ columns and let

$$\Sigma_2 = U_2^T X V_2 \equiv \text{diag}(\sigma_2, \dots, \sigma_p).$$

Let $\tilde{\sigma}_1$, \tilde{u}_1 , and \tilde{v}_1 be the corresponding quantities for the perturbed matrix $\tilde{X} = X + E$. Then

$$\tilde{\sigma}_1 = \sigma_1 + u_1^T E v_1 + O(\|E\|^2), \quad (3.12)$$

$$\begin{aligned} \tilde{u}_1 &= u_1 + U_2 (\sigma_1^2 I - \Sigma_2^2)^{-1} (\sigma_1 U_2^T E v_1 - \Sigma_2 V_2 E^T u_1) \\ &\quad + \sigma_1^{-1} U_3^T E v_1 + O(\|E\|^2), \end{aligned}$$

and

$$\tilde{v}_1 = v_1 + V_2 (\sigma_1^2 I - \Sigma_2^2)^{-1} (\sigma_1 V_2 E^T u_1 - \Sigma_2 U_2^T E v_1) + O(\|E\|^2).$$

Here are two comments on this theorem.

- The expression (3.12) can be written in the form

$$\tilde{\sigma}_1 = u_1^T \tilde{X} v_1 + O(\|E\|^2).$$

In this form it is seen to be an analogue of the Rayleigh quotient for an eigenvalue (Definition 1.5, Chapter 2).

- The hypothesis that σ_1 be nonzero is necessary. The problem is that singular values are not differentiable at zero. For example, the singular value of a scalar is its absolute value. For more on the behavior of zero singular values, see the notes and references.

The condition of singular vectors

We can use the results of Theorem 3.4 to derive a condition number for a singular vector. We will begin with the right singular vector.

Since we will be concerned with angles between vectors and orthogonal transformations do not change angles, we may work with the transformed matrices in (3.5) and (3.6). Now the normalized perturbed right singular vector is

$$\frac{1}{\sqrt{1 + \|h\|_2^2}} \begin{pmatrix} 1 \\ h_2 \end{pmatrix}. \quad (3.13)$$

By Lemma 3.12, Chapter 1, the sine of the angle between this vector and e_1 is

$$\|h_2\|_2 / \sqrt{1 + \|h_2\|_2^2}.$$

Thus to get a bound on the angle between e_1 and the vector (3.13)—or equivalently between v_1 and \tilde{v}_1 —we must obtain a bound on $\|h_2\|_2$.

Consider the approximation (3.11) for h_2 . Because Σ_2 is diagonal we may write it out component by component:

$$\eta \cong (\sigma_1^2 - \sigma_2^2)^{-1} (\sigma_1 \phi_{12} - \sigma_2 \phi_{21}).$$

Here η , ϕ_{12} , and ϕ_{21} represent corresponding components of h_2 , f_{12} , and f_{21} , and σ_2 is the corresponding diagonal element of Σ_2 . We can rewrite this equation in the form

$$(\sigma_1 - \sigma_2)^{-1} \frac{\sigma_1 \phi_{12} - \sigma_2 \phi_{21}}{\sigma_1 + \sigma_2}.$$

Since σ_1 and σ_2 are positive, the fraction in this expression is a weighted average of ϕ_{12} and $-\phi_{21}$. Hence it must lie between ϕ_{12} and $-\phi_{21}$, and its absolute value is bounded by $\max\{|\phi_{21}|, |\phi_{12}|\}$. It follows that

$$\|h_2\|_2 \leq \frac{\sqrt{\|f_{21}\|_2^2 + \|f_{12}\|_2^2}}{\delta}, \quad (3.14)$$

where

$$\delta = \min\{|\sigma_1 - \sigma_i| : i > 1\}. \quad (3.15)$$

Since U_2 and v_1 are orthonormal,

$$\|f_{21}\|_2 = \|U_2^T E v_1\|_2 \leq \|E\|_2.$$

Similarly, $\|f_{12}\| \leq \|E\|_2$. Hence from (3.14),

$$\|h_2\|_2 \leq \frac{\sqrt{2}\|E\|_2}{\delta}.$$

Now up to terms of order $\|E\|_2$, the quantity $\|h_2\|_2$ is the sine of the angle between v_1 and \tilde{v}_1 . Hence:

Let δ be defined by (3.15). Then

$$\sin \angle(v, \tilde{v}) \leq \frac{\sqrt{2}\|E\|_2}{\delta} + O(\|E\|^2). \quad (3.16)$$

Turning now to the left singular vectors, which we seek in the form $(1 \ g_2^T \ g_3^T)^T$, we see that the sine of the angle between u_1 and \tilde{u}_1 is essentially $\sqrt{\|g_2\|_2^2 + \|g_3\|_2^2}$. We can bound $\|g_2\|_2$ as above; however, if $n > p$, we must take into account g_3 defined by (3.8). We can do this by adjusting δ to take into account the honorary zero singular values — i.e., the additional zero eigenvalues of XX^T . Specifically:

Let

$$\delta_0 = \begin{cases} \delta & \text{if } n = p, \\ \min\{\delta\sigma_1\} & \text{if } n > p, \end{cases} \quad (3.17)$$

where δ is defined by (3.15). Then

$$\sin \angle(u_1, \tilde{u}_1) \leq \frac{\sqrt{2}\|E\|_2}{\delta_0} + O(\|E\|^2).$$

Thus δ^{-1} (or δ_0^{-1}) is a condition number for the singular vectors corresponding to the singular value σ_1 . The number δ is the separation of σ_1 from its companions, and it plays a role analogous to sep in the perturbation of eigenvectors of Hermitian matrices [see (3.28), Chapter 1].

3.2. THE CROSS-PRODUCT ALGORITHM

In (3.4) we have seen that the eigenvalues of the cross-product matrix $A = X^T X$ are the squares of the singular values of X and that columns of the matrix V of the eigenvectors of A are the right singular vectors of X . Moreover, if we know V we

Let $X \in \mathbb{R}^{n \times p}$ ($n \geq p$). This algorithm computes the singular values, the right singular vectors, and the first p left singular vectors of X .

1. Form $A = X^T * X$
2. Compute the spectral decomposition $A = V \Lambda V^T$
3. $\Sigma = \Lambda^{\frac{1}{2}}$
4. $U_p = X * V$
5. Normalize the columns of U_p

Algorithm 3.1: The cross-product algorithm



can recover the first p columns of U via the formula $XV = U_p \Sigma$. All this suggests Algorithm 3.1, which for definiteness we will call the cross-product algorithm.

This algorithm has some attractive features. The formation of A is cheaper than the reduction to bidiagonal form that we will propose later—much cheaper if X is sparse. We already have algorithms to calculate the spectral decomposition. The computation of U_p involves only matrix multiplication, which is cheaper than accumulating transformations. Nonetheless, the algorithm has severe limitations, and it is important to understand what they are.

Inaccurate singular values

The first problem is that we lose information about the singular values in passing from X to A . Let us first consider what happens to the singular values when we round X . This amounts to replacing X by $\tilde{X} = X + E$, where

$$\|E\|_2 \leq \gamma \|X\|_2 \epsilon_M = \sigma_1(X) \epsilon_M.$$

Here ϵ_M is the rounding unit for the machine in question [see (2.3), Chapter 1]. By Theorem 3.3, this perturbation can change the singular values of X by as much as $\sigma_1 \epsilon_M$. Specifically, if $\tilde{\sigma}_i$ is the i th singular value of \tilde{X} , then the relative error in $\tilde{\sigma}_i$ satisfies

$$\frac{|\tilde{\sigma}_i - \sigma_i|}{\sigma_i} \leq \frac{\sigma_1}{\sigma_i} \epsilon_M \equiv \kappa_i \epsilon_M. \quad (3.18)$$

As σ_i gets smaller, κ_i becomes larger and $\tilde{\sigma}_i$ has fewer and fewer accurate digits. By the time κ_i is of the order ϵ_M^{-1} , the perturbed singular value $\tilde{\sigma}_i$ will be totally inaccurate. Otherwise put, to expect any accuracy in $\tilde{\sigma}_i$ we must have

$$\kappa_i \leq \epsilon_M^{-1}.$$

Turning now to the cross-product method, let us now assume that we compute $A = X^T X$ exactly and then round it to get \hat{A} (we could hardly do better than that

in statement 1 of Algorithm 3.1). We can show as above that the eigenvalues of A and \hat{A} satisfy

$$\frac{|\hat{\lambda}_i - \lambda_i|}{\lambda_i} \leq \frac{\lambda_1}{\lambda_i} \epsilon_M.$$

This bound has the same interpretation as (3.18). In particular we must have $\lambda_1/\lambda_i \leq \epsilon_M^{-1}$ to expect any accuracy in $\hat{\lambda}_i$.

We now note that $\lambda_i = \sigma_i^2$. Consequently the condition $\lambda_1/\lambda_i \leq \epsilon_M^{-1}$ is equivalent to

$$\kappa_i \leq \epsilon_M^{-\frac{1}{2}}.$$

This says that the range of singular values that the cross-product algorithm can compute with any accuracy has been severely restricted. For example, if $\epsilon_M = 10^{-16}$ then any singular value satisfying $\kappa_i \geq 10^8$ will have no significant figures when represented as an eigenvalue of A — even though it may have as many as eight significant figures as a singular value of X . Thus we should not expect the cross-product algorithm to work well if the ratio of the largest to the smallest singular value of X is too large.

An example will illustrate the above considerations.

Example 3.5. A matrix X was generated in the form $X = UDV^T$, where U and V are random orthogonal matrices and

$$D = \text{diag}(1, 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}).$$

The singular values of X and the square roots of the eigenvalues of $A = X^T X$ were computed with the following results.

$\tilde{\sigma}$	$\sqrt{\lambda}$
1.000000000000000e+00	1.000000000000000e+00
1.0000000000000001e-02	1.0000000000000072e-02
1.000000000000487e-04	1.000000001992989e-04
1.000000000014129e-06	1.000009260909809e-06
9.99999970046386e-09	1.177076334615305e-08

Since $\epsilon_M \cong 10^{-16}$, it is seen that the errors increase as predicted by the analysis above.

Although it is not illustrated by this example, when $\kappa_p \geq \epsilon_M^{-\frac{1}{2}}$, it is possible for $\hat{\lambda}_p$ to be negative, so that the cross-product algorithm produces an imaginary singular value.

Inaccurate right singular vectors

We can perform a similar analysis to show that the right singular vectors of X can be less accurately represented in A . Specifically, consider the effects of rounding X on the left singular vector v_i . From (3.16), we see that the perturbation of v_i depends on the separation of σ_i from its nearest neighbor—call it σ_j . If \tilde{v}_i denotes the singular vector of the rounded matrix, then

$$\sin \angle(v_i, \tilde{v}_i) \lesssim \frac{\sqrt{2}\|E\|_2}{|\sigma_i - \sigma_j|}.$$

Since $\|E\|_2 \leq \sigma_1 \epsilon_M$, we have

$$\sin \angle(v_i, \tilde{v}_i) \lesssim \frac{\sqrt{2}\epsilon_M}{|\kappa_i^{-1} - \kappa_j^{-1}|}. \quad (3.19)$$

A similar analysis for A shows that

$$\sin \angle(v_i, \hat{v}_i) \lesssim \frac{\sqrt{2}\epsilon_M}{|(\lambda_i/\lambda_1) - (\lambda_j/\lambda_1)|},$$

where \hat{v}_i is the i th eigenvector of \hat{A} . Noting once again that $\lambda_i = \sigma_i^2$, we can rewrite this bound in the form

$$\sin \angle(v_i, \tilde{v}_i) \lesssim \frac{\sqrt{2}\epsilon_M}{|\kappa_i^{-1} - \kappa_j^{-1}| |\kappa_i^{-1} + \kappa_j^{-1}|}. \quad (3.20)$$

If we compare (3.19) and (3.20), we see that when κ_i and κ_j are large, (3.20) has the smaller denominator. The singular vector in A is more sensitive to rounding errors than the same vector in X .

It is a curious and useful fact that the singular vector corresponding to the smallest singular value can be quite accurate, even when the singular value itself has no accuracy. Specifically, suppose $\sigma_1 \cong \sigma_{p-1}$ while σ_p is small. Then with $i = p$ and $j = p-1$, we find that κ_p is much greater than κ_{p-1} , so that right-hand sides of the bounds (3.19) and (3.20) are approximately

$$\sqrt{2}\kappa_{p-1}\epsilon_M \quad \text{and} \quad \sqrt{2}\kappa_p^2\epsilon_M.$$

Since $\kappa_{p-1} \cong 1$, the vector v_p is well determined by both algorithms, although the small singular value σ_p is ill determined. The common sense of these bounds is that the subspace corresponding to the large singular values is well determined and hence its orthogonal complement—the space spanned by v_p —is also well determined. Note, however, that as σ_{p-1} drifts away from σ_1 , the bound for the cross-product algorithm deteriorates more rapidly.

Inaccurate left singular vectors

The left singular vectors computed by our algorithm will also be less accurate, although the cause of the loss of accuracy is different. Let \hat{v}_i be the i th right singular vector computed by the cross-product algorithm, and assume that

$$\hat{v}_i = v_i + e, \quad \|e\|_2 = \epsilon. \quad (3.21)$$

The cross-product algorithm computes the i th left singular vector by forming $w_i = X\hat{v}_i$ and normalizing it. Now an elementary rounding error analysis shows that the computed w_i — call it \tilde{w}_i — satisfies

$$\tilde{w}_i = (X + E)\hat{v}_i,$$

where

$$\|E\|_2 \leq \gamma\sigma_1\epsilon_M.$$

Here γ is a constant that depends on the dimensions of X .

Now $\tilde{w}_i = Xv_i + Xe_i + E\hat{v}_i \equiv w_i + Xe_i + Ev_i$. Hence

$$\|\tilde{w}_i - w_i\|_2 \lesssim \|X\|_2\|e_i\| + \|E\|_2\|v_i\| \leq \sigma_1(\epsilon + \gamma\epsilon_M).$$

Since $w_i = \sigma_i u_i$, we have $\|w_i\|_2 = \sigma_i$. Hence the normwise relative error in \tilde{w}_i is

$$\frac{\|\tilde{w}_i - w_i\|_2}{\|w_i\|_2} \lesssim \kappa_i(\epsilon + \gamma\epsilon_M).$$

Thus, as previously, w_i will be computed inaccurately if σ_i is small. Normalizing it will not restore lost accuracy.

We have not specified the size of ϵ in our analysis because the technique for computing left singular vectors is sometimes used independently with accurate v_i ($\epsilon \approx \epsilon_M$). However, if v_i is computed from the cross-product algorithm, ϵ can be quite large and have a devastating effect. For example, if $\epsilon_M = 10^{-16}$ and $\kappa_i = 10^6$, then ϵ in (3.21) will be about 10^{-4} , and the computed \tilde{w}_i will have no accuracy at all.

Assessment of the algorithm

The above analysis shows that the cross-product algorithm is not suitable as a general-purpose algorithm. Perhaps more important, the bounds from the analysis suggest when the algorithm can be used. For example, if κ_p is not too large, then the computed singular value $\tilde{\sigma}_p$ may be satisfactory. Even when κ_p is large, the right singular vectors \tilde{v}_p can, as we have seen, be very accurate. For more see the notes and references.

3.3. REDUCTION TO BIDIAGONAL FORM

The classical algorithm for computing the singular value decomposition is a variant of the QR algorithm. Like its counterparts for general and symmetric matrices, its efficient implementation requires that the matrix be reduced to a simpler form. In this case the form is an upper bidiagonal form, which we will write as

$$\begin{pmatrix} d_1 & e_1 & & & \\ & d_2 & e_2 & & \\ & & d_3 & e_3 & \\ & & & \ddots & \ddots \\ & & & & d_{p-1} & e_{p-1} \\ & & & & & d_p \end{pmatrix}. \quad (3.22)$$

The reduction

As usual let X be an $n \times p$ matrix with $n \geq p$. The reduction to bidiagonal form proceeds by alternately pre- and postmultiplying X by Householder transformations. For the first step, we determine a Householder H transformation to introduce zeros in the hatted elements in the following Wilkinson diagram:

$$\begin{pmatrix} X & X & X & X \\ \hat{X} & X & X & X \end{pmatrix}.$$

Then $H X$ has the form

$$\begin{pmatrix} X & X & \hat{X} & \hat{X} \\ 0 & X & X & X \end{pmatrix}. \quad (3.23)$$

We next choose a Householder transformation K such that $H X K$ has zeros in the hatted elements of (3.23) to give a matrix of the form

$$\begin{pmatrix} X & X & 0 & 0 \\ 0 & X & X & X \end{pmatrix}.$$

Given an $n \times p$ matrix X with $n \geq p$, *BiReduce* reduces X to the bidiagonal form (3.22). The transformations from the left are accumulated in the $n \times p$ matrix U , and the transformations from the right in the $p \times p$ matrix V . On return, the array X contains the generators of the Householder transformations used in the reduction.

1. *BiReduce*(X, d, e, U, V)
2. Reduce X
3. **for** $k = 1$ **to** p
4. **if** ($k < p$ or $n > p$)
5. *housegen*($X[k:n, k], a, d[k]$)
6. $X[k:n, k] = a$
7. $b^T = a^T * X[k:n, k+1:p]$
8. $X[k:n, k+1:p] = X[k:n, k+1:p] - a * b^T$
9. **end if**
10. **if** ($k < p-1$)
11. *housegen*($X[k, k+1:p], b^T, e[k]$)
12. $X[k, k+1:p] = b^T$
13. $a = X[k+1:n, k+1:p] * b$
14. $X[k+1:n, k+1:p] = X[k+1:n, k+1:p] - a * b^T$
15. **end if**
16. **end for** k
17. $e[p-1] = X[p-1, p]$
18. **if** ($n = p$) $d[p] = X[p, p]$; **fi**
19. Accumulate U
20. $U = \begin{pmatrix} I_p \\ 0_{n-p,p} \end{pmatrix}$
21. **for** $k = \min\{p, n-1\}$ **to** 1 **by** -1
22. $a = X[k:n, k]$
23. $b^T = a^T * U[k:n, k:p]$
24. $U[k:n, k:p] = U[k:n, k:p] - a * b^T$
25. **end for** k
26. Accumulate V
27. $V = I_p$
28. **for** $k = p-2$ **to** 1 **by** -1
29. $b^T = X[k, k+1:p]$
30. $a^T = b^T * V[k+1:p, k+1:p]$
31. $V[k+1:p, k+1:p] = V[k+1:p, k+1:p] - b * a^T$
32. **end for** k
33. **end BiReduce**

Algorithm 3.2: Reduction to bidiagonal form



The reduction proceeds recursively on the matrix $X[2:n, 2:p]$. Algorithm 3.2 implements this reduction. Here are some comments.

- As in Algorithm 2.2, Chapter 2, the algorithm does not accumulate the transformations as they are generated. Instead it saves the generating vectors in X and accumulates the transformations after the reduction beginning with the last transformations. This saves some operations.
- In deriving an operation count for the algorithm, it is convenient to divide the labor.

Algorithm 3.2 requires

$$\begin{aligned} 2np^2 - \frac{2}{3}p^3 \text{ flam} &\quad \text{for the reduction of } X, \\ 2np^2 - p^3 \text{ flam} &\quad \text{for the accumulation of } U, \\ p^3 \text{ flam} &\quad \text{for the accumulation of } V. \end{aligned}$$

Thus if $n \gg p$ the total work is about $4np^2$ flam. If $n = p$, it is about $\frac{8}{3}p^3$ flam.

- By accumulating only the first p columns of the transformations from the left, the algorithm sets the stage for the subsequent computation of the singular value factorization (3.3). If we accumulate the entire U , the operation count rises to $2n^2p - np^2$. When $n \gg p$ this part of the computation becomes overwhelmingly dominant.
- If we require the effect of the full matrix U , we can save the transformations that are generated and apply them whenever we need to form a product of the form Ux .
- The algorithm is backward stable in the usual sense.
- The first column of the matrix V is e_1 . This fact will prove important in §3.4.

Real bidiagonal form

When X is complex, the tridiagonal form will also be complex. Our subsequent manipulations of this form will use plane rotations, and the operation count will be reduced if the rotations are real. We can insure this by transforming the complex bidiagonal form into a real form.

Algorithm 3.3, which is an analogue of Algorithm 1.2, performs the reduction to real form. It alternately scales the rows and columns of the bidiagonal matrix, the row scalings making the d_i real and the column scalings making the e_i real.

3.4. THE QR ALGORITHM

In this subsection we are going to show how the QR algorithm can be used to compute the singular values of a bidiagonal matrix B . What distinguishes the bidiagonal singular value problem from the other problems we have considered is that small relative changes in the elements of B make only small relative changes in the singular

This algorithm takes the output of Algorithm 3.2 and transforms the bidiagonal matrix to real bidiagonal form. The transformations are accumulated in U and V .

```

1.  for  $k = 1$  to  $p$ 
2.     $a = |d[k]|$ 
3.    if ( $a \neq 0$ )
4.       $\nu = \bar{d}[k]/a$ 
5.       $d[k] = a$ 
6.      if ( $k \neq p$ )  $e[k] = \nu * e[k]$ ; fi
7.       $U[:, k] = \bar{\nu} * U[:, k]$ 
8.    end if
9.    if ( $k = p$ ) leave for  $k$ ; fi
10.    $a = |e[k]|$ 
11.   if ( $a \neq 0$ )
12.      $\nu = \bar{e}[k]/a$ 
13.      $e[k] = a$ 
14.      $d[k+1] = \nu * d[k+1]$ 
15.      $V[:, k+1] = \nu * V[:, k+1]$ 
16.   end if
17. end for  $k$ 
```

Algorithm 3.3: Complex bidiagonal to real bidiagonal form



values of B . This means that we must take care that our convergence and deflation procedures do not introduce large relative errors.

We begin with the basic QR step and then show how to compute the shift. We then treat the problem of when the superdiagonal elements of B can be considered negligible. The solution provides us with the wherewithal to describe a back search in the spirit of Algorithm 3.3, Chapter 2.

Throughout this subsection:

B will be an upper bidiagonal matrix represented in the form

$$B = \begin{pmatrix} d_1 & e_1 & & & \\ & d_2 & e_2 & & \\ & & d_3 & e_3 & \\ & & & \ddots & \ddots \\ & & & & d_{p-1} & e_{p-1} \\ & & & & & d_p \end{pmatrix}. \quad (3.24)$$

The bidiagonal QR step

The matrix $C = B^T B$ is easily seen to be symmetric and tridiagonal. Consequently, we could apply the tridiagonal QR algorithm to $B^T B$ to find the squares of the singular values of B . Unfortunately, this procedure is a variant of the cross-product algorithm, which we have shown to have substantial drawbacks. However, we can mimic a QR step on $B^T B$ while working with B . The procedure goes as follows.

1. Given a shift σ , determine a plane rotation P_{12} such that the $(1, 2)$ -element of $(C - \sigma^2 I)P_{12}$ is zero.
2. Determine an orthogonal matrix U and an orthogonal matrix V whose first column is a multiple of e_1 such that $\hat{B} = U^T B P_{12} V$ is bidiagonal.

Since $\hat{C} = \hat{B}^T \hat{B} = V^T P_{12}^T B^T B P_{12} V = V^T P_{12}^T C P_{12} V$, the reasoning following (3.6), Chapter 2, shows that \hat{C} is the matrix that would have resulted from applying one step of the QR algorithm with shift σ^2 to C . Since the repeated application of this algorithm (with suitable shifts) will cause $c_{p-1,p}$ to converge to zero, we may expect to see e_{p-1} also converge to zero, in which case the problem deflates.

To implement this procedure, we need the first column of C . Since B is bidiagonal, this column is easily computed:

$$\begin{pmatrix} d_1^2 \\ d_1 e_1 \end{pmatrix}.$$

Thus the cosine and sine of the initial rotation P_{12} are determined by the numbers $d_1^2 - \sigma^2$ and $d_1 e_1$. Since these numbers are composed of products, some scaling should be done to prevent overflow and underflow.

We could use by Algorithm 3.2 to reduce $P_{12}B$ to bidiagonal form. However, the fact that B is bidiagonal can be used to simplify the algorithm. Specifically, the matrix BP_{12} has the form

$$\begin{pmatrix} X & X & 0 & 0 & 0 \\ \widehat{X} & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{pmatrix}.$$

Note that BP bulges out of its bidiagonal form at the hatted $(2, 1)$ -element. We can remove this bulge by premultiplying by a plane rotation U_{12} in the $(1, 2)$ -plane to get the matrix

$$\begin{pmatrix} X & X & \widehat{X} & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{pmatrix}.$$

The bulge has now moved to the $(1, 3)$ -element. We can eliminate it by postmultiplying by a rotation V_{23} in the $(2, 3)$ -plane:

$$\begin{pmatrix} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & \widehat{X} & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{pmatrix}.$$

The bulge is now at the $(3, 2)$ element. We can continue in this manner until the bulge is chased out the bottom of the matrix. The entire procedure is illustrated in Figure 3.1.

In practice the QR step will not be applied to the entire matrix. Algorithm 3.4 implements a QR step between rows $i1$ and $i2$. Here are some comments.

- The accumulation of the transformations accounts for most of the work, at least when p is large. Specifically:

Algorithm 3.4 requires

$$2(i2 - i1)p \text{ flrot.}$$

- The step is backward stable in the usual sense.

Given an upper bidiagonal matrix represented as in (3.24) and a shift σ , $BiQR$ performs a QR step between rows $i1$ and $i2$. The transformations are accumulated in U and V .

```

1.  BiQR( $d, e, \sigma, i1, i2, U, V$ )
2.   $scl = \max\{|d[i1]|, |e[i1]|, \sigma\}$ 
3.   $d1 = d[i1]/scl; e1 = e[i1]/scl; sig = \sigma/scl$ 
4.   $f = (d1 + sig)*(d1 - sig); g = d1*e1$ 
5.  for  $k = i1$  to  $i2-1$ 
6.    rotgen( $f, g, c, s$ )
7.    if ( $k \neq i1$ )  $e[k-1] = f$ ; fi
8.     $f = c*d[k] + s*e[k]$ 
9.     $e[k] = c*e[k] - s*d[k]$ 
10.    $g = s*d[k+1]$ 
11.    $d[k+1] = c*d[k+1]$ 
12.   rotapp( $c, s, V[:, k], V[:, k+1]$ )
13.   rotgen( $f, g, c, s$ )
14.    $d[k] = f$ 
15.    $f = c*e[k] + s*d[k+1]$ 
16.    $d[k+1] = c*d[k+1] - s*e[k]$ 
17.   if ( $k < i2-1$ )
18.      $g = s*e[k+1]$ 
19.      $e[k+1] = c*e[k+1]$ 
20.   end if
21.   rotapp( $c, s, U[:, k], U[:, k+1]$ )
22. end for  $k$ 
23.  $e[i2-1] = f$ 
24. end gqr
```

Algorithm 3.4: Bidiagonal QR step

$$\begin{array}{l}
 \left(\begin{array}{ccccc} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \xrightarrow{P_{12}} \left(\begin{array}{ccccc} X & X & 0 & 0 & 0 \\ \hat{X} & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \xrightarrow{U_{12}} \left(\begin{array}{ccccc} X & X & \hat{X} & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \\
 \\
 \xrightarrow{V_{23}} \left(\begin{array}{ccccc} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & \hat{X} & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \xrightarrow{U_{23}} \left(\begin{array}{ccccc} X & X & 0 & 0 & 0 \\ 0 & X & X & \hat{X} & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \xrightarrow{V_{34}} \left(\begin{array}{ccccc} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & \hat{X} & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \\
 \\
 \xrightarrow{U_{34}} \left(\begin{array}{ccccc} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & \hat{X} \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right) \xrightarrow{V_{45}} \left(\begin{array}{ccccc} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & \hat{X} & X \end{array} \right) \xrightarrow{U_{45}} \left(\begin{array}{ccccc} X & X & 0 & 0 & 0 \\ 0 & X & X & 0 & 0 \\ 0 & 0 & X & X & 0 \\ 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & X \end{array} \right)
 \end{array}$$

Figure 3.1: A bidiagonal QR step

Computing the shift

The shift σ in Algorithm 3.4 is generally computed from the trailing principal submatrix $B[i1-1:i1, i1-1:i1]$. A reasonable choice is to take σ to be the smallest singular value of this matrix. The following theorem gives an explicit formula for this singular value.

Theorem 3.6. *Let*

$$T = \begin{pmatrix} p & r \\ 0 & q \end{pmatrix}.$$

Then the largest singular value of T is given by the formula

$$\sigma_{\max} = \frac{\sqrt{(p+q)^2 + r^2} + \sqrt{(p-q)^2 + r^2}}{2}. \quad (3.25)$$

If $T \neq 0$, then the smallest singular value is given by

$$\sigma_{\min} = \frac{|pq|}{\sigma_{\max}}. \quad (3.26)$$

Proof. The eigenvalues of the matrix

$$T^T T = \begin{pmatrix} p^2 & pr \\ pr & q^2 + r^2 \end{pmatrix}$$

are the squares of the singular values of T . The characteristic equation of this matrix is

$$\lambda^2 - (p^2 + q^2 + r^2)\lambda + (pq)^2 = 0. \quad (3.27)$$

If we set

$$\sigma_{\min} = \frac{|\sqrt{(p+q)^2 + r^2} - \sqrt{(p-q)^2 + r^2}|}{2},$$

then it is easily verified that

$$\sigma_{\max}^2 \sigma_{\min}^2 = (pq)^2 \quad \text{and} \quad \sigma_{\max}^2 + \sigma_{\min}^2 = p^2 + q^2 + r^2.$$

Thus σ_{\max}^2 and σ_{\min}^2 , which is not greater than σ_{\max}^2 , are roots of (3.27), and hence σ_{\max} and σ_{\min} are the largest and smallest singular values of T . The formula (3.26) follows from the fact that $\sigma_{\max}\sigma_{\min} = |pq|$. ■

Here are three comments on this theorem.

- If at least one of p , q , and r are zero, the singular values of T can be computed by inspection.
- The formulas yield approximations to the singular values that have high relative accuracy.
- In practice the formulas must be scaled and rearranged to avoid overflows and harmful underflows. For more see the notes and references.

Negligible superdiagonal elements

The formulas in Theorem 3.6 not only produce singular values to high relative accuracy, but they are themselves insensitive to small relative perturbations in the quantities p , q , and r . This is a special case of an important fact about the singular values of bidiagonal matrices.

Small relative perturbations in the elements of a bidiagonal matrix cause only small relative perturbations in its singular values.

Unfortunately, in testing convergence, we must decide when we can set a superdiagonal element to zero. Since this is an absolute change, there is a real danger of compromising the relative accuracy of the singular values. Fortunately, the following result can be used to tell when we can safely ignore superdiagonals.

Let the quantities τ_i be defined as follows.

1. $\tau_n = d_n$
 2. **for** $i = n-1$ **to** 1 **by** -1
 3. $\tau_i = |d_i| * (\tau_{i+1}/(\tau_{i+1} + |e_i|))$
 4. **end for**
- (3.28)

It can be shown that

$$\min_{k \leq i \leq n} \tau_i = \frac{1}{\|B[k:n, k:n]^{-1}\|_\infty^{-1}}.$$

However, the real significance of the number τ_i is contained in the following theorem.

Theorem 3.7 (Demmel–Kahan). *Let B be a bidiagonal matrix in the form (3.24), and for fixed j let \hat{B} be the matrix obtained by setting e_j to zero. Let σ_i and $\hat{\sigma}_i$ denote the singular values of B and \hat{B} arranged in descending order. Let*

$$\epsilon = \frac{1}{\sqrt{2}} \frac{|e_j|}{\tau_{j+1}},$$

and consider the intervals

$$\mathcal{I}_k = \{\sigma : -\epsilon \leq \ln(\sigma/\hat{\sigma}_k) \leq \epsilon\}.$$

If σ_i lies in the connected union of m intervals \mathcal{I}_k that are disjoint from the other intervals, then

$$-m\epsilon \leq \ln(\hat{\sigma}_i/\sigma_i) \leq m\epsilon. \quad (3.29)$$

To see what (3.29) means, take the exponential to get

$$e^{-m\epsilon} \leq \hat{\sigma}_i/\sigma_i \leq e^{m\epsilon}.$$

If $m\epsilon$ is small, then $e^{\pm m\epsilon} \approx 1 \pm m\epsilon$. It follows that

$$\frac{|\sigma_i - \hat{\sigma}_i|}{\sigma_i} \lesssim m\epsilon = \frac{m}{\sqrt{2}} \frac{|e_j|}{\tau_{j+1}}. \quad (3.30)$$

Thus the result of setting e_j to zero can introduce a relative error of no greater than $m\epsilon$ in any singular value of B . If the singular value in question is isolated and ϵ is sufficiently small, then the relative error is bounded by ϵ .

Back searching

We can combine (3.28) and the results of Theorem 3.7 to get Algorithm 3.5, which searches back for a block in which the problem deflates. It generates the τ_i from Theorem 3.7 as it proceeds backward through the matrix. Note that whenever the problem deflates (i.e., $i1 = i2$), the sequence of τ_i is restarted. By (3.30) the parameter tol is approximately the relative error we can expect from setting the current superdiagonal to zero. A reasonable value is the rounding unit.

Final comments

The algorithmic pieces we have created in this and the previous section can be assembled to produce a reliable algorithm for computing the singular values of a general matrix. Although no theorems can be proved, it works well on graded matrices, provided the grading is downward. Nonetheless, there are further embellishments that can improve the algorithm. Some of these are discussed in the notes and references.

This algorithm takes a bidiagonal matrix B represented as in (3.24), a tolerance tol , and an index ℓ ($1 < \ell \leq p$) and determines indices $i1, i2 \leq \ell$ such that one of the following conditions holds.

1. $1 \leq i1 < i2 \leq \ell$, in which case the matrix deflates at rows $i1$ and $i2$.
 2. $i1 = i2 = \ell$, in which case all the superdiagonal elements of B in rows 1 through $\ell-1$ are zero.
- ```

1. BdBacksearch(d, e, tol, ℓ, i1, i2)
2. $i1 = i2 = \ell$
3. while ($i1 > 1$)
4. if ($i1 = i2$)
5. $\tau = d[i1]$
6. else
7. if ($e[i1-1] \neq 0$)
8. $\tau = d[i1]*(\tau/(\tau + |e[i1-1]|))$
9. end if
10. end if
11. if ($|e[i1-1]| \leq tol*\tau$)
12. $e[i1-1] = 0$
13. if ($i1 = i2$)
14. $i1 = i2 = i1-1$
15. else
16. return
17. end if
18. else
19. $i1 = i1-1$
20. end if
21. end while
22. end BdBacksearch

```

**Algorithm 3.5:** Back searching a bidiagonal matrix

Let  $X \in \mathbb{R}^{n \times p}$ , where  $n \gg p$ . This algorithm computes the singular value factorization  $X = U\Sigma V^H$ .

1. Compute the QR factorization  $X = QR$
2. Compute the singular value decomposition  $R = W\Sigma V^H$
3.  $U = QW$

### Algorithm 3.6: Hybrid QR-SVD algorithm

---

## 3.5. A HYBRID QRD-SVD ALGORITHM

The algorithm sketched in the last subsection is reasonably efficient when  $n$  is not much greater than  $p$ . When  $n \gg p$ , however, one pays a high price for the accumulation of plane rotations in  $U$ . An alternative is to first compute the QR factorization of  $A$ , compute the singular value decomposition of  $R$ , and multiply the left singular vectors into  $Q$ .

Algorithm 3.6 implements this scheme. Here are three comments.

- When  $n \gg p$ , statement 2, which requires  $O(p^3)$  operations, contributes only negligibly to the total operation count, which is  $O(np^2)$ . Specifically, the QR decomposition of  $X$  can be computed in  $2np^2$  fflam while the generation of  $U$  requires  $np^2$  fflam, bringing the total operation count to  $3np^2$  fflam.
- The computations can be arranged so that  $Q$  and then  $U$  overwrites  $X$ .
- The matrix  $Q$  will generally be computed as the product of the Householder transformations used to reduce  $X$  to triangular form (§I:4.1.2). If instead, we store the transformations themselves and retain the matrix  $W$ , we save  $np^2$  fflam and in the bargain have an implicit basis for both the columns space of  $X$  and its orthogonal complement. The price to be paid is that one must access that basis through subprograms that manipulate the Householder transformations.

## 3.6. NOTES AND REFERENCES

### The singular value decomposition

The singular value decomposition was established independently and almost simultaneously by Beltrami [19, 1873] and Jordan [137, 1874]. For more historical material see [263] and §I:1.4.5. Missing proofs in §3.1 can be found in §I:1.4.

### Perturbation theory

Parts of Theorem 3.4 were presented in [269, Theorem I.4.6], where the expansion of the singular value is carried out to the second order. For ordinary singular values the second-order term makes no essential difference, but it shows that small singular

values tend to increase in size under perturbation. The derivation of the first-order bounds given here is new.

### The cross-product algorithm

The cross-product algorithm is widely and unknowingly used by statisticians and data analysts. For example, given a data matrix  $X$ , a standard transformation is to subtract column means, normalize, and form a cross-product matrix  $C$  called the correlation matrix. Subsequent analyses are cast in terms of  $C$ . Whenever an analysis involves the eigenvalues and eigenvectors of  $C$ , it amounts to an implicit use of the cross-product algorithm. Fortunately, much data is noisy enough that the errors introduced by passing to  $C$  are insignificant—at least when computations are performed in double precision.

The analysis of the cross-product algorithm appears to be new, although partial analyses and instructive examples may be found in most texts on numerical linear algebra—often with warnings never to form the cross-product matrix. The lesson of the analysis given here is that one should not condemn but discern.

### Reduction to bidiagonal form and the QR algorithm

The reduction to bidiagonal form is due to Golub and Kahan [88, 1965].

It is a curious fact that the QR algorithm for the singular values of bidiagonal matrices was first derived by Golub [86, 1968] without reference to the QR algorithm. The derivation given here may be found in the Handbook contribution [91].

Much of the material on the implementation of the QR algorithm for bidiagonal matrices is taken from an important paper by Demmel and Kahan [58]. Owing to limitations of space we cannot do the paper full justice in this chapter. But here are some additional details.

- **Additional tests for negligibility.** The authors present a forward recurrence that computes the quantity  $\|B[k:n, k:n]^{-1}\|_1^{-1}$ . From this, it is possible to compute a lower bound  $\underline{\sigma}$  for the smallest singular value. This permits a cheap test for negligibility of the  $e_i$ . Namely,  $e_i$  is negligible if  $|e_i| \leq \text{tol} \cdot \underline{\sigma}$ , where tol is the relative error required in the singular values.
- **Zero shifted QR.** If the shift in the QR algorithm is zero, the computations can be arranged so that all the singular values retain high relative accuracy. This algorithm is used whenever the relative accuracy  $\text{tol} \cdot \underline{\sigma}$  required of the smallest singular value is less than the error  $\bar{\sigma} \epsilon_M$  that can be expected from the shifted QR algorithm (here  $\bar{\sigma}$  is an estimate of the largest singular value).
- **Bottom to top QR.** Our implementation of the shifted bidiagonal QR algorithm proceeds from the top of the matrix to the bottom, which is fine for matrices graded downward. For matrices that are graded upward there is a variant of both the shifted and unshifted algorithm that proceed from bottom to top. If the current block extends from  $i1$  to  $i2$ , a natural, though not foolproof, method for choosing among these meth-

ods is to use the top-down method if  $d[i1] \geq d[i2]$  and otherwise use the bottom-up method.

---

Code based on these ideas may be found in the LAPACK routine `xBDSQR`.

### The QRD-SVD algorithm

The idea of saving work by computing a singular value decomposition from a QR decomposition is due to Chan [34, 35]. Higham [115] has shown that row and column pivoting in the computation of the QR decomposition improves the accuracy of the singular values.

### The differential qd algorithm

When only singular values are desired, Fernando and Parlett [73] have devised a new variant of the shifted qd algorithm that calculates them efficiently to high relative accuracy. The algorithm has been implemented in the LAPACK routine `SLASQ1`.

### Divide-and-conquer algorithms

Gu and Eisenstat [101] give a divide-and-conquer algorithm for the bidiagonal singular value problem. Their paper also contains a summary of other divide-and-conquer algorithms with extensive references.

### Downdating a singular value decomposition

The singular value decomposition is useful in applications where  $X$  is a data matrix with each row representing an item of data. An interesting problem is to update the singular value decomposition after a row has been added or removed. The latter problem is called downdating and comes in several varieties, depending on what parts of the singular value decomposition we have at hand. Gu and Eisenstat [102] give three algorithms for the downdating problem, as well as references to previous work.

### Jacobi methods

Jacobi methods (see p. 202) can be generalized in two ways to find the singular value decomposition of a matrix  $X$ . The *two-sided method*, due to Kogbetliantz [149, 1955], works with square  $X$ . A typical iteration step goes as follows. For some  $i < j$ , let  $U$  and  $V$  be plane rotations in the  $(i, j)$ -plane such that the  $(i, j)$ - and  $(j, i)$ -elements of  $U^T X V$  are zero — i.e., solve the singular value problem for the embedded  $2 \times 2$  matrix

$$\begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix},$$

applying the transformations to rows and columns  $i$  and  $j$  of  $X$ . As with the Jacobi algorithm, this transformation increases the sum of squares of the diagonal elements of  $X$  and decreases the sum of squares of the off-diagonal elements by the same amount.

An interesting and work-saving observation about this method is that if  $X$  is upper triangular and elements are eliminated rowwise, then the result of a complete sweep is a lower triangular matrix.

The *one-sided method* does not require  $X$  to be square. Choose two columns  $i$  and  $j$  of  $X$  and determine a rotation  $V$  such that the  $i$ th and  $j$ th columns of  $XV$  are orthogonal. It is easy to see that the rotation  $V$  is the same as the rotation that would be obtained by applying a Jacobi step to  $X^T X$  with pivot indices  $i$  and  $j$ . There is an analogous two-sided method in which rotations are also applied from the left.

The techniques that are used to analyze the Jacobi method generally carry over to the variants for the singular value decomposition. For more see [8, 37, 67, 75, 107, 108, 109, 203].

## 4. THE SYMMETRIC POSITIVE DEFINITE (S/PD) GENERALIZED EIGENVALUE PROBLEM

A generalized eigenvalue problem  $Ax = \lambda Bx$  is said to be symmetric positive definite (S/PD) if  $A$  is symmetric and  $B$  is positive definite. The problem arises naturally in many applications. Mathematically it is a generalization of the symmetric eigenvalue problem and can, in fact, be reduced to an equivalent symmetric eigenvalue problem. Although this reduction, which is due to Wilkinson, does not yield an unconditionally stable algorithm, it is the best we have, and the purpose of this subsection is to describe the problem and its solution.

We begin by developing the theory of the S/PD generalized eigenvalue problem. In §4.2 we present Wilkinson's algorithm, including some modifications to handle the case where  $B$  is ill conditioned with respect to inversion. As usual in this chapter, all matrices are real.

### 4.1. THEORY

Here we establish the basic theory of S/PD pencils.

#### The fundamental theorem

We begin with a definition.

**Definition 4.1.** *Let  $A$  be symmetric and  $B$  be positive definite. Then the pencil  $(A, B)$  is said to be SYMMETRIC POSITIVE DEFINITE, abbreviated S/PD.*

The slash in S/PD is there to stress that two matrices are involved—the symmetric matrix  $A$  and the positive definite matrix  $B$  (which is by definition symmetric).

In transforming S/PD pencils it is natural to use symmetry-preserving congruences, i.e., transformations of the form  $(X^T A X, X^T B X)$ , where  $X$  is nonsingular. The following theorem shows that we can diagonalize  $A$  and  $B$  by such transformations.

**Theorem 4.2.** Let  $(A, B)$  be an S/PD pencil. Then there is a nonsingular matrix  $X$  such that

$$X^T AX = D_A = \text{diag}(\alpha_1, \dots, \alpha_n)$$

and

$$X^T BX = D_B = \text{diag}(\beta_1, \dots, \beta_n),$$

where  $\beta_i \geq 0$ .

**Proof.** Let  $Z$  be any matrix of order  $n$  satisfying

$$B = Z^T Z.$$

For example  $Z$  could be the Cholesky factor of  $B$ . Since  $B$  is positive definite and hence nonsingular,  $Z$  is also nonsingular.

The matrix  $Z^{-T} AZ^{-1}$  is symmetric. Let

$$U^T (Z^{-T} AZ^{-1}) U = D_A$$

be its spectral decomposition. Set

$$X = Z^{-1} U.$$

Then by construction  $X^T AX$  is diagonal. Moreover,

$$X^T BX = U^T (Z^{-T} B Z^{-1}) U = U^T I U = I$$

is also diagonal. ■

There are four comments to be made about this theorem.

- Since  $X^T BX = I$ , it is natural to say that  $X$  is  $B$ -orthogonal. In this language, we have proved that  $A$  can be diagonalized by a  $B$ -orthogonal transformation. When  $B = I$ , the theorem says  $A$  can be diagonalized by an orthogonal transformation. In this sense the S/PD eigenvalue problem is a generalization of the symmetric eigenvalue problem.
- If we write  $AX = X^{-T} D_A$  and denote the  $i$ th column of  $X$  by  $x_i$ , then

$$Ax_i = AX\mathbf{e}_i = X^{-T} D_A \mathbf{e}_i = \alpha_i X^{-T} \mathbf{e}_i.$$

Similarly,  $Bx_i = \beta_i X^{-T} \mathbf{e}_i$ . It follows that

$$\beta_i Ax_i = \alpha_i Bx_i,$$

so that  $\lambda_i = \alpha_i / \beta_i$  is an eigenvalue of  $(A, B)$  and  $x_i$  is the corresponding eigenvector. Thus, our theorem states that the S/PD pencil  $(A, B)$  has real eigenvalues and a complete set of  $B$ -orthonormal eigenvectors.

- The proof of the theorem gives eigenvectors  $x_i$  normalized so that  $x_i^T B x_i = 1$ . But other normalizations are possible.
- Whatever the normalization, the numbers  $\beta_i$  are positive. This implies that an S/PD pencil cannot have infinite eigenvalues.

### Condition of eigenvalues

Let  $(\lambda, x)$  be a simple eigenvalue of the S/PD pencil  $(A, B)$  with  $\|x\|_2 = 1$ . Let  $\alpha = x^T Ax$  and  $\beta = x^T Bx$  so that  $\langle \alpha, \beta \rangle$  is a projective representation of  $\lambda$ . If  $(A+E, B+F)$  is a perturbed pencil, then by Theorem 4.12, Chapter 2, for  $E$  and  $F$  sufficiently small there is an eigenvalue  $\tilde{\lambda}$  of the perturbed pencil such that

$$\chi(\lambda, \tilde{\lambda}) \lesssim \frac{\sqrt{\|E\|_F^2 + \|F\|_F^2}}{\sqrt{\alpha^2 + \beta^2}}. \quad (4.1)$$

Thus  $(\alpha^2 + \beta^2)^{-\frac{1}{2}}$  is a condition number for the eigenvalue  $\lambda$  in the chordal metric.

The bound (4.1) shows that the eigenvalues of an S/PD problem, unlike the eigenvalues of a Hermitian matrix, can be ill conditioned. This happens when both  $\alpha_i$  and  $\beta_i$  are small. For example, in the problem

$$\begin{pmatrix} 2 & 0 \\ 0 & 10^{-5} \end{pmatrix} x = \lambda \begin{pmatrix} 1 & 0 \\ 0 & 10^{-5} \end{pmatrix} x,$$

the eigenvalue 2 is generally insensitive to perturbations bounded by  $10^{-5}$ , while perturbations of the same size can move the eigenvalue 1 anywhere in the interval  $[0, \infty)$ . (However, see Example 4.13, Chapter 2.)

On the other hand, the well-conditioned eigenvalues of an S/PD pencil are not necessarily determined to high relative accuracy. This is generally true of the small eigenvalues, as Example 3.9, Chapter 1, shows. However, well-conditioned large eigenvalues can also be ill determined in the relative sense. For example, consider the problem

$$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} x = \lambda \begin{pmatrix} 1 & 0 \\ 0 & 10^{-5} \end{pmatrix} x.$$

If the  $(2, 2)$ -element of  $B$  is perturbed to  $2 \cdot 10^{-5}$ , the eigenvalue  $10^5$  becomes  $\frac{1}{2} \cdot 10^5$ .

## 4.2. WILKINSON'S ALGORITHM

In this subsection we consider an algorithm due Wilkinson for solving the S/PD eigenvalue problem. We begin with the algorithm itself and then consider modifications to improve its performance in difficult cases.

### The algorithm

Wilkinson's algorithm is based on the proof of Theorem 4.2. Let  $(A, B)$  be an S/PD pencil and let

$$B = R^T R$$

be the Cholesky decomposition of  $B$  — i.e.,  $R$  is upper triangular with positive diagonals (see §A.5). Let

$$R^{-T} A R^{-1} = U \Lambda U^T$$

Given an S/PD pencil  $(A, B)$  of order  $n$ , this algorithm produces a matrix  $X$  such that

$$X^TAX = \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n) \quad \text{and} \quad X^TBX = I.$$

1. Compute the Cholesky factor  $R$  of  $B$
2.  $C = R^{-T}AR^{-1}$
3. Compute the spectral decomposition  $U\Lambda U^T$  of  $C$
4.  $X = R^{-1}U$ .

**Algorithm 4.1:** Solution of the S/PD generalized eigenvalue problem

---

be the spectral decomposition of  $R^{-T}AR^{-1}$ . Then the matrix

$$X = R^{-1}U$$

diagonalizes the pencil.

Algorithm 4.1 implements this scheme. Here are some comments on this algorithm.

- Only the upper triangles of  $A$  and  $B$  need be stored. See page 159.
- The Cholesky factor  $R$  can be computed by a variant of Gaussian elimination — e.g., Algorithm I:3.2.1. If  $B$  is not needed later, its upper triangle may be overwritten by  $R$ .
- We will treat the computation of  $C$  in a moment. If  $A$  is not needed later, its upper triangle may be overwritten by the upper triangle of  $C$ .
- The columns  $x_j$  of  $X$  may be formed from the columns  $u_j$  of  $U$  by solving the triangular systems

$$Rx_i = u_i.$$

The calculations can be arranged so that  $X$  overwrites  $U$  [cf. (I:2.2.7)].

- The exact operation count for the algorithm depends on how long the method used to compute the spectral decomposition of  $C$  takes, which is determined in part by  $C$ . If the QR algorithm (§1) or the divide-and-conquer algorithm (§2.2) is used, the operation count is  $O(n^3)$  with a modest order constant.

### Computation of $R^{-T}AR^{-1}$

The computation of  $C = R^{-T}AR^{-1}$  is tricky enough to deserve separate treatment. In fact, there is more than one algorithm. The one given here proceeds by bordering.

Specifically, consider the partitioned product

$$\begin{pmatrix} C & c \\ c^T & \gamma \end{pmatrix} = \begin{pmatrix} R & r \\ 0 & \rho \end{pmatrix}^{-T} \begin{pmatrix} A & a \\ a^T & \alpha \end{pmatrix} \begin{pmatrix} R & r \\ 0 & \rho \end{pmatrix}^{-1}. \quad (4.2)$$

Assuming that we already know  $C = R^{-T}AR^{-1}$ , we will show how to compute  $c$  and  $\gamma$ . This will enable us to solve our problem by successively computing the leading principal minors of  $R^{-T}AR^{-1}$  of order  $1, 2, \dots, n$ .

We begin by rewriting (4.2) in the form

$$\begin{pmatrix} C & c \\ c^T & \gamma \end{pmatrix} = \begin{pmatrix} R^{-T} & 0 \\ -\rho^{-1}r^T R^{-T} & \rho^{-1} \end{pmatrix} \begin{pmatrix} A & a \\ a^T & \alpha \end{pmatrix} \begin{pmatrix} R^{-1} & -\rho^{-1}R^{-1}r \\ 0 & \rho^{-1} \end{pmatrix}.$$

Then by direct computation we find that

$$c = \rho^{-1}(R^{-T}a - Cr)$$

and

$$\gamma = \rho^{-2}(r^TCr - r^T R^{-T}a - a^T R^{-1}r + \alpha).$$

Thus we can compute  $\gamma$  and  $c$  by the following algorithm.

1.  $c = R^{-T}a$
  2.  $\beta = r^Tc$
  3.  $c = c - Cr$
  4.  $\gamma = (\alpha - \beta - r^Tc)/\rho^2$
  5.  $c = \rho^{-1}c$
- (4.3)

Algorithm 4.2 implements this scheme. Three comments.

- The algorithm takes  $\frac{1}{2}n^3$  flam.
- The arrangement of the computations in (4.3) may seem artificial, but they are designed to allow Algorithm 4.2 to overwrite  $A$  by  $C$ . Simply replace all references to  $C$  by references to  $A$ .
- The bulk of the work in this algorithm is performed in computing the product of the current principal submatrix with a vector. For a scheme in which the bulk of the work is contained in a rank-one update see the LAPACK routine xSYGS2.

### Ill-conditioned $B$

Wilkinson's algorithm is effectively a way of working with  $B^{-1}A$  while preserving symmetry. A defect of this approach is that if  $B$  is ill conditioned with respect to inversion—i.e., if  $\|B\|_2\|B^{-1}\|_2$  is large (see §I:3.3)—then  $B^{-1}A$  will be calculated inaccurately. There is no complete cure for this problem, since the large eigenvalues of  $(A, B)$  are ill determined. A partial cure—one which keeps the errors confined to the larger eigenvalues—is to arrange the calculations so that the final matrix  $C$  is

Given a symmetric matrix  $A$  and a nonsingular upper triangular matrix  $R$ , this algorithm computes  $C = R^{-T}AR^{-1}$ .

1.  $C[1, 1] = A[1, 1]/(R[1, 1]*R[1, 1])$
2. **for**  $k = 2$  **to**  $n$
3.     Solve the system  $R[1:k-1, 1:k-1]^T C[1:k-1, k] = A[1:k-1, k]$
4.      $\beta = R[1:k-1, k]^T * C[1:k-1, k]$
5.      $C[1:k-1, k] = C[1:k-1, k] - C[1:k-1, 1:k-1]^T * R[1:k-1, k]$
6.      $C[k, k] = A[k, k] - \beta - R[1:k-1, k]^T * C[1:k-1, k]$
7.      $C[k, k] = C[k, k]/(R[k, k]*R[k, k])$
8.      $C[1:k-1, k] = R[k, k]^{-1} * C[1:k-1, k]$
9.      $C[k, 1:k-1] = C[1:k-1, k]^T$
10. **end for**  $k$

**Algorithm 4.2:** The computation of  $C = R^{-T}AR^{-1}$

---

graded downward. As we have observed (p. 108 ff) the reduction to tridiagonal form and the QR algorithm tend to perform well on this kind of problem.

One way to produce a graded structure is to compute the spectral decomposition  $B = VMV^T$ , where the diagonal elements of  $M$  are ordered so that  $\mu_1 \leq \mu_2 \leq \dots \leq \mu_n$ . If we define  $C = M^{-\frac{1}{2}}V^TAVM^{-\frac{1}{2}}$  and compute its spectral decomposition  $C = U\Lambda U^T$ , then the diagonal elements of  $\Lambda$  are eigenvalues of  $(A, B)$ , and the corresponding eigenvectors are the columns of  $X = VM^{-\frac{1}{2}}U$ .

A less expensive alternative is to use a pivoted Cholesky decomposition of  $B$  (see §A.5.3). This produces a decomposition of the form

$$P^TBP = R^TR,$$

where  $P$  is a permutation matrix and  $R$  is graded downward. If we set

$$C = F R^{-T} P^T A P R F,$$

where  $F$  is the cross matrix, then  $C$  will be graded downward. If we then compute the spectral decomposition  $C = U\Lambda U^T$ , then the diagonal elements of  $\Lambda$  are eigenvalues of  $(A, B)$ , and the corresponding eigenvectors are the columns of  $X = PR^{-1}FU$ .

### 4.3. NOTES AND REFERENCES

#### Perturbation theory

We have derived the perturbation theory for S/PD pencils from the perturbation theory for general pencils. As might be expected by analogy with the Hermitian eigenproblem, much more can be said about the eigenvalues and eigenvectors of S/PD pencils.

This theory is surveyed in [269, §IV.3]. Recently, Li and Mathias [163] have revisited this theory and, starting from a different perspective, have given significantly improved bounds.

### Wilkinson's algorithm

Wilkinson presented his algorithm in *The Algebraic Eigenvalue Problem* [300, §§68–72, Ch. 5]. Wilkinson mentions, but does not derive, an algorithm for problems of the form  $ABx = \lambda x$ , where  $A$  is symmetric and  $B$  is positive definite. Code for both problems may be found in the Handbook [170].

Wilkinson discusses the problem of ill-conditioned  $B$  and suggests that using the spectral decomposition of  $B$  instead of its Cholesky decomposition to reduce the problem will tend to preserve the small eigenvalues, although he does not mention the necessary sorting of the eigenvalues.

Chandrasekaran [36] has noted that the large eigenvalues and corresponding eigenvectors of  $C$  (obtained from the spectral decomposition of  $B$ ) are well determined. He shows how to deflate these well-determined eigenpairs from the original pencil, giving a smaller pencil containing the remaining eigenpairs. Since each deflation step requires the computation of a spectral decomposition, the algorithm potentially requires  $O(n^4)$  operations. It seems, however, that only a very few deflation steps are required.

The LAPACK routines `xxxGST` reduce the S/PD eigenproblem  $Ax = \lambda Bx$  to standard form. They also reduce the eigenproblem  $BAx = \lambda x$ , where  $A$  is symmetric and  $B$  is positive definite.

### Band matrices

Wilkinson's method results in a full symmetric eigenvalue problem, even when  $A$  and  $B$  are banded. To take advantage of the banded structure Wilkinson notes that Gaussian elimination on  $A - \lambda B$  gives a count of the eigenvalues less than  $\lambda$ , just as in Algorithm 2.4. Thus the eigenvalues of  $(A, B)$  can be found by interval bisections. Eigenvectors can then be computed by the inverse power method suitably generalized.

### The definite generalized eigenvalue problem

A Hermitian pencil  $(A, B)$  is *definite* if

$$\gamma(A, B) \stackrel{\text{def}}{=} \min_{\|x\|_2=1} \sqrt{(x^H A x)^2 + (x^H B x)^2} > 0. \quad (4.4)$$

A definite pencil can in principle be reduced to an S/PD pencil. Specifically, consider the set

$$\mathcal{F}(A, B) = \{x^H(A + iB)x : \|x\|_2 = 1\}.$$

The set  $\mathcal{F}(A, G)$  is called the *field of values* of  $A + iB$ , and it can be shown to be convex. If we define

$$A_\theta = A \cos \theta - B \sin \theta,$$

$$B_\theta = A \sin \theta + B \cos \theta,$$

then the field of values  $\mathcal{F}_\theta$  of  $A_\theta + B_\theta$  is just the set  $\mathcal{F}$  rotated by the angle theta. Now if  $\gamma(A, B) > 0$ , then  $\mathcal{F}(A, B)$  does not contain the origin; and since it is convex, it can be rotated to lie in the upper half of the complex plane, which is equivalent to  $B_\theta$  being positive definite. For more on definite pairs see [269, §§VI.1.3, VI.3]. [It is a curious fact that the above argument does not work for  $n = 2$  and real symmetric  $A$  and  $B$  when the vector  $x$  in (4.4) ranges only over  $\mathbb{R}^n$ .]

The above argument is nonconstructive in that it does not give a means for computing a suitable angle  $\theta$ . However, Crawford and Moon [44] describe an iterative algorithm for computing a  $\theta$  such that  $B_\theta$  is positive definite (provided, of course, that the original pencil was definite). Each step requires a partial Cholesky factorization of the current  $B_\theta$ , and the algorithm can be  $O(n^4)$ . For an implementation see [43]. For another approach see [40].

### The generalized singular value and CS decompositions

Let  $X \in \mathbb{R}^{n \times p}$  and  $Y \in \mathbb{R}^{m \times p}$ , where  $m, n \geq p$ . Then there are orthogonal matrices  $U_X$  and  $U_Y$  and a nonsingular matrix  $W$  such that

$$X = U_X \begin{pmatrix} \Sigma_X \\ 0 \end{pmatrix} W^T \quad \text{and} \quad Y = U_Y \begin{pmatrix} \Sigma_Y \\ 0 \end{pmatrix} W^T, \quad (4.5)$$

where  $\Sigma_X$  and  $\Sigma_Y$  are diagonal. This *generalized singular value decomposition* was proposed, with  $W^{-1}$  on the right-hand sides, by Van Loan [283, 1976] and in the above form by Paige and Saunders [202, 1981]. It is easily seen that

$$X^T X = W \Sigma_X^2 W^T \quad \text{and} \quad Y^T Y = W \Sigma_Y^2 W^T,$$

so that the generalized singular value decomposition is related to the generalized eigenvalue problem in the cross-product matrices just as the singular value decomposition is related to the eigenvalue problem for the cross-product matrix. For much the same reasons as given in §3.2, a computational approach to the generalized singular value decomposition through cross-product matrices is deprecated.

The generalized singular value decomposition is related to the CS decomposition of an orthonormal matrix (see [202, 258] or §I:1.4.6). Specifically, let

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} Q_X \\ Q_Y \end{pmatrix} R \quad (4.6)$$

be a QR factorization of  $(X^T \ Y^T)^T$ . Then the CS decomposition exhibits orthogonal matrices  $U_X$ ,  $U_Y$ , and  $V$ , such that

$$Q_X = U_X \begin{pmatrix} \Sigma_X \\ 0 \end{pmatrix} V^T \quad \text{and} \quad Q_Y = U_Y \begin{pmatrix} \Sigma_Y \\ 0 \end{pmatrix} V^T, \quad (4.7)$$

where  $\Sigma_X$  and  $\Sigma_Y$  are diagonal matrices with  $\Sigma_X^2 + \Sigma_Y^2 = I$ . (This last relation implies that the diagonal elements of  $\Sigma_X$  and  $\Sigma_Y$  can be interpreted as cosines and sines, whence the name CS decomposition.) If we combine (4.6) and (4.7) and set  $X^T = V^T R$ , the result is the generalized singular value decomposition (4.5).

Algorithms for computing the CS decomposition and hence the singular value decomposition have been proposed by Stewart [259, 260] and Van Loan [284]. Paige [200] has given a method that works directly through  $X$  and  $Y$ . The LAPACK routine xGGSVD uses Van Loan's approach.

*This page intentionally left blank*

# 4

---

## EIGENSPACES AND THEIR APPROXIMATION

The first half of this book has been concerned with dense matrices whose elements can be stored in the main memory of a computer. We now shift focus to problems that are so large that storage limitations become a problem. This problem has two aspects.

First, if the order of the matrix in question is sufficiently large, we will not be able to store all its elements. This limitation is not as burdensome as it might appear. Many large matrices arising in applications are sparse; that is, their zero elements greatly outnumber their nonzero elements. With appropriate data structures we can get away with storing only the latter. Unfortunately, these data structures restrict the operations we can perform on the matrix, and, in consequence, most algorithms for large sparse eigenvalue problems confine themselves to matrix-vector multiplications or perhaps the computation of an LU factorization.

Second, the eigenvectors of a sparse matrix are in general not sparse. It follows that if the matrix is large enough we cannot afford to store its entire eigensystem. Instead we must be satisfied with computing some subset of its eigenpairs. The space spanned by a set of eigenvectors is an example of an eigenspace, and it turns out that algorithms for large matrices are best explained in terms of eigenspaces. Consequently, this chapter is devoted to the basic facts about eigenspaces and their approximations.

We begin with the basic algebraic theory of eigenspaces, and in §2 we present the analytic theory — residual analysis and perturbation bounds. Because eigenspaces, like eigenvectors, cannot be computed directly, most algorithms for large problems proceed by generating a sequence of subspaces that contain increasingly accurate approximations to the desired eigenvectors. In §3 we treat one of the most popular ways of generating subspaces — the Krylov sequence — and in §4 we analyze the Rayleigh-Ritz method for extracting individual eigenvectors from a subspace.

Although all of the results of this chapter apply to matrices of any size, our chief concern is with large eigenproblems. Therefore:

*Unless otherwise specified  $A$  will denote a matrix of order  $n$ . It will often help to think of  $n$  as being very large.*

## 1. EIGENSPACES

In this subsection we are going to develop the theory of subspaces that remain invariant when acted on by a matrix. We have already noted that these invariant subspaces — or eigenspaces as we will call them — are useful in the derivation and analysis of algorithms for large eigenvalue problems. They are also important because one can use them to isolate ill-behaved groups of eigenvalues. For example, a matrix that does not have a complete system of eigenvectors may instead have a complete system of eigenspaces of low dimension corresponding to groups of defective eigenvalues.

In the first subsection we will present the basic definitions and properties. In §1.2 we will consider simple eigenspaces — the natural generalization of a simple eigenvector.

### 1.1. DEFINITIONS

We have already observed (p. 2) that the subspace spanned by an eigenvector is invariant under multiplication by its matrix. The notion of an eigenspace is a generalization of this observation.

**Definition 1.1.** *Let  $A$  be of order  $n$  and let  $\mathcal{X}$  be a subspace of  $\mathbb{C}^n$ . Then  $\mathcal{X}$  is an EIGENSPACE OR INVARIANT SUBSPACE OF  $A$  if*

$$A\mathcal{X} \equiv \{Ax : x \in \mathcal{X}\} \subset \mathcal{X}.$$

The space spanned by an eigenvector is a trivial but important example of an eigenspace. A less trivial example occurs in the proof of the existence of real Schur decompositions (Theorem 3.1, Chapter 2). There we showed that if  $(\lambda, x) \equiv (\lambda, y + iz)$  is a complex eigenpair of a real matrix  $A$ , then for some real  $2 \times 2$  matrix  $L$ ,

$$A(y \ z) = (y \ z)L. \tag{1.1}$$

Moreover, the eigenvalues of  $L$  are  $\lambda$  and  $\bar{\lambda}$ . Since the column space of  $(y \ z)L$  is a subset of the column space of  $(y \ z)$ , the space  $\mathcal{X} = \mathcal{R}[(y \ z)]$  is an eigenspace of  $A$ .

This construction illustrates the utility of eigenspaces. The information contained in the set of complex eigenpairs  $(\lambda, x)$  and  $(\bar{\lambda}, \bar{x})$  is captured more economically by the relation (1.1), which consists entirely of real quantities. Eigenspaces can be used to bring together combinations of eigenvalues and eigenvectors that would be difficult to treat separately.

### Eigenbases

The definition of eigenspace was cast in terms of a subspace of  $\mathbb{C}^n$ , while the expression (1.1) concerns a specific basis for the subspace in question. In computational practice we will always have to work with a basis. The following theorem summarizes the properties of a basis for an eigenspace.

**Theorem 1.2.** *Let  $\mathcal{X}$  be an eigenspace of  $A$  and let  $X$  be a basis for  $\mathcal{X}$ . Then there is a unique matrix  $L$  such that*

$$AX = XL.$$

*The matrix  $L$  is given by*

$$L = X^T AX,$$

*where  $X^T$  is a left inverse of  $X$  — i.e., a matrix satisfying  $X^T X = I$ .*

*If  $(\lambda, x)$  is an eigenpair of  $A$  with  $x \in \mathcal{X}$ , then  $(\lambda, X^T x)$  is an eigenpair of  $L$ . Conversely, if  $(\lambda, u)$  is an eigenpair of  $L$ , then  $(\lambda, X u)$  is an eigenpair of  $A$ .*

**Proof.** Let

$$X = (x_1 \ \cdots \ x_k) \quad \text{and} \quad Y = AX = (y_1 \ \cdots \ y_k)$$

be partitioned by columns. Since  $y_i \in \mathcal{X}$  and  $X$  is a basis for  $\mathcal{X}$ ,  $y_i$  can be expressed uniquely as a linear combination of the columns of  $X$ ; i.e.,

$$y_i = X \ell_i,$$

for some unique vector  $\ell_i$ . If we set  $L = (\ell_1 \ \cdots \ \ell_k)$ , then  $AX = XL$ , as required. If  $X^T$  is a left inverse of  $X$ , then

$$L = X^T X L = X^T A X.$$

Now let  $(\lambda, x)$  be an eigenpair of  $A$  with  $x \in \mathcal{X}$ . Then there is a unique vector  $u$  such that  $x = Xu$ . However,  $u = X^T x$ . Hence

$$\lambda x = Ax = AXu = XLu,$$

and the relation  $Lu = \lambda u$  follows on multiplying by  $X^T$ .

Conversely, if  $Lu = \lambda u$ , then

$$A(Xu) = (AX)u = X(Lu) = \lambda(Xu),$$

so that  $(\lambda, Xu)$  is an eigenpair of  $A$ . ■

We say that the matrix  $L$  is *the representation of  $A$  on the eigenspace  $\mathcal{X}$  with respect to the basis  $X$* . The theorem shows that  $L$  is independent of the choice of left inverse in the formula  $L = X^T AX$ .

The theorem also shows that there is a one-one correspondence between eigenvectors lying in  $\mathcal{X}$  and eigenvectors of  $L$ . In particular  $L$  has a complete set of eigenvectors if and only if the eigenvectors of  $A$  lying in  $\mathcal{X}$  span  $\mathcal{X}$ . If  $L$  does not have a complete system of eigenvectors, the relation between the eigensystem of  $A$  and that of  $L$  can be very complicated. As we will see, however, for the important case of a simple eigenspace (Definition 1.7), the relation is straightforward.

The existence of a unique representation  $L$  associated with a basis  $X$  for an eigenspace  $\mathcal{X}$  justifies the following extension of the notion of eigenpair.

**Definition 1.3.** Let  $A$  be of order  $n$ . For  $X \in \mathbb{C}^{n \times k}$  and  $L \in \mathbb{C}^{k \times k}$ , we say that  $(L, X)$  is an EIGENPAIR OF ORDER  $k$  or RIGHT EIGENPAIR OF ORDER  $k$  of  $A$  if

1.  $X$  is of full rank,
2.  $AX = XL$ .

We call the matrix  $X$  an EIGENBASIS and the matrix  $L$  an EIGENBLOCK. If  $X$  is orthonormal, we say that the eigenpair  $(L, X)$  is orthonormal.

If  $Y \in \mathbb{C}^{n \times k}$  has linearly independent columns and  $Y^H A = LY^H$ , we say that  $(L, Y)$  is a LEFT EIGENPAIR of  $A$ .

Note that if  $(L, Y)$  is a left eigenpair of  $A$ , then  $\mathcal{Y} = \mathcal{R}(Y)$  is a *left eigenspace* of  $A$  in the sense that

$$A^H \mathcal{Y} \subset \mathcal{Y}.$$

The following theorem shows how eigenpairs transform under change of basis and similarities. Its proof, which is simple and instructive, is left as an exercise.

**Theorem 1.4.** Let  $(L, X)$  be an eigenpair of  $A$ . If  $U$  is nonsingular, then the pair  $(U^{-1}LU, UX)$  is also an eigenpair of  $A$ . If  $W$  is nonsingular, then  $(L, W^{-1}X)$  is an eigenpair of  $W^{-1}AW$ .

An important consequence of this theorem is that the eigenvalues of the representation of an eigenspace with respect to a basis are independent of the choice of the basis. Consequently, we can speak of *the* eigenvalues of  $A$  associated with an eigenspace.

### Existence of eigenspaces

Given that every eigenspace has a unique set of eigenvalues, it is natural to ask if any set of eigenvalues has a corresponding eigenspace. The following theorem shows that indeed it has.

**Theorem 1.5.** Let  $\mathcal{L} = \{\lambda_1, \dots, \lambda_k\} \subset \Lambda(A)$  be a multisubset of the eigenvalues of  $A$ . Then there is an eigenspace  $\mathcal{X}$  of  $A$  whose eigenvalues are  $\lambda_1, \dots, \lambda_k$ .

**Proof.** Let

$$A(U_1 \ U_2) = (U_1 \ U_2) \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

be a partitioned Schur decomposition of  $A$  in which  $T_{11}$  is of order  $k$  and has the members of  $\mathcal{L}$  on its diagonal. Computing the first column of this relation, we get

$$AU_1 = U_1 T_{11}.$$

Hence the column space of  $U_1$  is an eigenspace of  $A$  whose eigenvalues are the diagonal elements of the triangular matrix  $T_{11}$  — i.e., the members of  $\mathcal{L}$ . ■

Thus a matrix  $A$  of order  $n$  with distinct eigenvalues has  $2^n$  distinct eigenspaces — one corresponding to each subset of  $\Lambda(A)$ . Unfortunately, if  $A$  has multiple eigenvalues, the eigenspaces of  $A$  are not as easily described. We will return to this point in the next subsection.

### Deflation

In establishing the existence of the real Schur form (Theorem 3.1, Chapter 2) we essentially used the eigenspace corresponding to a pair of complex conjugate eigenvalues to deflate the matrix in question. This is a special case of a general technique. The proof of the following theorem is left as an exercise.

**Theorem 1.6.** *Let  $(L, X)$  be an orthonormal eigenpair of  $A$  and let  $(X \ Y)$  be unitary. Then*

$$\begin{pmatrix} X^H \\ Y^H \end{pmatrix} A \begin{pmatrix} X & Y \end{pmatrix} = \begin{pmatrix} L & H \\ 0 & M \end{pmatrix}, \quad (1.2)$$

where

$$L = X^H A X, \quad M = Y^H A Y, \quad \text{and} \quad H = X^H A X.$$

Here are three comments on this theorem.

- The similarity transformation (1.2) of  $A$  deflates the problem in the usual sense. The spectrum of  $A$  is the union of spectra of  $L$  and  $M$ , which, for computational purposes, can be treated as separate eigenvalue problems.
- The relation (1.2) implies that

$$Y^H A = M Y^H.$$

Thus  $(M, Y)$  is a left eigenpair of  $A$ . Otherwise put:

If  $\mathcal{X}$  is an eigenspace of  $A$  and  $\mathcal{Y}$  is the orthogonal complement of  $\mathcal{X}$ , then  $\mathcal{Y}$  is a left eigenspace of  $A$  corresponding to the eigenvalues not associated with  $\mathcal{X}$ . We call  $\mathcal{Y}$  the COMPLEMENTARY LEFT EIGENSPACE OF  $\mathcal{X}$ .

- The construction of Theorem 1.2 is not the most general way to deflate an eigenspace. Specifically, if  $X$  is any basis for an eigenspace of  $A$  and  $(X \ Y)$  is nonsingular, then

$$(X \ Y)^{-1} A (X \ Y) = \begin{pmatrix} L & H \\ 0 & M \end{pmatrix}$$

for some matrices  $L$ ,  $M$ , and  $H$ . For numerical reasons, however, it is preferable to use unitary transformations to deflate an eigenspace. See the discussion on page 10.

## 1.2. SIMPLE EIGENSPACES

### Definition

We have seen (Theorem 1.5) that if  $\mathcal{L}$  is a subset of the spectrum of  $A$ , then there is an eigenspace  $\mathcal{X}$  of  $A$  whose eigenvalues are the members of  $\mathcal{L}$ . Unfortunately, this eigenspace need not be unique. For example, if  $\lambda$  is a double eigenvalue of  $A$  that is not defective, then it has two linearly independent eigenvectors, and any nonzero vector in the plane spanned by these two vectors spans a one-dimensional eigenspace corresponding to  $\lambda$ .

Computationally, it is impossible to pin down an object that is not unique. The above example suggests that the nonuniqueness in eigenspaces results from it not having all the copies of a multiple eigenvalue. Hence we make the following definition.

**Definition 1.7.** Let  $\mathcal{X}$  be an eigenspace of  $A$  with eigenvalues  $\mathcal{L}$ . Then  $\mathcal{X}$  is a SIMPLE EIGENSPACE of  $A$  if

$$\mathcal{L} \cap [\Lambda(A) \setminus \mathcal{L}] = \emptyset.$$

In other words, an eigenspace is simple if its eigenvalues are, counting multiplicities, disjoint from the other eigenvalues of  $A$ .

### Block diagonalization and spectral representations

We turn now to the existence of left eigenspaces corresponding to a right eigenspace. The problem is not as simple as the existence of left eigenvectors, since we cannot use determinantal arguments as in the second comment following Theorem 1.2, Chapter 1. We will approach the problem by way of block diagonalization.

**Theorem 1.8.** Let  $(L_1, X_1)$  be a simple orthonormal eigenpair of  $A$  and let  $(X_1, Y_2)$  be unitary so that

$$\begin{pmatrix} X_1^H \\ Y_2^H \end{pmatrix} A \begin{pmatrix} X_1 & Y_2 \end{pmatrix} = \begin{pmatrix} L_1 & H \\ 0 & L_2 \end{pmatrix} \quad (1.3)$$

(Theorem 1.2). Then there is a matrix  $Q$  satisfying the Sylvester equation

$$L_1 Q - Q L_2 = -H$$

such that if we set

$$X = (X_1 \ X_2) \quad \text{and} \quad Y = (Y_1 \ Y_2), \quad (1.4)$$

where

$$X_2 = Y_2 + X_1 Q \quad \text{and} \quad Y_1 = X_1 - Y_2 Q^H, \quad (1.5)$$

then

$$Y^H X = I \quad \text{and} \quad Y^H A X = \text{diag}(L_1, L_2). \quad (1.6)$$

**Proof.** Beginning with (1.3), apply Theorem 1.18, Chapter 1, to reduce  $A$  to block diagonal form. The formulas (1.5) result from accumulating the transformation in the successive reductions to block triangular form and block diagonal form. For example,

$$(X_1 \ X_2) \begin{pmatrix} I & Q \\ 0 & I \end{pmatrix} = (X_1 \ X_1 + X_2 Q),$$

which gives the formula for  $X_2$  in (1.5). ■

Here are some observations on this theorem.

- If we write the second equation in (1.6) in the form

$$A(X_1 \ X_2) = (X_1 \ X_2)\text{diag}(L_1, L_2),$$

we see that  $AX_2 = X_2L_2$ , so that  $(L_2, X_2)$  is an eigenpair of  $A$ . The eigenspace  $\mathcal{R}(X_2)$  is complementary to the eigenspace  $\mathcal{R}(X_1)$  in  $\mathbb{C}^n$ , and we call it the *complementary right eigenspace*. Likewise  $(L_1, Y_1)$  is a left eigenpair of  $A$ , so that  $\mathcal{R}(Y_1)$  is the *left eigenspace corresponding to  $\mathcal{R}(X_1)$* . Combining these facts with the observation that the orthogonal complement of  $\mathcal{R}(X_1)$  is the complementary left eigenspace of  $A$ , we have the following corollary.

**Corollary 1.9.** *Let  $\mathcal{X}$  be a simple eigenspace of  $A$ . Then  $\mathcal{X}$  has a corresponding left eigenspace and a complementary right eigenspace. The orthogonal complement of  $\mathcal{X}$  is the complementary left eigenspace.*

- If  $(\lambda, x)$  is a simple eigenpair of  $A$ , then there is a left eigenvector  $y$  corresponding to  $x$ , which can be normalized so that  $y^H x = 1$  [see (3.12), Chapter 1]. The fact that the eigenbasis  $Y_1$  for the left eigenspace corresponding to  $X_1$  is normalized so that  $Y_1^H X_1 = I$  generalizes this result. We say that the bases  $X_1$  and  $Y_1$  are *biorthogonal*. We will return to this point later when we consider canonical angles between subspaces (see Corollary 2.5).
- From (1.6) we can write

$$A = X_1 L_1 Y_1^H + X_2 L_2 Y_2^H. \quad (1.7)$$

We will call this formula a *spectral representation of  $A$* . Saying that (1.7) is a spectral representation of  $A$  is a convenient shorthand for (1.4) and (1.6).

The eigenbases in a spectral representation are not unique. If, for example,  $S$  is nonsingular, then we may replace  $X_1$  by  $X_1S$ ,  $Y_1$  by  $Y_1S^{-H}$ , and  $L_1$  by  $S^{-1}L_1S$ . In establishing the spectral representation we assumed that  $X_1$  and  $Y_2$  were orthonormal. We will call any such spectral representation a *standard representation*.

- Theorem 1.8 uses a single simple eigenspace to reduce a matrix to  $2 \times 2$  block diagonal form. However, an obvious induction shows that if we have  $k$  simple eigenspaces then we can reduce  $A$  to  $(k+1) \times (k+1)$  block diagonal form, in which the first

$k$  blocks are eigenblocks of the eigenspaces and the last eigenblock is the eigenblock of the complement of their union. In analogy with (1.7), there is a corresponding spectral representation of the form

$$A = \sum_{i=1}^{k+1} X_i L_i Y_i^H.$$

### Uniqueness of simple eigenspaces

We turn now to the question of whether a simple eigenspace is uniquely determined by its eigenvalues. Although it seems obvious that it is, the result is not easy to prove.

It will be sufficient to prove the uniqueness for a simple eigenspace corresponding to a single eigenvalue. For if a simple eigenspace has more than one eigenvalue, it can be decomposed by diagonalization into eigenspaces corresponding to its individual eigenvalues. If the latter are unique, then so is the original space.

Let  $\mathcal{X}$  be a simple eigenspace corresponding to a single eigenvalue, which without loss of generality we may assume to be zero. Then we can diagonalize  $A$  as in Theorem 1.8 to get the matrix  $\text{diag}(L, M)$ . Since  $L$  has only zero eigenvalues it is nilpotent (Theorem 1.13, Chapter 1). Since  $M$  cannot have zero eigenvalues, it is nonsingular.

Now let  $\hat{\mathcal{X}}$  be a second eigenspace corresponding to the eigenvalue zero. We can then use this eigenspace to diagonalize  $A$  to get  $\text{diag}(\hat{L}, \hat{M})$ , where again  $\hat{L}$  is nilpotent and  $\hat{M}$  is nonsingular. Since  $\text{diag}(L, M)$  and  $\text{diag}(\hat{L}, \hat{M})$  are both similar to  $A$ , they are similar to each other. Let  $X^{-1}\text{diag}(L, M)X = \text{diag}(\hat{L}, \hat{M})$ . Let  $Y = X^{-1}$ , and write the similarity transformation in the partitioned form

$$\begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix} \begin{pmatrix} L & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} \hat{L} & 0 \\ 0 & \hat{M} \end{pmatrix}.$$

Let  $k$  be an integer such that  $L^k = \hat{L}^k = 0$ . Then

$$\begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & M^k \end{pmatrix} \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & \hat{M}^k \end{pmatrix}. \quad (1.8)$$

Computing the  $(2, 2)$ -block of this relation, we get

$$Y_{22} M^k X_{22} = \hat{M}^k.$$

Since  $\hat{M}^k$  is nonsingular, so are  $X_{22}$  and  $Y_{22}$ . Now compute the  $(1, 2)$ -block of (1.8) to get

$$Y_{12} M^k X_{22} = 0.$$

Since  $M^k X_{22}$  is nonsingular,  $Y_{21} = 0$ . Similarly  $Y_{12} = 0$ . Thus  $X$  is block diagonal, and the spaces  $\mathcal{X}$  and  $\hat{\mathcal{X}}$  are the same.

### 1.3. NOTES AND REFERENCES

#### Eigenspaces

Although the term “eigenspace” can be found in Kato’s survey of perturbation theory [146], such spaces are usually called invariant subspaces. A trivial reason for preferring “eigenspace” over “invariant subspace” is that the latter need not be truly invariant under multiplication by  $A$  — an extreme example is  $A = 0$ , which collapses any subspace into  $\{0\}$ . A more substantial reason is that we can use the same terminology for generalized eigenvalue problems, where there is no one matrix to operate on the subspace.

The generalization of the notion of eigenpair as well as the terms “eigenbasis” and “eigenblock” (Definition 1.3) are new.

#### Spectral projections

From the spectral representation

$$A = X_1 L_1 Y_1^H + X_2 L_2 Y_2^H$$

we can form the matrices  $P_i = X_i Y_i^H$ . It is easily verified that  $P_i$  is idempotent — i.e.,  $P_i^2 = P_i$ . Such matrices project a vector onto its column space along its null space, and for that reason the  $P_i$  are called spectral projectors. Spectral projectors can be used to decompose a matrix without transforming it. For example, it is easy to verify that

$$A = P_1 AP_1 + P_2 AP_2,$$

an equation that corresponds to our reduction to block diagonal form. For more on spectral projections see [146].

#### Uniqueness of simple eigenspaces

That a simple eigenspace is determined by its eigenvalues seems to be taken for granted in the literature on matrix computations. The proof given here is new.

#### The resolvent

Our approach to eigenspaces through block triangularization and diagonalization reflects the way we compute with eigenspaces. An alternative approach is to use the resolvent defined by

$$R(\zeta, A) = (A - \zeta I)^{-1}.$$

For example, it can be shown that the spectral projection corresponding to a set  $\mathcal{L}$  of eigenvalues is given by the complex integral

$$-\frac{1}{2\pi i} \int_{\Gamma_{\mathcal{L}}} R(\zeta) d\zeta,$$

where  $\Gamma_{\mathcal{L}}$  is a closed path in the complex plane that contains  $\mathcal{L}$  and no other eigenvalues. For a systematic development of this approach see [146].

## 2. PERTURBATION THEORY

In this subsection we will treat the perturbation of eigenspaces and their associated eigenblocks. Much of this theory parallels the perturbation theory in §3, Chapter 1; however, because of the important role played by residuals in algorithms for large problems, we will first present the analysis of residuals and then go onto perturbation theory proper. Specifically, we begin with preliminary material on the canonical angles between subspaces. We then turn to the analysis of residuals. In §2.3 we use these residual bounds to derive a perturbation theory for simple eigenpairs. Finally, in §2.4 we present some residual bounds for Hermitian matrices.

### 2.1. CANONICAL ANGLES

We have defined the angle between two nonzero vectors as the angle whose cosine is  $|x^H y| / (\|x\|_2 \|y\|_2)$  [see (3.11), Chapter 1]. In this subsection we extend this definition to subspaces in  $\mathbb{C}^n$ . To avoid certain degenerate cases, we will restrict the dimensions of our subspaces to be not greater than  $\frac{n}{2}$ . For a more complete development of this subject see §I:4.5.

#### Definitions

Let  $\mathcal{X}$  and  $\mathcal{Y}$  be subspaces of the same dimension. Let  $X$  and  $Y$  be orthonormal bases for  $\mathcal{X}$  and  $\mathcal{Y}$ , and consider the matrix  $C = Y^H X$ . We have

$$\|C\|_2 \leq \|X\|_2 \|Y\|_2 = 1.$$

Hence all the singular values of  $C$  lie in the interval  $[0, 1]$  and can be regarded as cosines of angles. These angles are independent of the choice of bases for  $\mathcal{X}$  and  $\mathcal{Y}$ . For if  $\hat{X}$  and  $\hat{Y}$  are other bases, we can write  $\hat{X} = XV$  and  $\hat{Y} = YW$ , where  $V = X^H \hat{X}$  and  $W = Y^H \hat{Y}$  are unitary. Hence  $\hat{C} = \hat{Y}^H \hat{X} = W^H CV$  and  $C$  are unitarily equivalent and have the same singular values. This justifies the following definition.

**Definition 2.1.** *Let  $\mathcal{X}$  and  $\mathcal{Y}$  be subspaces of  $\mathbb{C}^n$  of dimension  $p$  and let  $X$  and  $Y$  be orthonormal bases for  $\mathcal{X}$  and  $\mathcal{Y}$ . Then the CANONICAL ANGLES between  $\mathcal{X}$  and  $\mathcal{Y}$  are the numbers*

$$\theta_i = \cos^{-1} \gamma_i,$$

where  $\gamma_i$  are the singular values of  $Y^H X$ . We write  $\Theta(\mathcal{X}, \mathcal{Y})$  for the  $i$ th canonical angle in descending order and define

$$\Theta(\mathcal{X}, \mathcal{Y}) = \text{diag}(\theta_1, \dots, \theta_p).$$

We will also write  $\Theta(X, Y)$  for  $\Theta(\mathcal{X}, \mathcal{Y})$ .

Geometrically, the largest canonical angle has the following characterization:

$$\theta_1(\mathcal{X}, \mathcal{Y}) = \max_{\substack{x \in \mathcal{X} \\ x \neq 0}} \min_{\substack{y \in \mathcal{Y} \\ y \neq 0}} \angle(x, y).$$

Thus for each  $x$  in  $\mathcal{X}$  we look for the vector  $y$  in  $\mathcal{Y}$  that makes the smallest angle  $\theta(x)$  with  $x$ . The largest such angle  $\theta(x)$  is the largest canonical angle.

### Computing canonical angles

In principle we could compute canonical angles from the characterization in Definition 2.1. However, if the canonical angle is small, this procedure will give inaccurate results. Specifically, for small  $\theta$ ,  $\cos \theta \cong 1 - \frac{1}{2}\theta^2$ . If  $\theta \leq 10^{-8}$ , then  $\cos \theta$  will evaluate to 1 in IEEE double-precision arithmetic, and we will conclude that  $\theta = 0$ . A cure for this problem is to compute the sine of the canonical angles. The following theorem, which is a generalization of Lemma 3.12, Chapter 1, provides the wherewithal.

**Theorem 2.2.** *Let  $X$  and  $Y$  be orthonormal bases for  $\mathcal{X}$  and  $\mathcal{Y}$ , and let  $Y_\perp$  be an orthonormal basis for the orthogonal complement of  $\mathcal{Y}$ . Then the singular values of  $Y_\perp^H X$  are the sines of the canonical angles between  $\mathcal{X}$  and  $\mathcal{Y}$ .*

**Proof.** Let

$$\begin{pmatrix} Y^H \\ Y_\perp^H \end{pmatrix} X = \begin{pmatrix} C \\ S \end{pmatrix}.$$

Since this matrix is orthonormal,

$$I = C^H C + S^H S.$$

Let  $V^H(C^H C)V = \Gamma^2$  be the spectral decomposition of  $C^H C$ . Since the eigenvalues of  $C^H C$  are the squares of the singular values of  $C$ , the diagonal elements of  $\Gamma$  are the cosines of the canonical angles between  $\mathcal{X}$  and  $\mathcal{Y}$ . But

$$I = V^H(C^H C + S^H S)V = \Gamma^2 + V^H(S^H S)V \equiv \Gamma^2 + \Sigma^2.$$

It follows that  $\Sigma^2$  is diagonal and its diagonal elements are  $1 - \gamma_i^2$ —the squares of the sines of the canonical angles. Hence the singular values of  $S = Y_\perp^H X$  are the sines of the canonical angles between  $\mathcal{X}$  and  $\mathcal{Y}$ . ■

In this chapter we will use Theorem 2.2 as a mathematical device for bounding canonical angles. But it also provides the means to compute the sines of canonical angles numerically. For details, see the notes and references.

### Subspaces of unequal dimensions

We will occasionally need to compare subspaces  $\mathcal{X}$  and  $\mathcal{Y}$  of differing dimensions. The statement that  $\mathcal{X}$  is near  $\mathcal{Y}$  makes sense only when  $\dim(\mathcal{X}) \leq \dim(\mathcal{Y})$ . For otherwise there would be a vector  $x$  in  $\mathcal{X}$  that is orthogonal to every vector in  $\mathcal{Y}$  and the

angle between that vector and  $\mathcal{Y}$  could only be  $\frac{\pi}{2}$ . When  $\dim(\mathcal{X}) < \dim(\mathcal{Y})$ , we can define canonical angles as in Definition 2.1. In this case, the number of canonical angles will be equal to the dimension of  $\mathcal{X}$ . These canonical angles can be calculated as in Theorem 2.2.

The following theorem contains a useful characterization of the angle between a vector and a subspace.

**Theorem 2.3.** *Let  $x$  be a vector with  $\|x\|_2 = 1$  and let  $\mathcal{Y}$  be a subspace. Then*

$$\sin \angle(x, \mathcal{Y}) = \min_{y \in \mathcal{Y}} \|x - y\|_2.$$

**Proof.** Let  $(Y \ Y_\perp)$  be unitary with  $\mathcal{R}(Y) = \mathcal{Y}$ . Let  $y \in \mathcal{Y}$  and consider the transformed vector

$$\begin{pmatrix} Y^H \\ Y_\perp^H \end{pmatrix} (x - y) = \begin{pmatrix} \hat{x} \\ \hat{x}_\perp \end{pmatrix} - \begin{pmatrix} \hat{y} \\ \hat{y}_\perp \end{pmatrix}.$$

In this new coordinate system we must have  $\hat{y}_\perp = 0$ . Hence

$$\|x - y\|_2 = \left\| \begin{pmatrix} \hat{x} - \hat{y} \\ \hat{x}_\perp \end{pmatrix} \right\|_2.$$

This norm is clearly minimized when  $\hat{y} = \hat{x}$ , and the norm at the minimum is  $\|\hat{x}_\perp\|_2$ , which, by Theorem 2.2, is  $\sin \angle(x, \mathcal{Y})$ . ■

### Combinations of subspaces

In Theorem 1.8 we expressed the left eigenspace corresponding to a simple eigenspace in the form  $X + YQ$ . We will meet such expressions later, where we will have to know the relations between the singular values of  $Q$  and  $X + YQ$ . The following theorem does the job.

**Theorem 2.4.** *Let  $X$  and  $Y$  be orthonormal matrices with  $X^H Y = 0$  and let  $Z = X + YQ$ . Let  $\sigma_i$  and  $\varsigma_i$  denote the nonzero singular values of  $Z$  and  $Q$  arranged in descending order. Let  $\theta_i$  denote the nonzero canonical angles between  $\mathcal{R}(X)$  and  $\mathcal{R}(Z)$ , also arranged in descending order. Then*

$$\sigma_i = \sec \theta_i$$

and

$$\varsigma_i = \tan \theta_i.$$

**Proof.** The squares of the nonzero singular values of  $Q$  are the nonsingular eigenvalues of  $Q^H Q$ , and the squares of the nonzero singular values of  $Z$  are the nonzero eigenvalues of

$$Z^H Z = I + Q^H Q.$$

Hence

$$\sigma_i^2 = 1 + \varsigma_i^2. \quad (2.1)$$

By definition the cosines of the canonical angles between  $\mathcal{R}(X)$  and  $\mathcal{R}(Z)$  are the singular values of  $X^H \hat{Z}$ , where  $\hat{Z}$  is any orthonormal basis for  $\mathcal{R}(Z)$ . One such basis is

$$\hat{Z} = Z(Z^H Z)^{-\frac{1}{2}} = (X + YQ)(I + Q^H Q)^{-\frac{1}{2}},$$

where  $(I + Q^H Q)^{-\frac{1}{2}}$  is the inverse of the positive definite square root of  $I + Q^H Q$ . Thus the cosines of the canonical angles are the singular values of

$$X^H \hat{Z} = (I + QQ^H)^{-\frac{1}{2}},$$

which by the above observation are the reciprocals of the singular values of  $Z$ . Hence  $\cos \theta_i = \sigma_i^{-1}$ , or  $\sec \theta_i = \sigma_i$ . The relation  $\tan \theta_i = \varsigma_i$  now follows from (2.1). ■

If we combine this theorem with Theorem 1.8, we obtain the following corollary, which generalizes (3.12), Chapter 1.

**Corollary 2.5.** *Let  $X$  be an orthonormal basis for a simple eigenspace  $\mathcal{X}$  of  $A$  and let  $Y$  be a basis for the corresponding left eigenspace  $\mathcal{Y}$  of  $A$  normalized so that  $Y^H X = I$ . Then the singular values of  $Y$  are the secants of the canonical angles between  $\mathcal{X}$  and  $\mathcal{Y}$ . In particular,*

$$\|Y\|_2 = \sec \theta_1(\mathcal{X}, \mathcal{Y}).$$

## 2.2. RESIDUAL ANALYSIS

Most methods for computing eigenspaces of a matrix  $A$  produce a sequence  $X_k$  of bases whose spans, it is hoped, converge to an eigenspace of  $A$ . In practice, the only way we have for determining convergence is to look at the residual  $R_k = AX_k - X_k L_k$ , where  $L_k$  is suitably chosen. If  $R_k$  is zero, then  $(L_k, X_k)$  is an eigenpair of  $A$ . It therefore seems reasonable to stop when  $R_k$  is sufficiently small.

In this subsection we will consider the implications of a small residual. First we will show how to compute an optimal block  $L$  — a generalization of Theorem 1.4, Chapter 2. We will then show that a small residual implies a small backward error — a generalization of Theorem 1.3, Chapter 2. Finally, we will develop residual bounds for the error in  $(L, X)$  as an approximate eigenpair.

### Optimal residuals

Most of our algorithms will furnish an approximate eigenbasis  $X$  but will allow us the freedom to choose the eigenblock. It is natural to choose this block to make the residual  $R = AX - X L$  as small as possible. The following theorem shows how to make this choice. We will actually prove a little more.

**Theorem 2.6.** Let  $(X \ X_{\perp})$  be unitary. Let  $R = AX - XL$  and  $S^H = X^H A - LX^H$ . Then in any unitarily invariant norm  $\|R\|$  and  $\|S\|$  are minimized when

$$L = X^H AX,$$

in which case

$$\|R\| = \|X_{\perp}^H AX\| \quad \text{and} \quad \|S\| = \|X^H AX_{\perp}\|. \quad (2.2)$$

**Proof.** We will prove the result for  $R$ , the proof for  $S$  being similar. Set

$$\begin{pmatrix} X^H \\ X_{\perp}^H \end{pmatrix} A(X \ X_{\perp}) = \begin{pmatrix} \hat{L} & H \\ G & M \end{pmatrix}.$$

Then

$$\begin{pmatrix} X^H \\ X_{\perp}^H \end{pmatrix} R = \begin{pmatrix} \hat{L} & H \\ G & M \end{pmatrix} \begin{pmatrix} X^H \\ X_{\perp}^H \end{pmatrix} X - \begin{pmatrix} X^H \\ X_{\perp}^H \end{pmatrix} X L = \begin{pmatrix} \hat{L} - L \\ G \end{pmatrix}.$$

Since  $\|\cdot\|$  is unitarily invariant,

$$\|R\| = \left\| \begin{pmatrix} \hat{L} - L \\ G \end{pmatrix} \right\|,$$

which is minimized when  $L = \hat{L} = X^H AX$ . The norm at the minimum is  $\|G\| = \|X_{\perp}^H AX\|$ . ■

### The Rayleigh quotient

The quantity  $L = X^H AX$  is a generalization of the scalar Rayleigh quotient of Definition 1.5, Chapter 2. If  $X$  is an eigenbasis of  $A$ , then the Rayleigh quotient is an eigen-block. Consequently, if  $X$  is near an eigenbasis of  $A$ , we may expect the Rayleigh quotient to contain good approximations to the eigenvalues of  $A$ . Moreover, by Theorem 1.2 we may expect that if  $u$  is an eigenvector of  $L$  then  $Xu$  will be near an eigenvector of  $A$ . We will give substance to these expectations in §4, where we consider the Rayleigh–Ritz method.

The matrix  $X^H AX$  is not the most general matrix that will reproduce an eigen-block when  $X$  is an eigenbasis. Theorem 1.2 suggests the following definition.

**Definition 2.7.** Let  $X$  be of full column rank and let  $X^I$  be a left inverse of  $X$ . Then the quantity  $X^I A X$  is a RAYLEIGH QUOTIENT of  $A$ .

It may seem odd to have a quotientless Rayleigh quotient, but in fact the division is implicit in  $X^I$ . To make it explicit, let  $X$  be of full rank and let  $Y^H X$  be nonsingular. Then  $(Y^H X)^{-1} Y^H$  is a left inverse of  $X$  and  $(Y^H X)^{-1} Y^H A X$  is a Rayleigh quotient.

### Backward error

In Theorem 1.3, Chapter 2, we saw that if  $r = Ax - \lambda x$ , then  $(\lambda, x)$  is an exact eigenpair of a perturbed matrix  $A + E$ , where  $E$  is of a size with  $r$ . The following theorem generalizes this result.

**Theorem 2.8.** *Let  $X$  be orthonormal and let*

$$R = AX - XL.$$

*Then there is a matrix*

$$E = -RX^H \quad (2.3)$$

*such that*

$$\|E\|_p = \|R\|_p, \quad p = 2, F,$$

*and*

$$(A + E)X = XL. \quad (2.4)$$

*If  $A$  is Hermitian and*

$$L = X^H AX,$$

*then there is a Hermitian matrix*

$$E = -(RX^H + X^H R) \quad (2.5)$$

*satisfying (2.4). Moreover,*

$$\|E\|_p = \begin{cases} \|R\|_p, & p = 2, \\ \sqrt{2}\|R\|_p, & p = F. \end{cases} \quad (2.6)$$

**Proof.** The proof is a generalization of the proof of Theorem 1.3, Chapter 2. The only tricky point is the verification of (2.6), which can be effected by observing that in the notation of the proof of Theorem 2.6,

$$\begin{pmatrix} X^H \\ X_\perp^H \end{pmatrix} E(X \ X_\perp) = - \begin{pmatrix} 0 & G^H \\ G & 0 \end{pmatrix}.$$

The singular values of this matrix are  $\pm\sigma_i$ , where the  $\sigma_i$  are the singular values of  $G$ . Since the singular values of  $G$  and  $R$  are the same, the 2-norm of this matrix is the largest singular value of  $R$ , and the square of the Frobenius norm is twice the sum of squares of the singular values of  $R$ . ■

The import of this theorem is discussed just after Theorem 1.3, Chapter 2. Briefly, if the residual is small, then  $(L, X)$  is an exact eigenpair of a slightly perturbed matrix  $A + E$ . This does not guarantee that  $(L, X)$  is a good approximation to the eigenpair, but for it not to be  $(L, X)$  must be ill conditioned—that is, sensitive to small changes in  $A + E$ . Thus basing convergence on the size of the residual guarantees that well-conditioned eigenpairs will be determined accurately.

### Residual bounds for eigenvalues of Hermitian matrices

By combining Theorems 3.8 and 3.10 in Chapter 1 with Theorem 2.8 we can obtain residual bounds on the eigenvalues of a Hermitian matrix. Specifically,  $\Lambda(L) \subset \Lambda(A+E)$ . Since the eigenvalues of  $A+E$  and  $A$ , arranged in descending order, cannot be separated by more than  $\|E\|_2$ , we have the following corollary.

**Corollary 2.9.** *Let  $\ell_1, \dots, \ell_l$  be the eigenvalues of  $L$ . Then there are eigenvalues  $\lambda_{j_1}, \dots, \lambda_{j_l}$  of  $A$  such that*

$$|\ell_i - \lambda_{j_i}| \leq \|R\|_2$$

and

$$\sqrt{\sum_{i=1}^l (\ell_i - \lambda_{j_i})^2} \leq \sqrt{2}\|R\|_F. \quad (2.7)$$

The factor  $\sqrt{2}$  in (2.7) can be eliminated (see the notes and references). We will see in §2.4 that if we know more about the eigensystem of  $A$  then we can get tighter residual bounds.

### Block deflation

Although the small backward error associated with a small residual is a powerful reason for using the residual norm as a convergence criteria, the question remains of how accurate the approximate eigenspace is. We will derive such bounds by a process of block deflation.

Let  $(X \ X_\perp)$  be unitary and let

$$\begin{pmatrix} X^H \\ X_\perp^H \end{pmatrix} A \begin{pmatrix} X & X_\perp \end{pmatrix} = \begin{pmatrix} L & H \\ G & M \end{pmatrix}. \quad (2.8)$$

Then by Theorem 2.6 we know that in any unitarily invariant norm  $\|\cdot\|$ ,  $\|R\| \equiv \|AX - XL\| = \|G\|$ . Consequently, if  $R$  is small, the right-hand side of (2.8) is almost block triangular. We will derive bounds by nudging it into full block triangularity.

Specifically, consider the similarity transformation

$$\begin{aligned} & \begin{pmatrix} I & 0 \\ -P & I \end{pmatrix} \begin{pmatrix} L & H \\ G & M \end{pmatrix} \begin{pmatrix} I & 0 \\ P & I \end{pmatrix} \\ &= \begin{pmatrix} L + HP & H \\ G + MP - PL - PHP & M - PH \end{pmatrix}. \end{aligned}$$

For the  $(2, 1)$  block of the right-hand side to be zero, we must have  $PL - MP = G - PHP$ , which can be written in the form

$$SP = G - PHP, \quad (2.9)$$

where  $\mathbf{S}$  is the Sylvester operator  $P \mapsto PL - MP$ .

Equation (2.9) is nonlinear in  $P$  and has no closed form solution. However, under appropriate conditions its solution can be shown to exist and can be bounded.

**Theorem 2.10.** *Let  $\|\cdot\|$  denote a consistent norm. In the matrix*

$$B = \begin{pmatrix} L & H \\ G & M \end{pmatrix}$$

let

$$\text{sep}(L, M) = \inf_{\|Q\|=1} \|\mathbf{S}Q\|,$$

where  $\mathbf{S}$  is defined by

$$\mathbf{S} = P \mapsto PL - MP. \quad (2.10)$$

If

$$4\|G\|\|H\| < \text{sep}^2(L, M),$$

there is a unique matrix  $P$  satisfying  $PL - MP = G - PHP$  and

$$\|P\| < 2 \frac{\|G\|}{\text{sep}(L, M)}$$

such that

$$\hat{B} = \begin{pmatrix} I & 0 \\ -P & I \end{pmatrix} B \begin{pmatrix} I & 0 \\ P & I \end{pmatrix} = \begin{pmatrix} L + HP & H \\ 0 & M - PH \end{pmatrix}.$$

The spectra of  $L + HP$  and  $M - PH$  are disjoint.

For the proof of this theorem see the notes and references.

We will first discuss this theorem in terms of the matrix  $B$  and then, later, in terms of the original matrix  $A$  in (2.8).

Let

$$X = \begin{pmatrix} I \\ 0 \end{pmatrix} \quad \text{and} \quad \hat{X} = \begin{pmatrix} I \\ P \end{pmatrix}.$$

Then  $(X, L)$  is an approximate eigenpair of  $B$  with residual norm  $\|G\|$ . The theorem shows that  $(L + HP, \hat{X})$  is a simple eigenpair of  $B$  with complementary eigenblock  $M - PH$ . By Theorem 2.4 the singular values of  $P$  are the tangents of the canonical angles between  $X$  and  $\hat{X}$ . Consequently, if  $\|\cdot\|$  is the spectral norm, the largest canonical angle  $\theta_1$  between  $X$  and  $\hat{X}$  satisfies

$$\tan \theta_1 < 2 \frac{\|G\|_2}{\text{sep}(L, M)}. \quad (2.11)$$

### The quantity $\text{sep}$

The quantity  $\text{sep}(L, M)$ , which appears in (2.11), is a generalization of the quantity  $\text{sep}(\lambda, N)$  introduced in Theorem 3.11, Chapter 1. By Theorem 1.16, Chapter 1, it is nonzero if and only if the spectra of  $L$  and  $M$  are distinct. By Corollary 1.17, Chapter 1,

$$\text{sep}(L, M) \leq \min\{|\Lambda(L) - \Lambda(M)|\} \equiv \delta,$$

so that  $\text{sep}$  is a lower bound for the physical separation  $\delta$  of the spectra of  $L$  and  $M$ . Thus the poorer the separation, the larger the canonical angle between  $X$  and  $\hat{X}$ . Unfortunately,  $\text{sep}(L, M)$  can be much smaller than the physical separation, as Example 3.15, Chapter 1, shows.

The quantity  $\text{sep}$  has a number of useful properties, which are stated without proof in the following theorem.

**Theorem 2.11.** *Let  $\text{sep}$  be defined by a consistent norm  $\|\cdot\|$ . Then the following statements are true.*

1. *If  $\text{sep}(L, M) > 0$ , then  $\text{sep}(L, M) = \|\mathbf{S}^{-1}\|^{-1}$ , where  $\mathbf{S}$  is the Sylvester operator defined by (2.10).*
2.  $|\text{sep}(L + E, M + F) - \text{sep}(L, M)| \leq \|E\| + \|F\|$ .
3. *If the norm  $\|\cdot\|$  is absolute (i.e.,  $\|A\| = \||A|\|$ ) and*

$$L = \text{diag}(L_1, \dots, L_\ell) \quad \text{and} \quad M = \text{diag}(M_1, \dots, M_m),$$

*then*

$$\text{sep}(L, M) = \min_{i,j} \text{sep}(L_i, M_j).$$

4. *If  $L$  and  $M$  are Hermitian, then in the Frobenius norm*

$$\text{sep}_F = \min |\Lambda(L) - \Lambda(M)|.$$

5.  $\text{sep}(X^{-1}LX, Y^{-1}MY) \geq \frac{\text{sep}(L, M)}{\kappa(X)\kappa(Y)}$ , where  $\kappa(A) = \|A\| \|A^{-1}\|$ .

It is a (not entirely trivial) exercise in the manipulation of norms to establish these properties. Item 1 provides an alternative definition of  $\text{sep}$ . Item 2 is a continuity property. It says that small changes in  $L$  and  $M$  make like changes in  $\text{sep}$ . This fact will be important later, when we derive perturbation bounds for eigenspaces. Items 3 and 4 exhibit  $\text{sep}$  for matrices of special form. Finally, item 5 shows how far  $\text{sep}$  may be degraded under similarity transformations. It is useful in assessing the separation of diagonalizable matrices.

### Residual bounds

We now return to our original matrix  $A$  and derive residual bounds for the approximate subspaces.

**Theorem 2.12.** *Let  $X$  be an approximate eigenbasis for  $A$  and let  $(X \ X_\perp)$  be unitary. Let  $L = X^H A X$  and  $M = X_\perp^H A X_\perp$  be the Rayleigh quotients corresponding to  $X$  and  $X_\perp$ , and let*

$$R = AX - XL \quad \text{and} \quad S^H = X^H A - LX^H.$$

*If in any unitarily invariant norm*

$$4\|R\|\|S\| < \text{sep}(L, M), \quad (2.12)$$

*then there is a simple eigenpair  $(\hat{L}, \hat{X})$  of  $A$  satisfying*

$$\|L - \hat{L}\| < 2 \frac{\|R\|\|S\|}{\text{sep}(L, M)} \quad (2.13)$$

*and*

$$\|\tan \Theta(X, \hat{X})\| < 2 \frac{\|R\|}{\text{sep}(L, M)}. \quad (2.14)$$

**Proof.** Write

$$\begin{pmatrix} X^H \\ X_\perp^H \end{pmatrix} A(X \ X_\perp) = \begin{pmatrix} L & H \\ G & M \end{pmatrix}.$$

Then by Theorem 2.6  $\|R\| = \|G\|$  and  $\|S\| = \|H\|$ . If we now apply Theorem 2.10 accumulating transformations, we see that if (2.12) is satisfied then there is an eigenblock of  $A$  of the form  $(L + HP, X + X_\perp P)$ , where

$$\|P\| < 2 \frac{\|R\|}{\text{sep}(L, M)}.$$

The bound (2.13) follows on taking the norm of the difference of  $L$  and  $L + HP$ . The bound (2.14) follows from Theorem 2.4. ■

From a mathematical standpoint, this theorem does exactly what we need. It provides conditions under which a purported eigenpair  $(L, X)$  approximates an actual eigenpair of the matrix in question. In large eigenproblems, however, the size  $n$  of the matrix will be very much greater than the number  $k$  of columns of the matrix  $X$ . Although, we can compute  $L$ ,  $R$ , and  $S$  easily enough, we cannot compute  $M$ , which is large and generally dense, or the quantity  $\text{sep}(L, M)$ . We shall see later that we can approximate a condition number for an eigenblock, provided we have an approximation to the corresponding left eigenbasis; but when it comes to eigenspaces of very large matrices, one must have additional information about the disposition of the eigenvalues to use the theorem.

### 2.3. PERTURBATION THEORY

In the last subsection we considered the problem of bounding the error in an approximate eigenspace. In this subsection we will treat the problem of bounding the perturbation of an exact eigenspace when the matrix changes.

#### The perturbation theorem

Given a simple eigenspace of a matrix  $A$ , we wish to determine when there is a nearby eigenspace of a perturbation  $\tilde{A} = A + E$  of  $A$  and to bound the angles between the two spaces. In principle, we can do this by regarding the exact eigenspace of  $A$  as an approximate eigenspace of  $\tilde{A}$  and applying Theorem 2.12. In practice, we obtain better bounds by proceeding indirectly through a spectral decomposition of  $A$ .

**Theorem 2.13.** *Let  $A$  have the spectral representation*

$$A = X_1 LY_1^H + X_2 MY_2^H. \quad (2.15)$$

Let  $\tilde{A} = A + E$  and set

$$\begin{pmatrix} Y_1^H \\ Y_2^H \end{pmatrix} (A + E)(X_1 \ X_2) = \begin{pmatrix} L + F_{11} & F_{12} \\ F_{21} & M + F_{22} \end{pmatrix}. \quad (2.16)$$

Let  $\|\cdot\|$  be a consistent family of norms and set

$$\delta = \text{sep}(L, M) - \|F_{11}\| - \|F_{22}\|.$$

If

$$4\|F_{21}\|\|F_{12}\| < \delta^2 \quad (2.17)$$

then there is a matrix  $P$  satisfying

$$\|P\| < 2 \frac{\|F_{21}\|}{\delta} \quad (2.18)$$

such that

$$(\tilde{L}, \tilde{X}_1) = (L + F_{11} + F_{12}P, X_1 + X_2P) \quad (2.19)$$

and

$$(\tilde{M}, \tilde{Y}_2) = (M + F_{22} - PF_{12}, Y_2 - Y_1P) \quad (2.20)$$

are complementary, simple, right and left eigenpairs of  $\tilde{A}$ . The spectra of  $\tilde{L}$  and  $\tilde{M}$  are disjoint.

**Proof.** By item 2 of Theorem 2.11 the quantity  $\delta$  is a lower bound on  $\text{sep}(L + F_{11}, M + F_{22})$ . Hence if (2.17) is satisfied, Theorem 2.10 guarantees the existence of a matrix  $P$  satisfying (2.18) that block triangularizes the matrix (2.16). The expressions (2.19) and (2.20) follow upon accumulating the transformations that reduce  $\tilde{A}$  to block triangular form. ■

We will now examine the consequences of this theorem.

### Choice of subspaces

Theorem (2.13) actually represents a continuum of results, since spectral representations are only partially unique (see p. 245). The variants do not say the same things, since the matrices  $L$ ,  $M$ , and  $F_{ij}$  also change. There is, unfortunately, no single choice of spectral representation that is suitable for all applications, but the one that comes closest is the standard representation in which  $X_1$  and  $Y_2$  are orthonormal. In the remainder of this subsection we will therefore make the following assumptions.

*In Theorem 2.13, the matrices  $X_1$  and  $Y_2$  are assumed to be orthonormal. In consequence (Theorem 2.10) there is a matrix  $Q$  such that*

$$X_2 = Y_2 + X_1 Q \quad \text{and} \quad Y_1 = X_1 - Y_2 Q^H. \quad (2.21)$$

*The singular values of  $Q$  are the tangents of the canonical angles between  $\mathcal{X}_1 = \mathcal{R}(X_1)$  and  $\mathcal{Y}_1 = \mathcal{R}(Y_1)$  (Theorem 2.4). We will set*

$$\theta_{xy} = \text{the largest canonical between } \mathcal{X}_1 \text{ and } \mathcal{Y}_1.$$

### Bounds in terms of $E$

The bounds of Theorem 2.13 are cast in terms of the matrices  $F_{ij}$  because they are the primary quantities in the proof of the theorem. In practice, however, we will seldom have access to these matrices. What we are likely to have is a bound on some norm of  $E$ . In this case we must weaken the bounds by replacing  $\|F_{ij}\|$  by  $\|Y_i^H\| \|E\| \|X_j\|$ . In the 2-norm, these bounds can be conveniently written in the terms of  $\theta_{xy}$  defined in (2.21). Specifically,

$$\begin{aligned} \|F_{11}\| &\leq \sec \theta_{xy} \|E\|_2, & \|F_{12}\| &\leq \sec^2 \theta_{xy} \|E\|_2, \\ \|F_{21}\| &\leq \|E\|_2, & \|F_{22}\| &\leq \sec \theta_{xy} \|E\|_2. \end{aligned}$$

With these bounds, we may redefine

$$\delta = \text{sep}(L, M) - 2 \sec \theta_{xy} \|E\|_2$$

and replace the condition (2.17) by

$$\sec \theta_{xy} \|E\|_2 < \frac{\delta}{2}, \quad (2.22)$$

so that the bound (2.18) becomes

$$\|P\|_2 \leq 2 \frac{\|E\|_2}{\delta}. \quad (2.23)$$

Thus how  $\mathcal{X}_1$  and  $\mathcal{Y}_1$  are situated with respect to one another affects the condition for the existence of the perturbed eigenspace, but it does not affect the final bound on  $\|P\|_2$ .

### The Rayleigh quotient and the condition of $L$

Let

$$\hat{L} = Y_1^H \tilde{A} X_1 = L + F_{21}. \quad (2.24)$$

Since  $Y_1^H X_1 = I$ , the matrix  $\hat{L}$  is a Rayleigh quotient in the sense of Definition 2.7. This Rayleigh quotient is an especially good approximation to the eigenblock of  $\tilde{A}$ . In fact, from the above bounds

$$\|\tilde{L} - \hat{L}\|_2 < \|F_{12}\|_2 \|P\|_2 \leq 2 \sec^2 \theta_{xy} \frac{\|E\|_2^2}{\delta}. \quad (2.25)$$

Thus the difference between the Rayleigh quotient and the exact representation goes to zero as the square of  $\|E\|_2$ .

We also have

$$\|\tilde{L} - L\|_2 \leq \sec \theta_{xy} \|E\|_2 + 2 \sec^2 \theta_{xy} \frac{\|E\|_2^2}{\delta}.$$

Hence as  $E \rightarrow 0$

$$\|\tilde{L} - L\|_2 \leq \sec \theta_{xy} \|E\|_2 + O(\|E\|_2^2).$$

Thus the secant of the largest canonical angle between  $X_1$  and  $Y_1$  is a condition number for the eigenblock  $L$  that determines how the error in  $E$  propagates to the representation  $L$ . Thus we have the following generalization of the assertion (3.24), Chapter 1.

*Under the assumptions (2.21), the condition number in the 2-norm of the eigenblock of a simple eigenspace is the secant of the largest canonical angle between the corresponding left and right eigenspaces.* (2.26)

### Angles between the subspaces

In the relation  $\tilde{X}_1 = X_1 + X_2 P$ , the size of  $P$  is a measure of the deviation of  $\tilde{X}_1 = \mathcal{R}(\tilde{X}_1)$  from  $X_1$ . It is natural then to ask how this measure relates to the canonical angles between the two spaces. If  $X_2$  were orthonormal and  $Y_2^H X_1$  were zero, we could apply Theorem 2.4 to obtain this relation. Unfortunately, neither condition holds. But with a little manipulation we can find a basis for  $\tilde{X}_1$  to which Theorem 2.4 does apply.

From (2.21) we have

$$\tilde{X}_1 = X_1 + X_2 P = X_1 + (Y_2 + X_1 Q)P = X_1(I + QP) + Y_2 P.$$

Now  $\|Q\|_2 = \tan \theta_{xy}$  and  $\|P\| < 2\|E\|/\delta$ . Hence from (2.23)

$$\|QP\|_2 < 2 \frac{\tan \theta_{xy} \|E\|_2}{\delta} < 1,$$

the last inequality following from (2.22) (remember  $\sec^2 \theta_{xy} = 1 + \tan^2 \theta_{xy}$ ). It follows that  $I + QP$  is nonsingular and

$$\|(I + PQ)^{-1}\|_2 < \frac{1}{1 - 2 \tan \theta_{xy} \|E\|_2 / \delta}. \quad (2.27)$$

We can therefore write

$$\tilde{X}_1(I + QP)^{-1} = X_1 + Y_2 P(I + QP)^{-1}.$$

Now the subspace spanned by  $\tilde{X}_1(I + QP)^{-1}$  is still  $\tilde{\mathcal{X}}_1$ . Since  $X_1$  and  $Y_2$  are orthonormal and  $Y_2^H X_1 = 0$ , it follows from Theorem 2.4 that the tangents of the canonical angles between  $\mathcal{X}_1$  and  $\tilde{\mathcal{X}}_1$  are the singular values of  $P(I + QP)^{-1}$ . On taking norms and using (2.23) and (2.27) we have the following corollary.

**Corollary 2.14.** *Let  $\theta_{xy}$  be the largest canonical angle between  $\mathcal{X}_1$  and  $\mathcal{Y}_1$  and let  $\theta(\mathcal{X}_1, \tilde{\mathcal{X}}_1)$  be the largest canonical angle between  $\mathcal{X}_1$  and  $\tilde{\mathcal{X}}_1$ . Then*

$$\tan \theta(\mathcal{X}_1, \tilde{\mathcal{X}}_1) < \frac{\|E\|_2}{\delta - 2 \tan \theta_{xy} \|E\|_2}, \quad (2.28)$$

where

$$\delta = \text{sep}(L, M) - 2 \sec \theta_{xy} \|E\|_2.$$

Here are two comments on this corollary.

- The denominator of (2.28), which is necessarily positive, approaches  $\delta$  as  $E \rightarrow 0$ , so that when  $E$  is small the bound is effectively  $\|E\|_2/\delta$ . Moreover,  $\delta$  itself approaches  $\text{sep}(L, M)$ . Thus  $\text{sep}(L, M)$  is a condition number for  $\mathcal{X}_1$ . In summary:

*Let  $A$  have the standard spectral representation  $X_1 LY_1^H + X_2 MY_2^H$ . Then in the 2-norm the quantity  $\text{sep}(L, M)^{-1}$  is a condition number for the eigenspace  $\mathcal{R}(X_1)$ .*

- As  $E \rightarrow 0$  the matrix  $P(I + QP)^{-1} \rightarrow P$ . Consequently, for  $E$  small enough the singular values of  $P$  are approximations to the tangents of the canonical angles between  $\mathcal{X}_1$  and  $\tilde{\mathcal{X}}_1$ . In fact, since for small  $\theta$  we have  $\theta \cong \tan \theta$ , the singular values approximate the angles themselves.

- If we are willing to replace  $\text{sep}(L, M)$  with  $\text{sep}(\tilde{L}, M)$  we can generalize the proof of Theorem 3.13, Chapter 1, to obtain the following bound:

$$\sin \theta_1(\mathcal{X}_1, \tilde{\mathcal{X}}_1) \leq \frac{\|E\|_2}{\text{sep}(\tilde{L}, M)}. \quad (2.29)$$

However, this approach has the disadvantage that it assumes rather than proves the existence of  $\tilde{\mathcal{X}}_1$ .

## 2.4. RESIDUAL BOUNDS FOR HERMITIAN MATRICES

The residual bound theory of §2.2 addresses two logically distinct problems: the existence of certain eigenspaces of  $A$  and the bounding of the error in those eigenspaces. For Hermitian matrices we can often say a priori that the required eigenspaces exist. For example, suppose that the Hermitian matrix  $A$  has the form

$$A = \begin{pmatrix} L & G^H \\ G & M \end{pmatrix}$$

and that  $\delta = \min |\Lambda(L) - \Lambda(M)| > 0$ . If  $\|G\|_2$  is less than  $\frac{\delta}{2}$ , then by Theorem 3.8, Chapter 1, the eigenvalues of  $A$  must divide into two groups—those that are within  $\|G\|_2$  of  $\Lambda(L)$  and those that are within  $\|G\|_2$  of  $\Lambda(M)$ . The eigenspaces corresponding to these two groups of eigenvalues are candidates for bounding. The fact that we know these eigenspaces exist before we start enables us to get direct bounds that are simpler and more useful than the ones developed above.

The purpose of this subsection is to state theorems containing such direct residual bounds. The proofs, which are not needed in this work, will be omitted.

### Residual bounds for eigenspaces

We begin with the simpler of the two results on eigenspaces treated here. As usual,  $\Theta(\mathcal{X}, \mathcal{Y})$  will denote the diagonal matrix of canonical angles between the subspaces  $\mathcal{X}$  and  $\mathcal{Y}$ .

**Theorem 2.15.** *Let the Hermitian matrix  $A$  have the spectral representation*

$$A = X L X^H + Y M Y^H,$$

*where  $(X \ Y)$  is unitary. Let the orthonormal matrix  $Z$  be of the same dimensions as  $X$ , and let*

$$R = AZ - ZN,$$

*where  $N$  is Hermitian. If*

$$\delta = \min |\Lambda(M) - \Lambda(N)| > 0, \tag{2.30}$$

*then*

$$\|\sin \Theta(\mathcal{R}(X), \mathcal{R}(Z))\|_F \leq \frac{\|R\|_F}{\delta}.$$

The separation hypothesis (2.30) is different from that required by the residual bounds of Theorem 2.12. There we required that the spectra of two Rayleigh quotients corresponding to the approximate subspaces be disjoint. Here the separation is between the certain eigenvalues of  $A$  (those of  $L$ ) and eigenvalues associated with the

approximating subspace (those of  $N$ ). Moreover, the separation criterion is less stringent in Theorem 2.15 than that of Theorem 2.12, and the final bound is smaller—the benefits of being Hermitian.

The theorem reflects an important tradeoff in bounds of this kind. The condition (2.30) requires only that the eigenvalues values of  $M$  and  $N$  be decently separated. Otherwise, they are free to interlace with eigenvalues of  $M$  lying between those of  $N$  and vice versa. The price we pay is that we can only bound the Frobenius norm of  $\sin \Theta(\mathcal{R}(X), \mathcal{R}(Z))$ —i.e., the square root of the sum of squares of the sines of the canonical angles. For individual angles the bound will generally be an overestimate. We can resolve this problem by restricting how the eigenvalues of  $M$  and  $N$  lie with respect to one another.

**Theorem 2.16.** *In the notation of Theorem 2.15 suppose that*

$$\Lambda(N) \subset [\alpha, \beta]$$

*and for some  $\delta > 0$*

$$\Lambda(M) \subset \mathbb{R} \setminus [\alpha - \delta, \beta + \delta]. \quad (2.31)$$

*Then for any unitarily invariant norm*

$$\|\sin \Theta(\mathcal{R}(X), \mathcal{R}(Z))\| \leq \frac{\|R\|}{\delta}. \quad (2.32)$$

The conditions on the spectra of  $N$  and  $M$  say that the spectrum of  $N$  lies in an interval that is surrounded at a distance of  $\delta$  by the spectrum of  $M$ . The bound is now in an arbitrary unitarily invariant norm. The natural norm to use is the 2-norm, in which case (2.32) bounds the largest canonical angle between the two spaces.

### Residual bounds for eigenvalues

In Corollary 2.9 we showed that if  $R = AX - X L$  is small then there are eigenvalues of  $A$  lying near those of  $L$ . The bound is of order  $\|R\|$ . The following shows that if we can specify a separation between the eigenvalues of  $L$  and a complementary set of eigenvalues of  $A$  then we can reduce the bound to be of order  $\|R\|^2$ .

**Theorem 2.17.** *Let  $(Z \ Z_\perp)$  be unitary, where  $Z$  has  $k$  columns. Let*

$$L = Z^H A Z, \quad M = Z_\perp^H A Z_\perp, \quad \text{and} \quad R = AZ - ZL.$$

*Let*

$$\begin{aligned} \Lambda(A) &= \{\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n\}, \\ \Lambda[\text{diag}(L, M)] &= \{\ell_1 \geq \ell_2 \geq \dots \geq \ell_n\}, \\ \Lambda(L) &= \{\ell_{i_1} \geq \ell_{i_2} \geq \dots \geq \ell_{i_k}\}. \end{aligned}$$

If

$$\delta = \min \{ |\Lambda(M) \setminus \{\lambda_{i_1}, \dots, \lambda_{i_k}\}| \} > \|R\|_2,$$

then

$$|\lambda_{i_j} - \ell_{i_j}| \leq \frac{\|R\|_2^2}{\delta}, \quad j = 1, \dots, k. \quad (2.33)$$

The theorem basically says that if  $A$  has a set  $\mathcal{L}$  of eigenvalues that are close to  $\Lambda(L)$  and that are removed from  $\Lambda(M)$  by  $\delta$ , then for  $R$  small enough, the eigenvalues  $\mathcal{L}$  and  $\Lambda(L)$  can be paired so that they are within  $O(\|R\|_2^2)$  of each other.

This theorem has the great advantage that we do not have to put restrictions such as (2.31) on the spectra. However, in sparse applications it has the disadvantage that we do not know  $N$ . But if we know that (in the notation of the last paragraph) there is a set  $\mathcal{L}_\perp$  of eigenvalues of  $A$  with  $\min\{|\Lambda(M) - \mathcal{L}_\perp|\} = \hat{\delta}$ , then we can write

$$\delta \geq \hat{\delta} - 2\|R\|_2.$$

Hence the condition  $\|R\|_2 < \delta$  can be replaced by the condition  $3\|R\| < \hat{\delta}$  and right-hand side of (2.33) by  $\|R\|_2^2/(\hat{\delta} - 2\|R\|_2)$ .

## 2.5. NOTES AND REFERENCES

### General references

For general references on perturbation theory see §3.3, Chapter 1.

### Canonical angle

The idea of canonical angles between subspaces goes back to Jordan [138, 1875], and has often been rediscovered. For a more complete treatment of this subject see Volume I of this series or [269, Ch. I].

The definitive paper on computing canonical angles is by Björck and Golub [26], who recommend computing the sines of the angles. In our applications  $n$  will be very large compared with the common dimension  $k$  of the subspaces. In this case  $Y_\perp$  is very large, and it is impossible to compute  $Y_\perp^H X$  directly. However, given a basis for  $Y$  one can compute an orthogonal matrix whose first  $k$  columns span  $Y$  as a product of  $k$  Householder transformations (Algorithm I:4.1.2). These transformations can then be applied to  $X$  to give  $Y_\perp^H X$  in the last  $n-k$  rows of the result.

### Optimal residuals and the Rayleigh quotient

Theorem 2.6 for Hermitian matrices is due to Kahan [142, 1967] (his proof works for non-Hermitian matrices). Stewart [252, 1973] stated the result explicitly for non-Hermitian matrices and gave the formula (2.2) for the optimal residual  $R$ . Earlier, Wilkinson [300, 1965, p. 264] proved the result for vectors.

### Backward error

For Hermitian matrices in the vector case, a result of Dennis and Moré [63, 1977] shows that (2.5) is optimal (also see [64, Theorem 9.11]).

The theorems stated here all have to do with approximate right eigenspaces, and of course they have trivial variants that deal with left eigenspaces. If we have small right and left residuals  $R = AX - XL$  and  $S^H = Y^H A - L^H Y^H$  it is natural to ask if there is a perturbation that makes  $X$  a right eigenbasis and  $Y$  a left eigenbasis. Kahan, Parlett, and Jiang [144] produce perturbations that are optimal in the spectral and Frobenius norms.

### Residual eigenvalue bounds for Hermitian matrices

Corollary 2.9 is due to Kahan [142, 1967], who showed that the factor  $\sqrt{2}$  can be removed from the right-hand side of (2.7). The result can be generalized to unitarily invariant norms [269, Theorem IV.4.14]. A treatment of other theorems of this kind may be found in [206].

### Block deflation and residual bounds

The use of block triangularization to derive residual bounds is due to Stewart [250, 1971], who also introduced the quantity  $\text{sep}$ . The approach taken here, in which the transforming matrix is triangular, is simpler than the original, which used an orthogonal transformation. For more on the history of such bounds see [269, p. 244].

### Perturbation theory

The passage from residual bounds to perturbation bounds was given in [254]. It is worth noting we can go the other way. Given a residual, we can generate a backward error and apply a perturbation theorem to get a residual bound.

The fact that the Rayleigh quotient (2.24) is accurate up to terms of order  $\|E\|^2$  shows that not all Rayleigh quotients are equal. The Rayleigh quotient  $X_1^H \tilde{A} X_1$ , for example, is accurate only up to terms of order  $\|E\|$ . The greater accuracy of  $Y_1^H \tilde{A} X_1$  is due to the fact that we use approximations to both the left and right eigenbases in its definition. Alternatively, we can regard the greater accuracy as due to the fact that the matrix (2.16) is an off-diagonal perturbation of the block diagonal matrix consisting of the two Rayleigh quotients  $Y_1^H \tilde{A} X_1$  and  $Y_2^H \tilde{A} X_2$ .

The bound (2.29) is due to Ipsen [126].

### Residual bounds for eigenspaces and eigenvalues of Hermitian matrices

The residual bounds for eigenspaces are due to Davis and Kahan [54, 1970], who prove much more in their ground-breaking paper. The residual bounds for eigenvalues are due to Mathias [174] and improve earlier bounds by Stewart [261].

### 3. KRYLOV SUBSPACES

Most algorithms for solving large eigenproblems proceed in two stages. The first stage generates a subspace containing approximations to the desired eigenvectors or eigen-spaces. The second stage extracts approximations to the eigenvectors from the sub-space generated in the first stage. If the approximations are not satisfactory, they may be used to restart the first stage.

In this section we will be concerned with the most widely used method for computing subspaces containing eigenvectors—the method of Krylov sequences. In the first subsection we will introduce Krylov sequences and their associated subspaces. We will then consider the accuracy of the approximates produced by the method in §3.2. Finally, in §3.3 we will introduce block Krylov sequences.

#### 3.1. KRYLOV SEQUENCES AND KRYLOV SPACES

##### Introduction and definition

The power method (§1, Chapter 2) is based on the observation that if  $A$  has a dominant eigenpair then under mild restrictions on  $u$  the vectors  $A^k u$  produce increasingly accurate approximations to the dominant eigenvector. However, at each step the power method considers only the single vector  $A^k u$ , which amounts to throwing away the information contained in the previously generated vectors. It turns out that this information is valuable, as the following example shows.

**Example 3.1.** A diagonal matrix  $A$  of order 100 was generated with eigenvalues

$$1, .95, .95^2, .95^3, \dots, .95^{99}.$$

Starting with a random vector  $u_1$ , the vectors  $u_k = A^k u_1$  were generated. The solid line in Figure 3.1 shows the tangent of the angle between  $A^k u_1$  and the dominant eigenvector of  $A$ . It is seen that the convergence is slow, as it must be, since the ratio of the subdominant eigenvalue to the dominant eigenvalue is 0.95. The dashed line gives the tangent between the dominant eigenvector and the space spanned by  $u_1, \dots, u_k$ . The convergence is much more rapid, reducing the error by eight orders of magnitude. (The dash-dot line shows the convergence to the subdominant eigenvector, of which more later.)

The swift convergence of an approximation to the dominant eigenvector in the subspace spanned by  $u_1, Au_1, \dots, A^{k-1}u_1$  justifies studying such subspaces. We begin with a definition.

**Definition 3.2.** Let  $A$  be of order  $n$  and let  $u \neq 0$  be an  $n$  vector. Then the sequence

$$u, Au, A^2u, A^3u, \dots$$

is a KRYLOV SEQUENCE BASED ON  $A$  AND  $u$ . We call the matrix

$$K_k(A, u) = (u \ Au \ A^2u \ \dots \ A^{k-1}u)$$

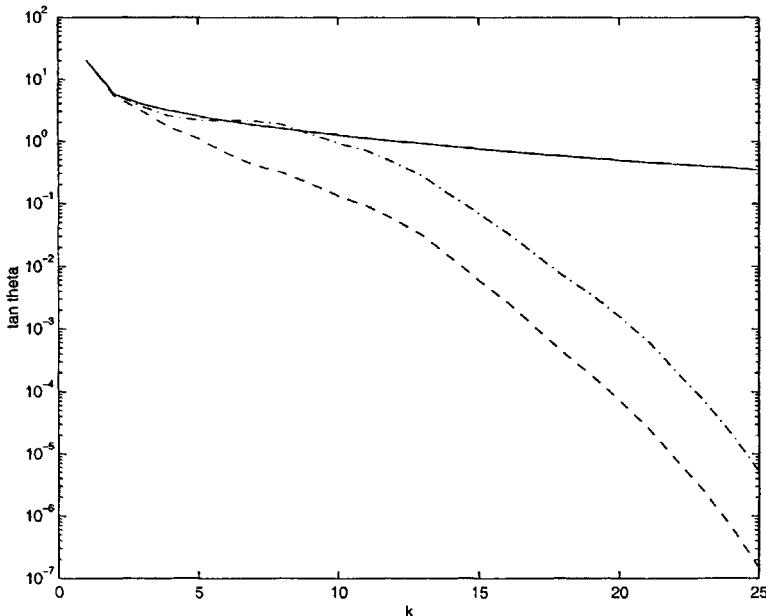


Figure 3.1: Power method and Krylov approximations

the  $k$ th KRYLOV MATRIX. The space

$$\mathcal{K}_k(A, u) = \mathcal{R}[K_k(A, u)]$$

is called the  $k$ th KRYLOV SUBSPACE.

When  $A$  or  $u$  are known from context, we will drop them in our notation for Krylov matrices and subspaces. Thus we may write  $\mathcal{K}_k(u)$  or simply  $\mathcal{K}_k$ .

### Elementary properties

The basic properties of Krylov subspaces are summarized in the following theorem. The proof is left as an exercise.

**Theorem 3.3.** Let  $A$  and  $u \neq 0$  be given. Then

1. The sequence of Krylov subspaces satisfies

$$\mathcal{K}_k(A, u) \subset \mathcal{K}_{k+1}(A, u)$$

and

$$A\mathcal{K}_k(A, u) \subset \mathcal{K}_{k+1}(A, u).$$

2. If  $\sigma \neq 0$ ,

$$\mathcal{K}_k(A, u) = \mathcal{K}_k(\sigma A, u) = \mathcal{K}_k(A, \sigma u).$$

3. For any  $\kappa$ ,

$$\mathcal{K}_k(A, u) = \mathcal{K}_k(A - \kappa I, u).$$

4. If  $W$  is nonsingular, then

$$\mathcal{K}_k(W^{-1}AW, W^{-1}u) = W^{-1}\mathcal{K}(A, u).$$

The second item in the above theorem states that a Krylov subspace remains unchanged when either  $A$  or  $u$  is scaled. The third says that it is invariant under shifting. Finally, the fourth says that the space behaves in a predictable way under similarity transformations. In particular if  $A$  is Hermitian or nondefective, we may diagonalize it—a strategy that sometimes simplifies the analysis of Krylov sequences.

### The polynomial connection

Krylov subspaces have an important characterization in terms of matrix polynomials. By definition any vector  $v$  in  $\mathcal{K}_k(A, u)$  can be written in the form

$$v = \gamma_1 u + \gamma_2 A u + \gamma_3 A^2 u + \cdots + \gamma_{k-1} A^{k-1} u.$$

Hence if we define the matrix polynomial  $p(A)$

$$p(A) = \gamma_1 I + \gamma_2 A + \gamma_3 A^2 + \cdots + \gamma_{k-1} A^{k-1},$$

then  $v = p(A)u$ . Conversely, if  $v = p(A)u$  for any polynomial of degree not exceeding  $k-1$ , then  $v \in \mathcal{K}_k(A, u)$ . Thus we have proved the following theorem.

**Theorem 3.4.** The space  $\mathcal{K}_k(A, u)$  can be written in the form

$$\mathcal{K}_k(A, u) = \{p(A)u : \deg(p) \leq k\}.$$

### Termination

If  $(\lambda, u)$  is an eigenpair of  $A$ , then  $A^k u = \lambda^k u$ , and

$$\mathcal{K}_k(A, u) = \mathcal{K}_1(A, u), \quad k = 1, 2, \dots$$

In other words the Krylov sequence terminates in the sense that the Krylov vectors after the first one provide no new information. More generally, we will say that a Krylov sequence *terminates at  $\ell$*  if  $\ell$  is the smallest integer such that

$$\mathcal{K}_{\ell+1}(A, u) = \mathcal{K}_\ell(A, u). \tag{3.1}$$

The following theorem states the basic facts about termination.

**Theorem 3.5.** A Krylov sequence terminates based on  $A$  and  $u$  at  $\ell$  if and only if  $\ell$  is the smallest integer for which

$$\dim[\mathcal{K}_{\ell+1}] = \dim[\mathcal{K}_\ell]. \quad (3.2)$$

If the Krylov sequence terminates at  $\ell$ , then  $\mathcal{K}_\ell$  is an eigenspace of  $A$  of dimension  $\ell$ . On the other hand, if  $u$  lies in an eigenspace of dimension  $m$ , then for some  $\ell \leq m$ , the sequence terminates at  $\ell$ .

**Proof.** Equation (3.2) follows from (3.1). On the other hand if (3.2) is satisfied, (3.1) follows from the fact that  $\mathcal{K}_\ell \subset \mathcal{K}_{\ell+1}$ .

If the sequence terminates at  $\ell$ , then

$$A\mathcal{K}_\ell \subset \mathcal{K}_{\ell+1} = \mathcal{K}_\ell,$$

so that  $\mathcal{K}_\ell$  is an invariant subspace of  $A$ .

If  $u$  is in an invariant subspace  $\mathcal{X}$  of dimension  $m$ , then so are the members of the sequence  $u, Au, \dots, A^{m-1}u$ . Now if the sequence terminates at  $\ell > m$  it follows that  $m = \dim(\mathcal{K}_\ell) > \dim(\mathcal{K}_m) = \dim(\mathcal{X})$ , which is impossible. ■

Termination is a two-edged sword. On the one hand, if a Krylov sequence terminates, it contains exact eigenvectors of  $A$ . On the other hand, it stops furnishing information about other eigenvectors. As we shall see, the sword is especially difficult to wield in the presence of rounding error.

### 3.2. CONVERGENCE

We have seen in Example 3.1 that a sequence of Krylov subspaces can contain increasingly accurate approximations to the certain eigenvectors of the matrix in question. In this case we will say, somewhat illogically, that the Krylov subspace *converges* to the vector. Experience has shown that these eigenvectors have eigenvalues that tend to lie on the periphery of the spectrum of the matrix. In this subsection we will give mathematical substance to this observation — at least for the Hermitian case.

#### The general approach

In what follows, we will assume that  $A$  is Hermitian and has an orthonormal system of eigenpairs  $(\lambda_i, x_i)$  ( $i = 1, \dots, n$ ). Let  $u$  be a starting vector for a Krylov sequence, and write  $u$  in the form

$$u = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n, \quad (3.3)$$

where  $\alpha_i = x_i^H u$ . By Theorem 3.4 any member of  $\mathcal{K}_k$  can be written in the form  $p(A)u$ , where  $p(\lambda)$  is a polynomial of degree not greater than  $k$ . Since  $p(A)x_i = p(\lambda_i)x_i$ , we have

$$p(A)u = \alpha_1 p(\lambda_1)x_1 + \alpha_2 p(\lambda_2)x_2 + \cdots + \alpha_n p(\lambda_n)x_n. \quad (3.4)$$

Thus if we can find a polynomial such that  $p(\lambda_i)$  is large compared with  $p(\lambda_j)$  ( $j \neq i$ ), then  $p(A)u$  will be a good approximation to  $x_i$ .

The following theorem quantifies this observation.

**Theorem 3.6.** *In the above notation, if  $x_i^H u \neq 0$  and  $p(\lambda_i) \neq 0$ , then*

$$\tan \angle(p(A)u, x_i) \leq \max_{j \neq i} \frac{|p(\lambda_j)|}{|p(\lambda_i)|} \cdot \tan \angle(u, x_i). \quad (3.5)$$

**Proof.** We begin by observing that if  $u$  has the expansion (3.3) then  $|\alpha_i|/\|u\|_2$  is the cosine of the angle  $\theta$  between  $u$  and  $x_i$  and  $\sqrt{\sum_{j \neq i} |\alpha_j|^2 / \|u\|_2^2}$  is the sine of  $\theta$ . Hence

$$\tan^2 \theta = \sum_{j \neq i} \frac{|\alpha_j|^2}{|\alpha_i|^2}.$$

It follows from (3.4) that

$$\begin{aligned} \tan^2 \angle(p(A)u, x_i) &= \sum_{j \neq i} \frac{|\alpha_i p(\lambda_j)|^2}{|\alpha_i p(\lambda_i)|^2} \\ &\leq \max_{j \neq i} \frac{|p(\lambda_j)|^2}{|p(\lambda_i)|^2} \sum_{j \neq i} \frac{|\alpha_j|^2}{|\alpha_i|^2} \\ &= \max_{j \neq i} \frac{|p(\lambda_j)|^2}{|p(\lambda_i)|^2} \cdot \tan \angle(u, x_i). \quad \blacksquare \end{aligned}$$

To apply this theorem note that the right-hand side of (3.5) does not depend on the scaling of  $p$ . We can therefore assume that  $p(\lambda_i) = 1$  and write

$$\tan \angle(p(A)u, x_i) \leq \max_{\substack{j \neq i \\ p(\lambda_j)=1}} |p(\lambda_j)| \cdot \tan \angle(u, x_i).$$

If  $\deg(p) \leq k - 1$ , then  $p(A)u \in \mathcal{K}_k$  and  $\tan \angle(p(A)u, x_i)$  is an upper bound for  $\tan \angle(x_i, \mathcal{K}_k)$ . Hence

$$\tan \angle(x_i, \mathcal{K}_k) \leq \min_{\substack{\deg(p) \leq k-1 \\ p(\lambda_i)=1}} \max_{j \neq i} |p(\lambda_j)| \cdot \tan \angle(u, x_i).$$

Thus if we can compute

$$\min_{\substack{\deg(p) \leq k-1 \\ p(\lambda_i)=1}} \max_{j \neq i} |p(\lambda_j)| \quad (3.6)$$

as a function of  $k$ , we will have a bound on the accuracy of approximations to  $x_i$  in the sequence of Krylov subspaces.

As it stands, this problem is too difficult. To simplify it, assume that

$$\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n$$

and that our interest is in the eigenvector  $x_1$ . Then

$$\min_{\substack{\deg(p) \leq k-1 \\ p(\lambda_1)=1}} \max_{\lambda \in [\lambda_n, \lambda_2]} |p(\lambda)|$$

is an upper bound for (3.6). Thus we are lead to the problem of calculating the bound

$$\tan \angle(x_1, \mathcal{K}_k) \leq \min_{\substack{\deg(p) \leq k-1 \\ p(\lambda_1)=1}} \max_{\lambda \in [\lambda_n, \lambda_2]} |p(\lambda)| \cdot \tan \angle(u, x_1). \quad (3.7)$$

This is a well-understood problem that has an elegant solution in terms of Chebyshev polynomials, to which we now turn.

### Chebyshev polynomials

The Chebyshev polynomials are defined by

$$c_k(t) = \begin{cases} \cos(k \cos^{-1} t), & |t| \leq 1, \\ \cosh(k \cosh^{-1} t), & |t| \geq 1. \end{cases} \quad (3.8)$$

Chebyshev polynomials are widely used in numerical analysis and are treated in many sources. Rather than rederive the basic results, we collect them in the following theorem.

**Theorem 3.7.** *Let  $c_k(t)$  be defined by (3.8). Then the following assertions are true.*

1.  $c_0(t) = 1, c_1(t) = t$ , and

$$c_{k+1}(t) = 2c_k(t) - c_{k-1}(t), \quad k = 1, 2, \dots$$

2. For  $|t| > 1$ ,

$$c_k(t) = (1 + \sqrt{t^2 - 1})^k + (1 + \sqrt{t^2 - 1})^{-k}. \quad (3.9)$$

3. For  $t \in [-1, 1], |c_k(t)| \leq 1$ . Moreover, if

$$t_{ik} = \cos \frac{(k-i)\pi}{k}, \quad i = 0, 1, \dots, k,$$

then

$$c_k(t_{ik}) = (-1)^{k-i}.$$

4. For  $s > 1$ ,

$$\min_{\substack{\deg(p) \leq k \\ p(s)=1}} \max_{t \in [0, 1]} |p(t)| = \frac{1}{c_k(s)}, \quad (3.10)$$

and the minimum is obtained only for  $p(t) = c_k(t)/c_k(s)$ .

### Convergence redux

We now return to the problem of bounding  $\tan \angle(x_1, \mathcal{K}_k)$ , which by (3.7) is equivalent to finding

$$\sigma_k = \min_{\substack{\deg(p) \leq k-1 \\ p(\lambda_1) = 1}} \max_{\lambda \in [\lambda_n, \lambda_2]} |p(\lambda)|.$$

We can apply (3.10) to compute  $\sigma_k$ , provided we change variables to transform the interval  $[\lambda_n, \lambda_1]$  to  $[-1, 1]$ . It is easily verified that the change of variables is

$$\lambda = \lambda_2 + (\mu - 1) \frac{\lambda_2 - \lambda_n}{2}.$$

The value of  $\mu$  at  $\lambda_1$  is

$$\mu_1 = 1 + 2 \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}.$$

Hence the value of  $\sigma_k$  is  $1/c_{k-1}(\mu_1)$ . In view of (3.9), we have the following theorem.

**Theorem 3.8.** *Let the Hermitian matrix  $A$  have an orthonormal system of eigenpairs  $(\lambda_i, x_i)$ , with its eigenvalues ordered so that*

$$\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n. \quad (3.11)$$

Let

$$\eta = \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}.$$

Then

$$\begin{aligned} \tan \angle[x, \mathcal{K}_k(A, u)] &\leq \frac{\tan \angle(x, u)}{c_{k-1}(1+2\eta)} \\ &= \frac{\tan \angle(x, u)}{(1+2\sqrt{\eta+\eta^2})^{k-1} + (1+2\sqrt{\eta+\eta^2})^{1-k}}. \end{aligned}$$

Three comments.

- The bound appears to be rather complicated, but it simplifies—at least asymptotically. For  $k$  large we can ignore the term with the exponent  $1 - k$  and write

$$\tan \angle[x, \mathcal{K}_k(A, u)] \lesssim \frac{\tan \angle(x, u)}{(1+2\sqrt{\eta+\eta^2})^{k-1}}.$$

- As with the power method, a nearby subdominant eigenvalue will retard convergence. However, in this case the bound gives us a bonus. For if  $\eta$  is small, the bound becomes

$$\tan \angle[x, \mathcal{K}_k(A, u)] \lesssim \frac{\tan \angle(x, u)}{(1+2\sqrt{\eta})^{k-1}}.$$

The square root increases the influence of  $\eta$  on the bound. For example, consider the matrix in Example 3.1. There  $\eta \cong 0.5/0.95 = 0.053$ . The square root of this number is about 0.23, and the bound on the convergence rate is  $1/(1 + 2 \cdot 0.23) = 0.69$ . Thus the square root effect gives a great improvement over the rate of 0.95 for the power method.

- By item 2 in Theorem 3.3, the Krylov subspaces of  $A$  remain unchanged when  $A$  is replaced by  $-A$ . Hence Theorem 3.8 can be used to bound convergence to the eigenvector corresponding to the smallest eigenvalue of  $A$ . However, it is important to keep in mind that the smallest eigenvalues of a matrix — particularly a positive definite matrix — often tend to cluster together, so that the bound will be unfavorable. In Example 3.1, the value of  $\eta$  corresponding to the smallest eigenvalue is about  $0.33 \cdot 10^{-4}$ , so that the bound on the convergence rate is about 0.97.
- 

The general approach can be used to derive bounds for the remaining eigenvectors  $x_2, x_3, \dots$ . For example, to derive a bound for  $x_2$ , let

$$p(t) = \frac{t - \lambda_1}{\lambda_2 - \lambda_1} q(t)$$

where  $q(t)$  is a polynomial of degree not greater than  $k - 2$  satisfying  $q(\lambda_2) = 1$ . Note that  $p(\lambda_1) = 0$  and  $p(\lambda_2) = 1$ . It then follows from (3.5) that

$$\begin{aligned} \tan \angle(x_2, \mathcal{P}_k) &\leq \min_{\substack{\deg(q) \leq k-2 \\ q(\lambda_2)=1}} \max_{j>2} \frac{\lambda_j - \lambda_1}{\lambda_2 - \lambda_1} q(\lambda_j) \cdot \tan \angle(x_2, u) \\ &\leq \frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1} \min_{\substack{\deg(q) \leq k-2 \\ q(\lambda_2)=1}} \max_{j>2} |q(\lambda_j)| \cdot \tan \angle(x_2, u) \\ &\leq \frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1} \frac{1}{c_{k-2}(1 + 2\eta_2)} \cdot \tan \angle(x_2, u), \end{aligned} \tag{3.12}$$

where

$$\eta_2 = \frac{\lambda_2 - \lambda_3}{\lambda_3 - \lambda_n}.$$

Bounds for  $\lambda_i$  ( $i > 2$ ) can be derived by considering a polynomial of the form

$$p(t) = \ell_i(t)q(t),$$

where  $\ell_i$  is the Lagrange polynomial that is zero at  $\lambda_1, \dots, \lambda_{i-1}$  and one at  $\lambda_i$ .

### Multiple eigenvalues

The hypothesis  $\lambda_1 > \lambda_2$  in Theorem 3.8 can be relaxed. To see this, suppose that  $\lambda_1 = \lambda_2 > \lambda_3$ . Let  $u$  be given, and expand  $u$  in the form

$$u = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \cdots + \alpha_n x_n,$$

where the  $x_i$  are the normalized eigenvectors corresponding to  $\lambda_i$ . Then

$$A^k u = (\alpha_1 x_1 + \alpha_2 x_2) \lambda_1^k + \alpha_3 x_3 \lambda_3^k + \cdots + \alpha_n x_n \lambda_n^k. \quad (3.13)$$

This shows that of all the vectors in the two-dimensional invariant subspace that is spanned by  $x_1$  and  $x_2$ , the spaces  $\mathcal{K}_k(A, u)$  contain only approximations to  $\alpha_1 x_1 + \alpha_2 x_2$ . It is just as though  $(\lambda_1, \alpha_1 x_1 + \alpha_2 x_2)$  were a simple eigenpair of  $A$ . Thus Theorem 3.8 continues to hold when  $\lambda_1$  is multiple provided we adopt the convention that only one copy of  $\lambda_1$  is listed in (3.11). Similarly, if we retain only one copy of  $\lambda_1$  and  $\lambda_2$ , the bound (3.12) continues to hold when  $\lambda_1$  and  $\lambda_2$  are multiple.

### Assessment of the bounds

It is instructive to apply the bounds to the Krylov sequence generated in Example 3.1. The bound predicts a value for  $\tan \angle(x_1, \mathcal{K}_{20})$  of  $2.0 \cdot 10^{-3}$ , where the actual value is  $1.5 \cdot 10^{-7}$ . The problem is that the theory predicts a convergence ratio of 0.67, while a glance at the graph in Figure 3.1 shows that the convergence ratios are steadily decreasing. Immediately after the first step they range from 0.55 to 0.75, but then they drop off. The last three convergence ratios are 0.31, 0.26, and 0.22. This phenomenon is not accounted for by our theory.

The dot-dash line in Figure 3.1 is  $\tan \angle(x_2, \mathcal{K}_{20})$ . The final value is  $5.2 \cdot 10^{-6}$ , whereas the bound predicts a value of  $4.0 \cdot 10^{-2}$  — once again, a poor prediction.

Actually, the graphs show that any exponentially decreasing bound of the form  $\alpha \rho^k$  must be weak. Simply take a ruler and draw a line from the common initial point that is tangent to the dashed line (at about  $k = 12$ ). Its value at  $k = 25$  is about  $7 \cdot 10^{-4}$ , which is the best we can expect and compares favorably with the bound  $2.0 \cdot 10^{-3}$ . The same procedure applied to the dot-dash line yields an optimal bound of about  $6 \cdot 10^{-3}$ . Thus part of the problem is with the mathematical form of our bounds, which does not reflect the true behavior of the Krylov approximations.

To summarize, the bounds provide support for the empirical observation that Krylov sequences generate good approximations to exterior eigenvalues, but they can underestimate how good the approximations actually are — sometimes by orders of magnitude. The reason is that the bounds must take into account the worst-case distribution of the eigenvalues. We will return to this point in a moment.

### Non-Hermitian matrices

The theory of Krylov subspaces for non-Hermitian matrices is less well developed. There are two reasons. First, the geometry of the eigenvalues is generally more complicated. Second, if the matrix is nonnormal, we may not have a complete system of eigenvectors to aid the analysis. The following example shows that a change in geometry can have a major effect on the rates of convergence.

**Example 3.9.** A normal (actually diagonal) matrix was constructed whose eigenvalues were those of a random complex matrix. The largest eigenvalue was recomputed

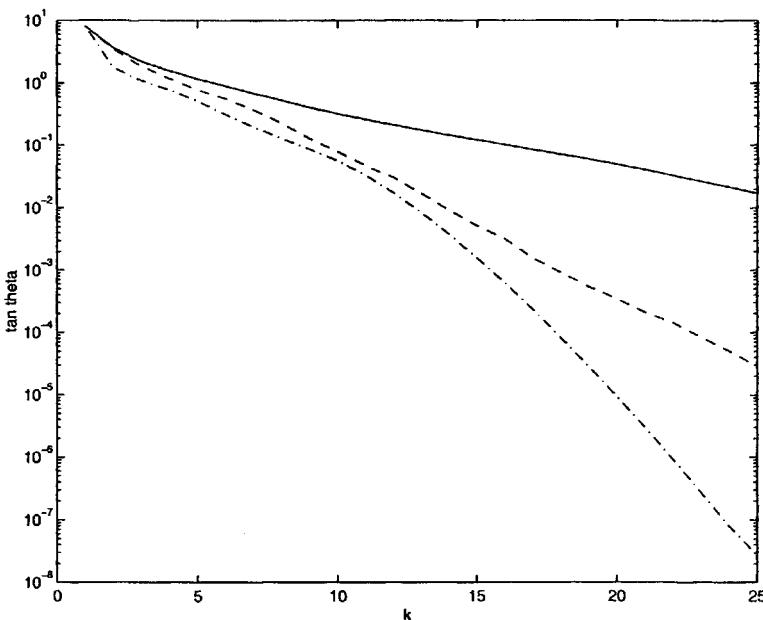


Figure 3.2: Power method and Krylov approximations

so that the absolute value of its ratio to the second largest eigenvalue was 0.95 (as in Example 3.1). The solid line in Figure 3.2 shows the tangent of the angle between the  $k$ th Krylov subspace and the dominant eigenvector. The convergence is not much better the power method in Example 3.1.

To vary the geometry, the eigenvalues were altered so that the ratio of the absolute values of the successive eigenvalues became 0.95 while their arguments remained the same. The dash-dot line shows how well the Krylov subspace approximates the eigenvector. The convergence is swift as in Example 3.1.

In a third experiment, the eigenvalues from the first experiment were replaced by their absolute values, so that the matrix is now Hermitian. The convergence, illustrated by the dashed line, is intermediate.

It would be foolish to conclude too much from these examples. It is clear that geometry matters and perhaps that exponentially decaying eigenvalues are a good thing. The example does have something to say about the convergence bounds from Theorem 3.8. Recall that the bound failed to predict the swift convergence to  $x_1$  for the matrix from Example 3.1. It turns out that the third experiment in Example 3.9, in which the matrix is Hermitian, has exactly the same bounds. Since it converges more slowly, the bounds cannot predict swifter convergence.

There is a polynomial connection for non-Hermitian matrices.

**Theorem 3.10.** Let  $\lambda$  be a simple eigenvalue of  $A$  and let

$$A = \lambda xy^H + XLY^H$$

be a spectral representation. Let

$$u = \alpha x + Xa,$$

where

$$\alpha = y^H u \quad \text{and} \quad a = Y^H u.$$

Then

$$\sin \angle[x_i, \mathcal{K}_k(A, u)] \leq |\alpha|^{-1} \min_{\substack{\deg(p) \leq k-1 \\ p(\lambda_i) = 1}} \|Xp(L)a\|_2.$$

**Proof.** By Theorem 2.3

$$\begin{aligned} \sin \angle[x_i, \mathcal{K}(A, u)] &= |\alpha_i|^{-1} \min_{y \in \mathcal{K}_k(A, u)} \|\alpha_i x_i - y\|_2 \\ &= |\alpha_i|^{-1} \min_{\deg(p) \leq k-1} \|\alpha_i x_i - p(A)u\|_2 \\ &\leq |\alpha_i|^{-1} \min_{\substack{\deg(p) \leq k-1 \\ p(\lambda_i) = 1}} \|\alpha_i x_i - p(A)u\|_2. \end{aligned}$$

Hence for any polynomial  $p$  of degree not greater than  $k-1$  with  $p(\lambda_i) = 1$ , we have

$$\begin{aligned} \sin \angle[x_i, \mathcal{K}(A, u)] &\leq |\alpha_i|^{-1} \|\alpha_i x_i - p(\lambda)xy^H u - Xp(L)Y^H u\| \\ &\leq |\alpha_i|^{-1} \|Xp(L)a\|_2. \quad \blacksquare \end{aligned}$$

This theorem is an analogue of Theorem 3.6. Actually, it is a continuum of theorems, since spectral representations are not unique. For example, if  $A$  is nondefective we can take  $L$  to be diagonal and the columns of  $X$  to have norm one, in which case the bound becomes (after a little manipulation)

$$\sin \angle[x_i, \mathcal{K}(A, u)] \leq \frac{\|a\|_1}{|\alpha|} \min_{\substack{\deg(p) \leq k-1 \\ p(\lambda_i) = 1}} \max_{i>1} |p(\lambda_i)|, \quad (3.14)$$

where  $\lambda_2, \dots, \lambda_n$  are the eigenvalues of  $A$  other than  $\lambda$ .

The theorem suggests that we attempt to choose polynomials  $p$  with  $p(\lambda) = 1$  so that  $p(L)$  is small. This is not, in general, an easy task, since the geometry of the eigenvalues can be complicated. What results exist suggest that under some conditions the Krylov sequence will produce good approximations to exterior eigenvectors. For more see the notes and references.

The theorem also illustrates the difficulties introduced by defectiveness and non-normality. Suppose, for example,  $\mu \neq \lambda$  is an eigenvalue of  $A$  with a Jordan block of order  $m$ . Then for  $p(L)$  to be zero, the polynomial  $p(\tau)$  must have the factor  $(\tau - \mu)^m$ . In other words, it is not sufficient to simply make  $p$  small at  $\mu$ . This implies that we cannot generalize the approach via Chebyshev polynomials to defective matrices.

### 3.3. BLOCK KRYLOV SEQUENCES

We saw above [see (3.13)] that when it comes to multiple eigenvalues, a Krylov sequence has tunnel vision. Specifically, if  $\lambda_1$  is double and the generating vector is expanded in the form

$$u = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \cdots + \alpha_n x_n,$$

the Krylov sequence can only produce the approximation  $\alpha_1 x_1 + \alpha_2 x_2$  to a vector in the eigenspace of  $\lambda_1$ . Of course, what we would like to approximate is the entire invariant subspace corresponding to that eigenvalue.

A cure for this problem is to iterate with a second vector, say

$$v = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n.$$

Then

$$A^k v = (\beta_1 x_1 + \beta_2 x_2) \lambda_1^k + \beta_3 x_3 \lambda_3^k + \cdots + \beta_n x_n \lambda_n^k,$$

so that  $\mathcal{K}_k(A, v)$  can approximate the linear combination  $\beta_1 x_1 + \beta_2 x_2$ . Since the sequence in  $u$  produces approximations to  $\alpha_1 x_1 + \alpha_2 x_2$ , unless we are unlucky the two approximations will span the required two-dimensional invariant subspace.

The foregoing suggests that we generate a *block Krylov sequence* by starting with a matrix  $U$  with linearly independent columns and computing

$$U, AU, A^2 U, \dots$$

The space  $\mathcal{K}_k(A, U)$  spanned by the first  $k$  members of this sequence is called the  $k$ th *block Krylov subspace*.

Although it requires more storage and work to compute a block Krylov sequence than an ordinary Krylov sequence, there are two compensating factors (aside from curing the problem of multiple eigenvalues). First, it frequently happens that the computation of the product of a matrix with a block  $U$  of, say,  $m$  vectors takes less than  $m$  times the time for a matrix-vector product. For example, if the matrix is stored on disk, it has to be retrieved only once to form  $AU$ . Second, passing to block Krylov sequences improves the convergence bound, as the following theorem shows.

**Theorem 3.11.** *Let  $A$  be Hermitian and let  $(\lambda_i, x_i)$  be a complete system of orthonormal eigenpairs of  $A$ , with the eigenvalues ordered so that*

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n,$$

*and assume that the multiplicity of  $\lambda_1$  is not greater than  $m$ . If*

$$B = \begin{pmatrix} x_1^H \\ \vdots \\ x_m^H \end{pmatrix} U$$

is nonsingular and we set

$$v = UB^{-1}\mathbf{e}_1,$$

then

$$\begin{aligned} \tan \angle[x_1, \mathcal{K}_k(A, U)] &\leq \frac{\tan \angle(x_1, v)}{c_{k-1}(1+2\eta)} \\ &= \frac{\tan \angle(x_1, v)}{(1+2\sqrt{\eta+\eta^2})^{k-1} + (1+2\sqrt{\eta+\eta^2})^{1-k}}, \end{aligned} \quad (3.15)$$

where

$$\eta = \frac{\lambda_1 - \lambda_{m+1}}{\lambda_{m+1} - \lambda_n}. \quad (3.16)$$

**Proof.** By construction  $v \in \mathcal{R}(U)$ . Hence  $\mathcal{K}_k(A, v) \subset \mathcal{K}_k(A, U)$  so that

$$\angle[x_1, \mathcal{K}_k(A, U)] \leq \angle[x_1, \mathcal{K}_k(A, v)].$$

From the definition of  $v$  it is easily verified that  $x_i^H v = 0$  ( $i = 2, \dots, m$ ). Hence the Krylov subspace  $\mathcal{K}_k(A, v)$  contains no components along  $x_2, \dots, x_m$ ; it is as if we were working with a matrix in which the pairs  $(\lambda_i, x_i)$  ( $i = 1, \dots, m$ ) had been deleted, so that the eigenvalue following  $\lambda_1$  is  $\lambda_{n+1}$ . We may now apply Theorem 3.8 to get (3.15). ■

Three comments.

- If the multiplicity of  $\lambda_1$  is less than  $m$ , then  $\eta$  defined by (3.16) is larger than the value

$$\frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}$$

of  $\eta$  in Theorem 3.8. Thus Theorem 3.11 predicts faster convergence for the block method. However, the order constant in the bound is  $\tan \angle(x_1, v)$  not  $\tan \angle(x_1, U)$ . In general the latter will be smaller, so that initially the bound may be unrealistic.

- By the same trick used to derive the bound (3.12), we can get convergence rates for the vectors  $x_2, x_3, \dots$

The argument concerning multiple eigenvalues suggests that we can use block Krylov sequences to good effect with non-Hermitian matrices. However, there is no convergence theory analogous to Theorem 3.11 for non-Hermitian block Krylov sequences.

An alternative to block Krylov methods is to deflate converged eigenpairs, so that the Krylov sequence can converge to a second eigenpair corresponding to a repeated eigenvalue. The two approaches each have their advantages and drawbacks. Since deflation is vital in other ways to many algorithms, we will stress it in this volume. But the reader should keep in mind that there are important block variants of many of the algorithms we discuss here.

### 3.4. NOTES AND REFERENCES

#### Sources

Three major references for Krylov sequences are *The Symmetric Eigenvalue Problem* by B. N. Parlett [206], *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms* by Y. Saad [233], and *Eigenvalues of Matrices* by F. Chatelin [39]. In addition to most of the results treated in this subsection, these books contain historical comments and extensive bibliographies.

#### Krylov sequences

According to Householder [121, §6.1], Krylov [153, 1931] introduced the sequence named for him to compute the characteristic polynomial of a matrix. Specifically, if the Krylov sequence terminates at  $\ell$  [see (3.1)], then there are constants  $\alpha_0, \dots, \alpha_{\ell-1}$  such that

$$A^\ell u - \alpha_{\ell-1} A^{\ell-1} u - \cdots - \alpha_1 A u - \alpha_0 u = 0.$$

It can then be shown that the polynomial

$$\phi(\lambda) = \lambda^\ell - \alpha_{\ell-1} \lambda^{\ell-1} - \cdots - \alpha_0$$

divides the characteristic polynomial of  $A$ . The term “Krylov sequence” is due to Householder.

The solution of eigenvalue problems by forming and solving the characteristic equation is a thing of the past. What makes the Krylov sequence important today is that it is the basis of the Lanczos algorithm for eigenpairs and the conjugate gradient method for linear systems as well as their non-Hermitian variants. Since this book is concerned with eigensystems, the focus of this section has been on the spectral properties of Krylov sequences.

A word on notation. In much of the literature the eigenvectors of the matrix in question are denoted generically by  $u$  (perhaps because  $u$  is commonly used for an eigenfunction of a partial differential equation), and Krylov sequences are often based on a vector  $v$ . To maintain consistency with the first half of this book, eigenvectors are denoted by  $x$ , and, to avoid underusing a good letter, Krylov sequences are based on  $u$ .

#### Convergence (Hermitian matrices)

It seems that Krylov had no idea of the approximating power of the subspaces spanned by his sequence. The first hint of the power of Krylov sequences to approximate eigenvectors is found in Lanczos’s 1950 paper [157], in which he proposed the algorithm that bears his name (Lanczos himself called it the method of minimized iterations). In the conclusion to his paper, he states

Moreover, the good convergence of the method in the case of large dispersion makes it possible to operate with a small number of iterations, obtaining  $m$  successive eigenvalues and principle axes by only  $m+1$  iterations.

It is unfortunate that the first convergence results for Krylov spaces were directed toward the Lanczos algorithm, which is a combination of Krylov sequences and the Rayleigh–Ritz method to be treated in the next subsection. Kaniel [145, 1966], who gave the first analysis of the method, started by using Chebyshev polynomials to bound the eigenvalues produced by the Lanczos method and then using the eigenvalue bounds to produce bounds on the eigenvectors. In his important thesis Paige [196, 1971] followed Kaniel’s approach, correcting and refining his eigenvector bounds.

In 1980 Saad [229] showed that more natural bounds could be derived by first bounding the eigenvector approximations contained in the Krylov sequence itself and then analyzing the Rayleigh–Ritz procedure separately. This is the approach we have taken here. The result is Theorem 3.6, Theorem 3.8, and the related bound (3.12). This bound, incidentally, shows that there is nothing sacrosanct about Chebyshev polynomials. One is free to use them — or not use them — in combination with other polynomials.

### Convergence (non-Hermitian matrices)

The inequality (3.14) was stated without proof by Saad in [230, 1980], who gave a direct proof later [233, p. 204]. The result requires that  $A$  be diagonalizable. Theorem 3.10 removes this restriction. These results raise the possibility of using polynomials to get eigenvalue bounds. It is known that Chebyshev polynomials defined on a suitable ellipse asymptotically satisfy an inequality like (3.10). Thus when an eigenvalue lies outside an ellipse containing the rest of the spectrum, bounds analogous to those of Theorem 3.8 are possible. For more see [229].

### Block Krylov sequences

Block Krylov sequences — limited to small powers of  $A$  — were introduced by Culum and Donath [45, 1974] in connection with a restarted Lanczos algorithm. Block Krylov sequences in their full generality were introduced by Underwood in his Ph.D. thesis [280, 1975] to generalize the Lanczos algorithm (also see [92]). Underwood established eigenvalue bounds in the style of Kaniel, using a technique related to the construction in the proof of Theorem 3.11. Theorem 3.11 itself is due to Saad [229].

## 4. RAYLEIGH–RITZ APPROXIMATION

We have seen how a sequence of Krylov subspaces based on a matrix  $A$  can contain increasingly accurate approximations to an eigenspace of  $A$ . But it is one thing to know that an approximation exists and another to have it in hand. The purpose of this subsection is to describe and analyze the most commonly used method for extracting an approximate eigenspace from a larger subspace — the Rayleigh–Ritz method.

It is important to keep in mind that there are two levels of approximation going on here. First there is a best approximation to the desired eigenspace in the larger subspace. The Rayleigh–Ritz method approximates that best approximation. Con-

sequently, the analysis of the method is somewhat complicated, although its use in practice is straightforward. In this section we will develop the theory of the method and examine its practical implications. To keep the treatment simple, we will confine ourselves to the approximation of eigenvectors and will point out where the results generalize to eigenspaces.

We begin with a description of the method and then turn to its analysis. The analysis shows that unless a certain regularity condition is satisfied, the method may not produce satisfactory approximations. Consequently, the last two subsections are devoted to ways to circumvent this problem.

#### 4.1. RAYLEIGH-RITZ METHODS

The essential ingredients for the application of the Rayleigh–Ritz method is a matrix  $A$  and a subspace  $\mathcal{U}$  containing an approximate eigenspace of  $A$ . No other assumptions are made about  $\mathcal{U}$ . It may and often will be a Krylov subspace, but that is irrelevant to what follows.

We will motivate the method by the following theorem, which treats the case when  $\mathcal{U}$  contains an exact eigenspace of  $A$ .

**Theorem 4.1.** *Let  $\mathcal{U}$  be a subspace and let  $U$  be a basis for  $\mathcal{U}$ . Let  $V$  be a left inverse of  $U$  and set*

$$B = V^H A U.$$

*If  $\mathcal{X} \subset \mathcal{U}$  is an eigenspace of  $A$ , then there is an eigenpair  $(L, W)$  of  $B$  such that  $(L, UW)$  is an eigenpair of  $A$  with  $\mathcal{R}(UW) = \mathcal{X}$ .*

**Proof.** Let  $(L, X)$  be an eigenpair corresponding to  $\mathcal{X}$  and let  $X = UW$ . Then from the relation

$$AUX = UWL$$

we obtain

$$BW = V^H A UW = V^H UW L = WL,$$

so that  $(L, W)$  is an eigenpair of  $B$ . ■

Thus we can find exact eigenspaces contained in  $\mathcal{U}$  by looking at eigenpairs of the Rayleigh quotient  $B$ . By continuity, we might expect that when  $\mathcal{U}$  contains an approximate eigenspace  $\tilde{\mathcal{X}}$  of  $A$  there would be an eigenpair  $(\tilde{L}, \tilde{W})$  of  $B$  such that  $(\tilde{L}, A\tilde{W})$  is an approximate eigenpair of  $A$  with  $\mathcal{R}(A\tilde{W}) \cong \tilde{\mathcal{X}}$ . This suggests that to approximate an eigenpair corresponding to  $\mathcal{X}$  we execute the following procedure.

1. Let  $U$  be a basis for  $\mathcal{U}$  and let  $V^H$  be a left inverse of  $U$ .
  2. Form the Rayleigh quotient  $B = V^H A U$ .
  3. Let  $(M, W)$  be a suitable eigenpair of  $B$ .
  4. Return  $(M, UW)$  as an approximate eigenpair of  $A$ .
- (4.1)

We will call any procedure of this form a *Rayleigh–Ritz procedure*. We will call the pair  $(M, UW)$  a *Ritz pair*;  $M$  its *Ritz block*; and  $UW$  its *Ritz basis*. The space  $\mathcal{R}(UW)$  will be called the *Ritz space*. We will call  $W$  itself the *primitive Ritz basis*. When the concern is with eigenvalues and eigenvectors so that the Ritz pair can be written in the form  $(\lambda, Uw)$ , we will call  $\lambda$  a *Ritz value*,  $Uw$  a *Ritz vector*, and  $w$  a *primitive Ritz vector*.

## Two difficulties

The procedure listed in (4.1) has two difficulties. First, we have not specified how to choose the eigenpair  $(M, W)$  in statement 3. Second, we have no guarantee that the result approximates the desired eigenpair of  $A$ . Let us examine these difficulties in turn.

In practice, the eigenpair  $(M, W)$  is chosen by looking at the eigenvalues of  $B$ . Recall that in large eigenproblems we cannot expect to compute the entire spectrum. Instead we must specify the part of the spectrum that we are interested in — e.g., a group of eigenvalues with largest real part. It is therefore natural to collect the eigenvalues of  $B$  lying in the same chosen part of the spectrum and use them to compute  $(M, W)$  (see Theorem 1.5). This procedure, however, will work only if we can insure that there are eigenvalues in  $B$  that approximate those of  $A$  in the the desired part of the spectrum.

The second difficulty has no easy answer, as the following example shows.

**Example 4.2.** Let  $A = \text{diag}(0, 1, -1)$  and suppose we are interested in approximating the eigenpair  $(0, e_1)$ . Assume that by some method we have come up with the basis

$$U = \begin{pmatrix} 1 & 0 \\ 0 & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{pmatrix}.$$

Then  $U$  is its own left inverse and we may take

$$B = U^H A U = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Since  $B$  is zero, any nonzero vector  $p$  is an eigenvector corresponding to the part of the spectrum near zero. Since  $B$  itself gives no clue as to which vector to choose, we might end up with  $p = (1 \ 1)^T$ , in which case our approximate eigenvector becomes  $(1 \ \sqrt{2} \ \sqrt{2})^T$ , which is completely wrong. Thus the method can fail, even though the space  $\mathcal{U}$  contains an exact approximation to the desired eigenvector.

Of course, in real life we would not expect  $B$  to be exactly zero. But nearly zero is bad enough. For example, by perturbing  $U$  by a matrix of random normal deviates with standard deviation of  $10^{-4}$ , we obtained a matrix  $B = (U^T U)^{-1} U^T A U$  of the

form

$$B = \begin{pmatrix} 2.2088e-09 & -5.9808e-05 \\ -5.9827e-05 & 1.4503e-04 \end{pmatrix}. \quad (4.2)$$

The eigenvector corresponding to the smallest eigenvalue of  $B$  (the primitive Ritz vector) is  $(9.4110e-01 \ 3.3813e-01)^T$  and the corresponding Ritz vector is

$$(9.4120e-01 \ 2.3905e-01 \ 2.3909e-01)^T.$$

This comes nowhere near reproducing the order  $10^{-4}$  approximation to  $\mathbf{e}_1$  that we know lies in  $\mathcal{R}(U)$ .

The problem arises because the process of forming the Rayleigh quotient creates a spurious eigenvalue of  $B$  near zero. This suggests that we may have to postulate a separation between the approximations to the eigenvalues we desire and the other eigenvalues of the Rayleigh quotient. It turns out that such a postulate combined with the convergence of the eigenvalues in the Rayleigh quotient allows us to prove that the Ritz spaces converge as the approximations in  $U$  converge. Symbolically:

$$\begin{aligned} & \text{convergence of the desired eigenvalues} \\ + & \frac{\text{separation of the desired eigenvalues}}{\text{convergence of the Ritz space}} \\ = & \end{aligned} \quad (4.3)$$

We will flesh out this equation in the next subsection.

### **Rayleigh–Ritz and generalized eigenvalue problems**

The algorithm (4.1) presupposes that  $V$  is a left inverse of  $U$ . If it is not, we can convert it to a left inverse by replacing it with  $\hat{V} = V(V^H U)^{-H}$ , provided of course that  $V^H U$  is nonsingular. If  $(\mu, w)$  is a primitive Ritz pair, then

$$(V^H U)^{-1} V^H A U w = \mu w,$$

and hence

$$(V^H A U)w = \mu(V^H U)w. \quad (4.4)$$

Thus the primitive Ritz pairs are the eigenpairs of the generalized eigenvalue problem (4.4).

### **The orthogonal Rayleigh–Ritz procedure**

In its full generality the algorithm (4.1) is called an *oblique Rayleigh–Ritz method* because the subspaces spanned by  $U$  and  $V$  can differ. This form of the method has important applications (e.g., to the bi-Lanczos algorithm). But in the overwhelming majority of applications  $U$  is taken to be orthonormal and  $V$  is taken equal to  $U$ . In

addition, the primitive Ritz basis  $W$  is taken to be orthonormal, so that the Ritz basis  $\hat{X} = UW$  is also orthonormal. We call this procedure the *orthogonal Rayleigh–Ritz method*.

When  $A$  is Hermitian, the orthogonal Rayleigh–Ritz method has the obvious advantage that it preserves symmetry throughout the procedure. However, the following theorem shows that for any matrix the orthogonal method also gives minimum residuals.

**Theorem 4.3.** *Let  $(M, \hat{X})$  be an orthogonal Rayleigh–Ritz pair with respect to  $U$ . Then*

$$R = A\hat{X} - \hat{X}M$$

*is minimal in any unitarily invariant norm.*

**Proof.** By Theorem 2.6 all we need to show is that  $M = \hat{X}^H A \hat{X}$ . But if  $W$  is the (orthonormal) primitive Ritz basis, then  $M = W^H B W$  (see Theorem 1.2). Since  $\hat{X} = UW$ , we have

$$\hat{X}^H A X = W^H U^H A U W = W^H B W = M. \blacksquare$$

## 4.2. CONVERGENCE

This subsection is concerned with the accuracy of the approximations produced by the Rayleigh–Ritz procedure. We will treat the orthogonal variant and for simplicity will only consider the case where we are attempting to approximate a simple eigenvalue and its eigenvector—by far the most common use of the method.

### Setting the scene

Consider the use of the Rayleigh–Ritz procedure in real life. We want to approximate the eigenpair  $(\lambda, x)$  of  $A$ . By some method—call it the U-machine—we obtain an orthonormal basis  $U_\theta$  for which  $\theta = \angle(x, U_\theta)$  is small. We apply the Rayleigh–Ritz process to obtain a Ritz pair  $(\mu_\theta, U_\theta p_\theta)$ . If the pair is unsatisfactory, we return to our U-machine, ask it to grind out another  $U_\theta$  with a smaller value of  $\theta$ , and try the Rayleigh–Ritz process again. To justify this process, we must show that the process in some sense converges as  $\theta \rightarrow 0$ .

We will treat this problem in two stages. First we will show that the Rayleigh quotient

$$B_\theta = U_\theta^H A U_\theta \tag{4.5}$$

contains a Ritz value  $\mu_\theta$  that converges to  $\lambda$ . We will then treat the more delicate question of when the Ritz vector  $x_\theta$  converges.

### Convergence of the Ritz value

Our approach to the convergence of Ritz values is through a backward perturbation theorem.

**Theorem 4.4.** *Let  $B_\theta$  be the Rayleigh quotient (4.5). Then there is a matrix  $E_\theta$  satisfying*

$$\|E_\theta\|_2 \leq \frac{\sin \theta}{\sqrt{1 - \sin^2 \theta}} \|A\|_2$$

such that  $\lambda$  is an eigenvalue of  $B_\theta + E_\theta$ .

**Proof.** Let  $(U_\theta \ U_\perp)$  be unitary and set

$$y = U_\theta^H x \quad \text{and} \quad z = U_\perp^H x.$$

Now  $U_\perp$  spans the orthogonal complement of the column space of  $U_\theta$ . Hence  $\|z\|_2 = \sin \theta$  and hence  $\|y\|_2 = \sqrt{1 - \sin^2 \theta}$ . Since  $Ax - \lambda x = 0$ , we have

$$U_\theta^H A (U_\theta \ U_\perp) \begin{pmatrix} U_\theta^H \\ U_\perp^H \end{pmatrix} x - \lambda U_\theta^H x = 0,$$

or

$$B_\theta y + U_\theta^H A U_\perp z - \lambda y = 0. \quad (4.6)$$

Let  $\hat{y} = y / \sqrt{1 - \sin^2 \theta}$  be the normalized value of  $y$ . If

$$r = B_\theta \hat{y} - \lambda \hat{y},$$

it follows from (4.6) that

$$\|r\|_2 \leq \frac{\sin \theta}{\sqrt{1 - \sin^2 \theta}} \|A\|_2.$$

Now define

$$E_\theta = -r \hat{y}^H.$$

Then clearly  $\|E_\theta\|_2 = \|r\|_2$  and  $(A + E_\theta)\hat{y} = \lambda \hat{y}$ . ■

We can now apply Elsner's theorem (Theorem 3.1, Chapter 1) to give the following corollary.

**Corollary 4.5.** *There is an eigenvalue  $\mu_\theta$  of  $B_\theta$  such that*

$$|\mu_\theta - \lambda| \leq 4(2\|A\|_2 + \|E_\theta\|_2)^{1-\frac{1}{m}} \|E_\theta\|_2^{\frac{1}{m}}, \quad (4.7)$$

where  $m$  is the order of  $B_\theta$ .

Two comments.

- Assume the number  $m$  of columns of  $U_\theta$  is bounded. Then the corollary assures us that we only have to wait until  $\theta$  is small enough for a Ritz value to emerge that is near  $\lambda$ . The factor  $\|E_\theta\|_2^{\frac{1}{m}}$  in the bound (4.7) suggests that we may have to wait a long time. But as we shall see later, under a certain separation condition, the convergence will be  $O(\theta)$ .
- If  $A$  is Hermitian, we may replace the right-hand side of (4.7) by  $\|E\|_2$  (see Theorem 3.8, Chapter 1). In this case we do not have to assume  $m$  is bounded. This has important implications for the Lanczos algorithm, which uses a sequence of Krylov spaces in which  $m$  can become rather large.

### Convergence of Ritz vectors

We now turn to the convergence of Ritz vectors. The key is the following theorem, which relates Ritz vectors to the angle  $\theta$ .

**Theorem 4.6.** *In the above notation, let  $(\mu_\theta, w_\theta)$  be a primitive Ritz pair and let the matrix  $(w_\theta \ W_\theta)$  be unitary, so that*

$$\begin{pmatrix} w_\theta^H \\ W_\theta^H \end{pmatrix} B_\theta (w_\theta \ W_\theta) = \begin{pmatrix} \mu_\theta & h_\theta^H \\ 0 & N_\theta \end{pmatrix}.$$

Then

$$\sin \angle(x, U_\theta w_\theta) \leq \sin \theta \sqrt{1 + \frac{\|h_\theta\|_2^2}{\text{sep}(\lambda, N_\theta)^2}}. \quad (4.8)$$

For the proof of the theorem see the notes and references.

To apply the theorem, suppose we have chosen a Ritz pair (whose existence is guaranteed by Corollary 4.5) for which  $\mu_\theta \rightarrow \lambda$ . Then by the continuity of  $\text{sep}$  (Theorem 2.11), we have

$$\text{sep}(\lambda, N_\theta) \geq \text{sep}(\mu_\theta, N_\theta) - |\mu_\theta - \lambda|.$$

Consequently, if  $\text{sep}(\mu_\theta, N_\theta)$  remains bounded below, so does the  $\text{sep}(\lambda, N_\theta)$  in (4.8), at least in the limit. Since  $\|h_\theta\|_2 \leq \|A\|_2$ , we have  $\sin \angle(x, U_\theta p_\theta) \rightarrow 0$  along with  $\theta$ . Thus we have the following corollary.

**Corollary 4.7.** *Let  $(\mu_\theta, U_\theta w_\theta)$  be a Ritz pair for which  $\mu_\theta \rightarrow \lambda$ . If there is a constant  $\alpha > 0$  such that*

$$\text{sep}(\mu_\theta, N_\theta) \geq \alpha > 0, \quad (4.9)$$

*then asymptotically*

$$\sin \angle(x, U_\theta w_\theta) \lesssim \sin \theta \sqrt{1 + \frac{\|A\|_2^2}{\alpha^2}}.$$

This corollary justifies the statement (4.3) to the effect that eigenvalue convergence plus separation equals eigenvector convergence. Specifically, it contains the convergence hypothesis that  $\mu_\theta$  approaches  $\lambda$  and the separation hypothesis that the quantity  $\text{sep}(\mu_\theta, N_\theta)$  is uniformly bounded below. The conclusion is that the sine of the angle between the eigenvector  $x$  and the Ritz vector  $U_\theta p_\theta$  approaches zero.

We call the condition (4.9) the *uniform separation condition*. What is unusual for results of this kind is that it consists entirely of computed quantities and can be monitored. For example, in the second part of Example 4.2 the separation of the eigenvalues of  $B$  is about  $1.9 \cdot 10^{-4}$ , which accounts for the failure of the Ritz vector to reproduce the eigenvector  $e_1$ , even though it is present in  $U$  to accuracy of order  $10^{-4}$ .

### Convergence of Ritz values revisited

The bound in Corollary 4.5 suggests that the convergence of the Ritz value can be quite slow. We will now use the fact that the Ritz vector converges to derive a more satisfactory result.

Let  $(\mu_\theta, x_\theta)$  be the Ritz approximation to  $(\lambda, x)$ . Then by construction,

$$\mu_\theta = x_\theta^H A x_\theta$$

is the Rayleigh quotient formed from  $x_\theta$  and  $A$ . If we write  $x_\theta = \gamma x + \sigma y$ , where  $y \perp x$  and  $\|y\|_2 = 1$ , then  $|\gamma| = \cos \angle(x_\theta, x)$  and  $|\sigma| = \sin \angle(x_\theta, x)$ . If the uniform separation is satisfied, by Corollary 4.7 we have  $|\sigma| = \sin \angle(x_\theta, x) = O(\theta)$ .

By direct computation of the Rayleigh quotient  $x_\theta^H A x_\theta$  we find that

$$\mu_\theta = |\gamma|^2 \lambda + \bar{\gamma} \sigma x^H A y + |\sigma|^2 y^H A y$$

(remember  $y^H A x = \lambda y^H x = 0$ ). Hence

$$\begin{aligned} |\mu_\theta - \lambda| &= |(|\gamma|^2 - 1)\lambda + \bar{\gamma} \sigma x^H A y + |\sigma|^2 y^H A y| \\ &\leq |\sigma|^2 |\lambda| + |\gamma| |\sigma| |x^H A y| + |\sigma|^2 |y^H A y| \\ &\leq |\sigma|(1 + |\sigma|) \|A\|_2 \\ &= O(\theta). \end{aligned} \tag{4.10}$$

Thus the Ritz value converges at least as fast as the eigenvector approximation of  $x$  in  $\mathcal{R}(U_\theta)$ .

### Hermitian matrices

When  $A$  is Hermitian, we can say more. Let the eigenvalues of  $A$  be ordered so that

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

and let the Ritz values — the eigenvalues of the Hermitian matrix  $U^H A U$  — be ordered so that

$$\mu_1 \geq \mu_2 \geq \cdots \geq \mu_m.$$

Then by the interlacing theorem (Theorem 3.6, Chapter 1),

$$\lambda_1 \geq \mu_1 \geq \lambda_2 \geq \mu_2 \geq \cdots \geq \lambda_{n-1} \geq \mu_{n-1} \geq \lambda_n.$$

Consequently, if  $\mu_i$  converges to  $\lambda_i$  then it must converge from below. Similarly, if  $\mu_i$  converges to  $\lambda_{n-m+i}$  it must converge from above.

The Ritz values converge more quickly in the Hermitian case. For in (4.10) we have  $x^H A y = 0$ , and hence

$$|\mu_\theta - \lambda| = O(\theta^2).$$

### Convergence of Ritz blocks and bases

Theorem 4.4, Corollary 4.5, Theorem 4.6, and Corollary 4.7 all have natural analogues for Ritz blocks and Ritz bases, and, with the exception of Corollary 4.7, the proofs are essentially the same. Consequently, we can use the Rayleigh–Ritz method to approximate eigenspaces with the same assurance that we use it to approximate eigenvectors. For more see the notes and references.

### Residuals and convergence

The results of this subsection have been cast in terms of the sine of the angle  $\theta$  between an eigenvector  $x$  and its approximating subspace  $\mathcal{U}$ . Although these results give us insight into the behavior of the Rayleigh–Ritz method, in the last analysis we cannot know  $\theta$  and hence cannot compute error bounds. Thus, as always, we must look to the residual as an indication of convergence. The following residual bound actually applies to any approximate eigenpair.

**Theorem 4.8.** *Let  $A$  have the spectral representation*

$$A = \lambda x y^H + X L Y^H, \tag{4.11}$$

where  $\|x\|_2 = 1$  and  $Y$  is orthonormal. Let  $(\mu, \tilde{x})$  be an approximation to  $(\lambda, x)$  and let

$$\rho = \|A\tilde{x} - \mu\tilde{x}\|_2.$$

Then

$$\sin \angle(\tilde{x}, x) \leq \frac{\rho}{\text{sep}(\mu, L)} \leq \frac{\rho}{\text{sep}(\lambda, L) - |\mu - \lambda|}. \tag{4.12}$$

**Proof.** Because (4.11) is a spectral decomposition,  $Y$  is an orthonormal basis for the orthogonal complement of the space spanned by  $x$ . Thus  $\sin \angle(\tilde{x}, x) = Y^H \tilde{x}$ . Let  $r = A\tilde{x} - \mu x$ . Then

$$Y^H r = Y^H A\tilde{x} - \mu Y^H x = (L - \mu I) Y^H \tilde{x}.$$

It follows that

$$\sin \angle(\tilde{x}, x) = \|(L - \mu I)^{-1} Y^H r\|_2 \leq \text{sep}(\mu, L) \|r\|_2,$$

from which the first inequality in (4.12) follows. The second inequality follows from the first and the fact that  $\text{sep}(\mu, L) \geq \text{sep}(\lambda, L) - |\mu - \lambda|$  (see Theorem 2.11). ■

Since  $\lambda$  is assumed to be simple, this theorem says that:

A sufficient condition for  $\tilde{x}$  to converge to  $x$  is for  $\mu$  to converge to  $\lambda$  and for the residual to converge to zero. (4.13)

Unfortunately, as Example 4.2 shows, spurious Ritz values can cause the residual to fail to converge. We therefore turn to two alternatives to Ritz vectors.

### 4.3. REFINED RITZ VECTORS

#### Definition

By hypothesis, the space  $\mathcal{U}_\theta$  contains increasingly accurate approximations  $\check{x}_\theta$  to the vector in the simple eigenpair  $(\lambda, x)$ . By Corollary 4.5, there is a Ritz value  $\mu_\theta$  that is approaching  $\lambda$ . It follows that the residual  $A\check{x}_\theta - \mu_\theta \check{x}_\theta$  approaches zero. Hence if we define  $\hat{x}_\theta$  to be the vector whose residual  $A\hat{x}_\theta - \mu_\theta \hat{x}_\theta$  has the smallest norm, that residual must approach zero. This suggests the following definition.

**Definition 4.9.** Let  $\mu_\theta$  be a Ritz value associated with  $\mathcal{U}_\theta$ . A **REFINED RITZ VECTOR** is a solution of the problem

$$\begin{aligned} & \text{minimize} && \|A\hat{x}_\theta - \mu_\theta \hat{x}_\theta\|_2 \\ & \text{subject to} && \hat{x}_\theta \in \mathcal{U}_\theta, \|\hat{x}_\theta\|_2 = 1. \end{aligned} \quad (4.14)$$

We will worry about the computation of refined Ritz vectors later. First let us return to Example 4.2, which showed how the Rayleigh–Ritz method could fail to deliver an accurate approximation to an accurate eigenvector contained in a subspace. Specifically, if we solve the problem using the smallest eigenvalue of  $B$  in (4.2) we get the vector

$$\hat{x} = \begin{pmatrix} -1.0000e+00 \\ 4.2290e-05 \\ -4.2300e-05 \end{pmatrix},$$

which is an excellent approximation to the original eigenvector  $e_1$  of  $A$ .

#### Convergence of refined Ritz vectors

We now turn to the convergence of refined Ritz vectors. We have argued informally that the convergence of the refined Ritz value  $\mu_\theta$  to the simple eigenvalue  $\lambda$  is sufficient to insure the convergence of the refined Ritz vector. The following theorem gives quantitative substance to this assertion.

**Theorem 4.10.** Let  $A$  have the spectral representation

$$A = \lambda xy^H + XLY^H,$$

where  $\|x\|_2 = 1$  and  $Y$  is orthonormal. Let  $\mu_\theta$  be a Ritz value and  $\hat{x}_\theta$  the corresponding refined Ritz vector. If  $\text{sep}(\lambda, L) - |\mu_\theta - \lambda| > 0$ , then

$$\sin \angle(x, \hat{x}_\theta) \leq \frac{\|A - \mu_\theta I\|_2 \sin \theta + |\lambda - \mu_\theta|}{\sqrt{1 - \sin^2 \theta} [\text{sep}(\lambda, L) - |\lambda - \mu_\theta|]}. \quad (4.15)$$

**Proof.** Let  $U$  be an orthonormal basis for  $U$  and let  $x = y + z$ , where  $y = UU^Hx$ . Then  $\|z\|_2 = \|U^Hx\|_2 = \sin \theta$ . Moreover since  $y$  and  $z$  are orthogonal,  $\|y\|_2^2 = 1 - \sin^2 \theta$ . Let

$$\hat{y} = \frac{y}{\sqrt{1 - \sin^2 \theta}}$$

be the normalized value of  $y$ .

We have

$$\begin{aligned} (A - \mu_\theta I)\hat{y} &= \frac{(A - \mu_\theta I)y}{\sqrt{1 - \sin^2 \theta}} \\ &= \frac{(A - \mu_\theta I)(x - z)}{\sqrt{1 - \sin^2 \theta}} \\ &= \frac{(\lambda - \mu_\theta I)x - (A - \mu_\theta I)z}{\sqrt{1 - \sin^2 \theta}}. \end{aligned}$$

Hence

$$\|(A - \mu_\theta I)\hat{y}\|_2 \leq \frac{|\lambda - \mu_\theta| - \|A - \mu_\theta I\|_2 \sin \theta}{\sqrt{1 - \sin^2 \theta}}.$$

By the definition of a refined Ritz vector we have

$$\|(A - \mu_\theta I)\hat{x}\|_2 \leq \frac{|\lambda - \mu_\theta| - \|A - \mu_\theta I\|_2 \sin \theta}{\sqrt{1 - \sin^2 \theta}}.$$

The result now follows from Theorem 4.8. ■

Here are two comments on this theorem.

- By Corollary 4.5 we know that  $\mu_\theta \rightarrow \lambda$ . It follows from (4.15) that  $\sin \angle(x, \hat{x}_\theta) \rightarrow 0$ . In other words, refined Ritz vectors, unlike ordinary Ritz vectors, are guaranteed to converge.
- Although the Ritz value  $\mu_\theta$  was used to define the refined Ritz vector  $\hat{x}_\theta$ , the Rayleigh quotient  $\hat{\mu}_\theta = \hat{x}_\theta^T A \hat{x}_\theta$  is likely to be a more accurate approximation to  $\lambda$ . Moreover, by Theorem 2.6, the residual norm  $\|A\hat{x}_\theta - \hat{\mu}_\theta \hat{x}_\theta\|_2$  will be optimal.

### The computation of refined Ritz vectors

The computation of a refined Ritz vector amounts to solving the optimization problem

$$\begin{aligned} \text{minimize } & \|A\hat{x} - \mu\hat{x}\|_2 \\ \text{subject to } & \hat{x} \in \mathcal{U}, \|\hat{x}\|_2 = 1. \end{aligned}$$

Let  $U$  be an orthonormal basis for  $\mathcal{U}$ . Then any vector of 2-norm one in  $\mathcal{U}$  can be written in the form  $Uz$ , where  $\|z\|_2 = 1$ . Thus we need to solve the problem

$$\begin{aligned} \text{minimize } & \|(A - \mu I)Uz\|_2 \\ \text{subject to } & \|z\|_2 = 1. \end{aligned} \tag{4.16}$$

By Theorem 3.2, Chapter 3, the solution of this problem is the right singular vector of  $(A - \mu I)U$  corresponding to its smallest singular value. Thus we can compute refined Ritz vectors by computing the singular value decomposition of  $(A - \mu I)U$ . This suggests the following algorithm.

1.  $V = AU$
2.  $W = V - \mu U$
3. Compute the smallest singular value of  $W$  and its right singular vector  $z$
4.  $\hat{x} = Uz$

It is natural to compare this process with the computation of an ordinary Ritz vector. We will assume that  $U$  is  $n \times p$ , where  $n \gg p$ . The computation is dominated by the formation of  $V$  and the computation of the singular value decomposition.

The formation of  $V$  requires  $p$  multiplications of a vector by  $A$ . This computation must also be done by the Rayleigh–Ritz method. As with the formation of block Krylov sequences, it will often happen that the formation of  $AU$  takes less time than  $p$  individual matrix–vector multiplications.

The next nontrivial step is the computation of the singular value decomposition of  $W$ , which requires  $O(np^2)$  operations. On the other hand, the Rayleigh–Ritz method must calculate  $H = U^H V$ , which is also an  $O(np^2)$  process, albeit with a smaller order constant. Thus, the computation of a single refined Ritz vector and a single ordinary Ritz vector require the same order of work.

The situation is different when several vectors, say  $m$ , are to be computed. In this case, we will have to compute  $m$  singular value decompositions to get the refined vectors. On the other hand, the Rayleigh–Ritz method requires only that we compute the eigensystem of  $H$ . Thus *all* Ritz vectors can be calculated with  $O(np^2)$  operations, while the computation of *each* refined Ritz vector requires  $O(np^2)$  operations. Hence the computation of  $m$  refined Ritz vectors will take longer than the computation of  $m$  ordinary Ritz vectors by a factor of at least  $m$ .

A cure for this problem is to use the cross-product algorithm (Algorithm 3.1, Chapter 3) to compute the singular value decompositions. Specifically, the cross-product

matrix corresponding to  $AU - \mu U$  is

$$\begin{aligned} B_\mu &= (AU - \mu U)^H (AU - \mu U) \\ &= (AU)^H (AU) - \mu [(U^H AU) + (U^H AU)^H] + \mu^2 I. \end{aligned}$$

If in the notation of (4.17) we precompute the matrices

$$C_0 = V^H V \quad \text{and} \quad C_1 = V^H U + U^H V,$$

then for any  $\mu$  we can form

$$B_\mu = \mu^2 I - \mu C_1 + C_0$$

at negligible cost. Thus the cost of computing an additional refined Ritz vectors is the same as computing a spectral decomposition of a matrix of order  $p$ .

Our analysis of the cross-product algorithm applies here. Specifically, denote the singular vectors of  $AU - \mu I$  by  $\sigma_1 \geq \dots \geq \sigma_p$ . If  $\mu$  is an accurate approximation to an eigenvalue  $\lambda$ , we expect  $\sigma_p$  to be small. Then by (3.20), Chapter 3, with  $i = p$  and  $j = p-1$ , we see that the sine of the angle between the computed vector and the true vector is of order  $(\sigma_{p-1}/\sigma_1)^2 \epsilon_M$ . If  $\sigma_{p-1}$  is not too small — something we can check during the calculation — then the refined Ritz vector computed by the cross-product algorithm will be satisfactory.

Finally, we note that if  $\mathcal{U}$  is a Krylov subspace, the computation of refined Ritz values can be considerably abbreviated, as we shall see in §1.4, Chapter 5.

#### 4.4. HARMONIC RITZ VECTORS

The eigenvalue 0 of  $A$  in Example 4.2 lies between the eigenvalues 1 and  $-1$  and for this reason is called an *interior eigenvalue* of  $A$ . Interior eigenvalues are particularly susceptible to being impersonated by Rayleigh quotients of the form  $u^H A u$ , where  $u$  is not near an eigenvector of  $A$ . In fact, such an impersonation is the cause of the failure of the Rayleigh–Ritz method in the example. The use of refined Ritz vectors solves the problem by insuring that the residual  $Ax - \mu x$  is small — something that is not true of the spurious Ritz vector. Unfortunately, the quality of the refined Ritz vector depends on the accuracy of  $\mu$ , and, as we have seen, each refined Ritz vector must be calculated independently from its own distinct value of  $\mu$ .

Harmonic Ritz vectors, on the other hand, can provide a complete set of approximating pairs while at the same time protecting pairs near a shift  $\kappa$  against impersonators. There is no really good way of introducing them other than to begin with a definition and then go on to their properties.

**Definition 4.11.** Let  $\mathcal{U}$  be a subspace and  $U$  be an orthonormal basis for  $\mathcal{U}$ . Then  $(\kappa + \delta, Uw)$  is a HARMONIC RITZ PAIR WITH SHIFT  $\kappa$  if

$$U^H (A - \kappa I)^H (A - \kappa I) U w = \delta U^H (A - \kappa I)^H U w. \quad (4.18)$$

We will refer to the process of approximating eigenpairs by harmonic Ritz pairs as the harmonic Rayleigh–Ritz method. This method shares with the ordinary Rayleigh–Ritz method the property that it retrieves eigenvectors that are exactly present in  $\mathcal{U}$ . Specifically, we have the following result.

**Theorem 4.12.** *Let  $(\lambda, x)$  be an eigenpair of  $A$  with  $x = Uw$ . Then  $(\lambda, Uw)$  is a harmonic Ritz pair.*

**Proof.** Calculating the left- and right-hand sides of (4.18), we find

$$(\lambda - \kappa)U^H(A - \kappa I)^H Uw = \delta U^H(A - \kappa I)^H Uw.$$

Hence  $Uw$  is an eigenvector of (4.18) with eigenvalue  $\delta = \lambda - \kappa$ ; i.e.,  $(\lambda, Uw)$  is a harmonic Ritz pair. ■

This theorem leads us to expect that if  $x$  is approximately represented in  $\mathcal{U}$ , then the harmonic Rayleigh–Ritz, like its unharmonious counterpart, will produce an approximation to  $x$ . Unfortunately, as of this writing the method has not been analyzed. But it is easy to see informally that it protects eigenpairs near  $\kappa$  from imposters. Specifically, on multiplying (4.18) by  $w^H$ , we get  $\|(A - \kappa I)Uw\|_2^2 \leq |\delta| \|(A - \kappa I)Uw\|_2$ , or

$$\|(A - \kappa I)Uw\|_2 \leq |\delta|. \quad (4.19)$$

Consequently, if any harmonic Ritz value is within  $\delta$  of  $\kappa$  the pair must have a residual norm (with respect to  $\kappa$ ) bounded by  $|\delta|$ . For example, if  $\kappa \rightarrow \lambda$  and  $\delta \rightarrow 0$ , then it eventually becomes impossible for the harmonic Rayleigh–Ritz method to produce impersonators.

It should be stressed that  $\lambda$  does not have to be very near  $\kappa$  for this screening to occur. If we return to Example 4.2 and compute the harmonic Ritz vector associated with the eigenvalue zero setting  $\kappa = 0.3$ , we obtain the following excellent approximation to  $\mathbf{e}_1$ :

$$\begin{pmatrix} 1.0001e+00 \\ -6.3447e-05 \\ 2.1155e-05 \end{pmatrix}.$$

### Computation of harmonic Ritz pairs

When  $U$  contains a good approximation to an eigenvector whose eigenvalue lies near  $\kappa$ ,  $(A - \kappa I)U$  in (4.18) will have a small singular value which will be squared when we form  $U^H(A - \kappa I)^H(A - \kappa I)U$ , which can result in subsequent inaccuracies (see §3.2, Chapter 3). To circumvent this problem, we can compute the QR factorization

$$(A - \kappa I)U = QR. \quad (4.20)$$

Then (4.18) assumes the form

$$R^H R w = \delta R^H Q^H U w. \quad (4.21)$$

Hence

$$(Q^H U) w = \delta^{-1} R w. \quad (4.22)$$

This eigenproblem can be solved by the QZ algorithm (§4.3, Chapter 2).

An alternative is to work with the ordinary eigenproblem

$$(R^{-1} Q^H U) w = \delta^{-1} w. \quad (4.23)$$

In this case a small singular value in  $R$  will cause  $R^{-1}$  and hence  $R^{-1} Q^H U$  to be large. Fortunately, we are interested in harmonic Ritz values that are near  $\kappa$ , i.e., values for which  $\delta$  is small. Thus we want eigenpairs corresponding to large eigenvalues in (4.23). Although the eigenvalue itself will not necessarily be accurately computed (the loss of accuracy occurs in forming  $R$ , not in forming  $R^{-1} Q^H U$ ), if the eigenvalue is sufficiently well separated from its neighbors the eigenvector  $p$  will be well determined.

The harmonic Ritz values are useful primarily in locating the harmonic Ritz vectors we need. When it comes to approximating the associated eigenvalue, we should use the ordinary Rayleigh quotient  $w^H U A U w$ , which minimizes the residual norm. In computing this Rayleigh quotient we can avoid a multiplication by  $A$  by using quantities already calculated. Specifically, from (4.20)

$$U^H A U = U^H Q R + \kappa I.$$

Hence

$$w^H U^H A U w = (w^H U^H)(Q R w) + \kappa = \hat{x}^H (Q R w) + \kappa,$$

where  $\hat{x}$  is the harmonic Ritz vector.

When  $A$  is Hermitian, both the eigenproblems (4.22) and (4.23), which are essentially nonsymmetric, can give complex harmonic Ritz values in the presence of rounding error. However, we can use the same technique that we used for the S/PD problem (§4, Chapter 3). Specifically, we can transform (4.23) into the Hermitian eigenproblem

$$(Q^H U R^{-1})(R w) = \delta^{-1}(R w) \quad (4.24)$$

for the vector  $R w$ . This approach has the slight disadvantage that having computed a solution  $g$  we must solve the system  $R w = g$  to retrieve the primitive harmonic Ritz vector. However, the usual Rayleigh quotient has the simpler form

$$w^H U^H A U w = \hat{x}^H Q g + \kappa.$$

## 4.5. NOTES AND REFERENCES

### Historical

In 1899, Lord Rayleigh [215] showed how to compute the fundamental frequency of a vibrating system. His approach was to approximate the fundamental mode by a linear combination of functions. He chose the linear combination  $x$  for which the quantity  $\lambda = x^T A x / x^T x$  was minimized and took  $\lambda$  for the fundamental frequency.

In 1909 Ritz [217] proposed a general method for solving continuous problems that could be phrased as the minimization of a functional. His idea was to choose a suitable subspace (e.g., of low-order polynomials) and determine coefficients to minimize the functional. His example of an eigenproblem — a second-order differential equation — does not, strictly speaking, fit into his general scheme. Although the mode corresponding to the smallest eigenvalue minimizes a functional, the other eigenvalues are found from the stationary points.

The Rayleigh–Ritz method is sometimes called a Galerkin method or a Galerkin–Petrov method. These methods solve a linear system  $Ax = b$  in the operator  $A$  by choosing two subspaces  $\mathcal{U}$  and  $\mathcal{V}$  and choosing  $x$  to satisfy the conditions

1.  $x \in \mathcal{U}$
  2.  $Ax - b \perp \mathcal{V}$ .
- (4.25)

If  $\mathcal{U} = \mathcal{V}$  the method is called a Galerkin method; if not it is called a Galerkin–Petrov method. Now let  $\mathcal{V}$  be the subspace spanned by  $\mathcal{V}$  in (4.1). Then any Ritz pair  $(\mu, x)$  satisfies the conditions

1.  $x \in \mathcal{U}$
  2.  $Ax - \lambda x \perp \mathcal{V}$ .
- (4.26)

The similarity of (4.25) and (4.26) explains the nomenclature. The subject is not well served by this proliferation of unnecessary terminology, and here we subsume it all under the terms Rayleigh–Ritz method, Ritz pair, etc.

The Rayleigh–Ritz method is also called a projection method because a Ritz pair  $(\mu, x)$  is an eigenpair of the matrix  $P_{\mathcal{V}} A P_{\mathcal{U}}$ , where  $P_{\mathcal{V}}$  and  $P_{\mathcal{U}}$  are the orthogonal projections onto  $\mathcal{U}$  and  $\mathcal{V}$ .

### Convergence

For earlier work on the convergence of the Rayleigh–Ritz method see the books by Parlett [206] for Hermitian matrices and by Saad [233] for both Hermitian and non-Hermitian matrices. The convergence theory developed here is essentially that of Jia and Stewart for Ritz vectors [133] and for Ritz bases [134]. Theorem 4.6 is a generalization by Stewart of a theorem of Saad [231] for Hermitian matrices (also see [148]) and can in turn be extended to Ritz bases. For a proof see [268].

### Refined Ritz vectors

Refined Ritz vectors whose residual norms are not necessarily minimal were introduced by Ruhe [220] in connection with Krylov sequence methods for the general-

ized eigenvalue problem. Refined Ritz vectors in the sense used here are due to Jia [131], who showed that they converge when the Ritz vectors fail to converge. The proof given here may be found in [134]. The use of the cross-product algorithm to compute refined Ritz vectors was suggested in [132].

### Harmonic Ritz vectors

For a symmetric matrix, the (unshifted) harmonic Ritz values are zeros of certain polynomials associated with the MINRES method for solving linear systems. They are called “harmonic” because they are harmonic averages of the Ritz values of  $A^{-1}$ . For a survey and further references see the paper [201] by Paige, Parlett, and van der Vorst, who introduced the appellation harmonic.

Morgan [179] introduced the harmonic Ritz vectors as good approximations to interior eigenvectors. His argument (still for symmetric matrices) was that since eigenvalues near  $\kappa$  become exterior eigenvalues of  $(A - \kappa I)^{-1}$ , to compute their eigenvectors we should apply the Rayleigh–Ritz method with basis  $U$  to  $(A - \kappa I)^{-1}$ . However, we can eliminate the need to work with this costly matrix by instead working with the basis  $(A - \kappa I)U$ . This leads directly to equation (4.18). But the resulting Ritz vector  $(A - \kappa I)Up$  will be deficient in components in eigenvalues lying near  $\kappa$ . Hence we should return the vectors  $Up$ , which are not Ritz vectors, at least in the usual sense for symmetric matrices. Morgan also recommended replacing the harmonic Ritz value with the ordinary Rayleigh quotient. Since Morgan was working with Hermitian matrices, he was able to establish some suggestive results about the process.

Harmonic Ritz values for non-Hermitian matrices were described by Freund [78], again in connection with iterative methods for solving linear systems. Harmonic Ritz vectors were introduced by Sleijpen and van der Vorst [240], who characterized them (in our notation) by the Galerkin–Petrov condition

1.  $x \in \mathcal{U}$
2.  $(A - \kappa I)x - \delta x \perp (A - \kappa I)\mathcal{U}$ .

The use of the inequality (4.19) to justify the use of harmonic Ritz vectors is new.

The explicit reduction of the harmonic eigenproblem to the forms (4.22), (4.23), and the symmetric form (4.24) is new, although the symmetric form is implicit in the implementation of the Jacobi–Davidson algorithm in [243]. Another way of looking at these reductions is that when  $\mathcal{R}(U)$  contains an eigenvector of  $A$  corresponding to eigenvalue  $\tau$ , the pencil in the original definition (4.18) is not regular, while the transformed pencil (4.22) is.

For the generalized eigenvalue problem  $Ax = \lambda Bx$ , a natural generalization of a harmonic Ritz pair is a solution of

$$U^H(A - \kappa B)^H(A - \kappa B)Up = \delta U^H(A - \kappa B)^HBUp,$$

which was first introduced in [74]. An argument similar to that leading to (4.19) shows that this method also discourages imposters.

# 5

---

## KRYLOV SEQUENCE METHODS

In this chapter and the next we will treat algorithms for large eigenproblems. There are two ways of presenting this subject. The first is by the kind of problem — the symmetric eigenproblem, the generalized eigenproblem, etc. This approach has obvious advantages for the person who is interested in only one kind of problem. It has the disadvantage that, at the very least, a thumbnail sketch of each algorithm must accompany each problem type. The alternative is to partition the subject by algorithms, which is what we will do here.

In the present chapter we will be concerned with methods based on Krylov sequences — specifically, Arnoldi's method and the Lanczos method. In the first subsection we will introduce certain decompositions associated with Krylov subspaces. The decompositions form the basis for the the Arnoldi and Lanczos algorithms to be treated in §2 and §3.

### 1. KRYLOV DECOMPOSITIONS

For a fixed matrix  $A$ , not every subspace of dimension  $k$  is a Krylov subspace of  $A$ . This means that any basis of a Krylov subspace must satisfy certain constraining relations, which we will call *Krylov decompositions*. It turns out that a knowledge of these decompositions enables us to compute and manipulate Krylov subspaces. Consequently, this section is devoted to describing the relations that must hold for a basis to span a Krylov subspace.

The most widely used Krylov decompositions are those that result from orthogonalizing a Krylov sequence. They are called *Arnoldi decompositions* for non-Hermitian matrices and *Lanczos decompositions* for Hermitian matrices. We will treat these special decompositions in the first two subsections. Unfortunately, these decompositions are not closed under certain natural operations, and in §1.3 we introduce more general decompositions that are closed — decompositions that will be used in our computational algorithms.

### 1.1. ARNOLDI DECOMPOSITIONS

An explicit Krylov basis of the form

$$K_k(A, u) = (u \ A u \ \cdots \ A^{k-1} u)$$

is not suitable for numerical work. The reason is that as  $k$  increases, the columns of  $K_k$  become increasingly dependent (since they tend increasingly to lie in the space spanned by the dominant eigenvectors). For example, the following table gives the condition number (the ratio of the largest to the smallest singular values) for the Krylov basis  $K_k(A, e)$ , where  $A$  is the matrix of Example 3.1, Chapter 4.

| $k$ | Condition |
|-----|-----------|
| 5   | 5.8e+02   |
| 10  | 3.4e+06   |
| 15  | 2.7e+10   |
| 20  | 3.1e+14   |

A large condition number means that the space spanned by the basis is extremely sensitive to small perturbations in the vectors that form the basis.

It is therefore natural to replace a Krylov basis with a better-conditioned set of vectors, say an orthonormal set. Unfortunately, a direct orthogonalization of the columns of  $K_k$  will not solve the problem; once the columns have become nearly dependent, information about the entire Krylov subspace has been lost, and orthogonalization will not restore it. We therefore proceed indirectly through what we will call an Arnoldi decomposition.

#### The Arnoldi decomposition

Let us suppose that the columns of  $K_{k+1}$  are linearly independent and let

$$K_{k+1} = U_{k+1} R_{k+1} \quad (1.1)$$

be the QR factorization of  $K_{k+1}$  (see §6, Chapter 2.3). Then the columns of  $U_{k+1}$  are the results of successively orthogonalizing the columns of  $K_{k+1}$ . It turns out that we can eliminate  $K_{k+1}$  from equation (1.1). The resulting *Arnoldi decomposition* characterizes the orthonormal bases that are obtained from a Krylov sequence.

**Theorem 1.1.** *Let  $u_1$  have 2-norm one and let the columns of  $K_{k+1}(A, u_1)$  be linearly independent. Let  $U_{k+1} = (u_1 \ \cdots \ u_{k+1})$  be the Q-factor of  $K_{k+1}$ . Then there is a  $(k+1) \times k$  unreduced upper Hessenberg matrix  $\hat{H}_k$  such that*

$$AU_k = U_{k+1} \hat{H}_k. \quad (1.2)$$

*Conversely, if  $U_{k+1}$  is orthonormal and satisfies (1.2), where  $\hat{H}_k$  is a  $(k+1) \times k$  unreduced Hessenberg matrix, then  $U_{k+1}$  is the Q-factor of  $K_{k+1}(A, u_1)$ .*

**Proof.** If we partition (1.1) in the form

$$(K_k \ A^k u) = (U_k \ u_{k+1}) \begin{pmatrix} R_k & r_{k+1} \\ 0 & \rho_{k+1,k+1} \end{pmatrix},$$

we see that  $K_k = U_k R_k$  is the QR factorization of  $K_k$ . Now let  $S_k = R_k^{-1}$ . Then

$$\begin{aligned} AU_k &= AK_k S_k \\ &= K_{k+1} \begin{pmatrix} 0 \\ S_k \end{pmatrix} \\ &= U_{k+1} R_{k+1} \begin{pmatrix} 0 \\ S_k \end{pmatrix} \\ &= U_{k+1} \hat{H}_k, \end{aligned}$$

where

$$\hat{H}_k = R_{k+1} \begin{pmatrix} 0 \\ S_k \end{pmatrix}.$$

It is easily verified that  $\hat{H}_k$  is a  $(k+1) \times k$  Hessenberg matrix whose subdiagonal elements are  $h_{i+1,i} = r_{i+1,i+1}s_{ii} = r_{i+1,i+1}/r_{ii}$ . Thus by the nonsingularity of  $R_k$ ,  $\hat{H}_k$  is unreduced.

Conversely, suppose that the orthonormal matrix  $U_{k+1}$  satisfies the relation (1.2), where  $H_k$  is a  $(k+1) \times k$  unreduced Hessenberg matrix. Now if  $k = 1$ , then  $Au_1 = h_{11}u_1 + h_{21}u_2$ . Since  $h_{21} \neq 0$ , the vector  $u_2$  is a linear combination of  $u_1$  and  $Au_1$  that is orthogonal to  $u_1$ . Hence  $(u_1 \ u_2)$  is the Q-factor of  $K_2$ .

Assume inductively that  $U_k$  is the Q-factor of  $K_k$ . If we partition

$$\hat{H}_k = \begin{pmatrix} \hat{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{pmatrix},$$

then from (1.3)

$$Au_k = U_k h_k + h_{k+1,k} u_{k+1}.$$

Thus  $u_{k+1}$  is a linear combination of  $Au_k$  and the columns of  $U_k$  and lies in  $\mathcal{K}_{k+1}$ . Hence  $U_{k+1}$  is the Q-factor of  $K_{k+1}$ . ■

This theorem justifies the following definition.

**Definition 1.2.** Let  $U_{k+1} \in \mathbb{C}^{n \times (k+1)}$  be orthonormal and let  $U_k$  consist of the first  $k$  columns of  $U_{k+1}$ . If there is a  $(k+1) \times k$  unreduced upper Hessenberg matrix  $\hat{H}_k$  such that

$$AU_k = U_{k+1} \hat{H}_k, \tag{1.3}$$

then we call (1.3) an ARNOLDI DECOMPOSITION OF ORDER  $k$ . If  $H$  is reduced, we say the Arnoldi decomposition is REDUCED.

In this nomenclature, Theorem 1.1 says that every nondegenerate Krylov sequence is associated with an Arnoldi decomposition and vice versa.

An Arnoldi decomposition can be written in a different form that will be useful in the sequel. Specifically, partition

$$\hat{H}_k = \begin{pmatrix} H_k \\ h_{k+1,k} \mathbf{e}_k^T \end{pmatrix},$$

and set

$$\beta_k = h_{k+1,k}.$$

Then the Arnoldi decomposition (1.2) is equivalent to the relation

$$AU_k = U_k H_k + \beta_k u_{k+1} \mathbf{e}_k^T. \quad (1.4)$$

In what follows it will reduce clutter if we dispense with iteration subscripts and write (1.4) in the form

$$AU = UH + \beta u.$$

When we do, the compact form (1.3) will be written in the form

$$AU = \hat{U} \hat{H}.$$

Thus the hats over the matrices  $U$  and  $H$  represent an augmentation by a column and a row respectively.

### Uniqueness

The language we have used above—*an* Arnoldi decomposition—suggests the possibility of having more than one Arnoldi decomposition associated with a Krylov subspace. In fact, this can happen only when the sequence terminates, as the following theorem shows.

**Theorem 1.3.** *Suppose the Krylov sequence  $K_{k+1}(A, u_1)$  does not terminate at  $k+1$ . Then up to scaling of the columns of  $U_{k+1}$ , the Arnoldi decomposition of  $K_{k+1}$  is unique.*

**Proof.** Let

$$AU_k = U_k H_k + \beta_k u_{k+1} \mathbf{e}_k^T \quad (1.5)$$

be the Arnoldi decomposition from Theorem 1.1. Then by this theorem  $H_k$  is unreduced and  $\beta_k \neq 0$ . Let

$$A\tilde{U}_k = \tilde{U}_k \tilde{H}_k + \tilde{\beta}_k \tilde{u}_{k+1} \mathbf{e}_k^T,$$

where  $\tilde{U}_{k+1}$  is an orthonormal basis for  $\mathcal{K}_{k+1}(A, u_1)$ . We claim that  $\tilde{U}_k^H u_{k+1} = 0$ . For otherwise there is a  $\tilde{u}_j$  such that  $\tilde{u}_j = \alpha u_{k+1} + U_k a$ , where  $\alpha \neq 0$ . Hence  $A\tilde{u}_j$  contains a component along  $A^{k+1}u$  and does not lie in  $\mathcal{K}_{k+1}$ , a contradiction.

It follows that  $\mathcal{R}(U_k) = \mathcal{R}(\tilde{U}_k)$  and hence that  $\tilde{U}_k = U_k Q$ , for some unitary matrix  $Q$ . Hence

$$A(\tilde{U}_k Q) = (\tilde{U}_k Q)(Q^H \tilde{H}_k Q) + \tilde{\beta}_k \tilde{u}_{k+1} (\mathbf{e}_k^T Q), \quad (1.6)$$

or

$$AU_k = U_k(Q^H \tilde{H}_k Q) + \tilde{\beta}_k \tilde{u}_{k+1} \mathbf{e}_k^T Q. \quad (1.7)$$

On premultiplying (1.5) and (1.7) by  $U_k$  we obtain

$$H_k = U_k^H AU_k = Q^H \tilde{H}_k Q.$$

Similarly, premultiplying by  $u_{k+1}^H$ , we obtain

$$\beta_k \mathbf{e}_k^T = u_{k+1}^H AU_k = \tilde{\beta}_k \mathbf{e}_k^T Q. \quad (1.8)$$

Now by the independence of the columns of  $K_{k+1}$ , both  $\beta_k$  and  $\tilde{\beta}_k$  are nonzero. It then follows from (1.8) that the last row of  $Q$  is  $\omega_k \mathbf{e}_k^T$ , where  $|\omega_k| = 1$ . Since the norm of the last column of  $Q$  is one, the last column of  $Q$  is  $\omega_k \mathbf{e}_k$ . Since  $H_k$  is unreduced, it follows from the implicit Q theorem (Theorem 3.3, Chapter 2) that  $Q = \text{diag}(\omega_1, \dots, \omega_k)$ , where  $|\omega_j| = 1$  ( $j = 1, \dots, k$ ). Thus up to column scaling  $U_k = \tilde{U}_k Q$  is the same as  $\tilde{U}_k$ . Subtracting (1.7) from (1.5) we find that

$$\beta_k u_{k+1} = \omega_k \tilde{\beta}_k u_{k+1}$$

so that up to scaling  $u_{k+1}$  and  $\tilde{u}_{k+1}$  are the same. ■

This theorem allows us to speak of *the Arnoldi decomposition* associated with a nonterminating Krylov subspace. Perhaps more important, it implies that there is a one-one correspondence between Krylov subspaces and their starting vectors.

*A nonterminating Krylov subspace uniquely determines its starting vector.*

The nontermination hypothesis is necessary. If, for example,  $\mathcal{K}_{k+1} = \mathcal{K}_{k+2}$ , then  $AU_{k+1} = U_{k+1} H_{k+1}$ , and for any  $Q$  such that  $Q^H H_{k+1} Q$  is upper Hessenberg, the decomposition  $A(U_{k+1} Q) = (U_{k+1} Q)(Q^H H_{k+1} Q)$  is another Arnoldi decomposition.

### The Rayleigh quotient

In the proof of Theorem 1.3 we used the fact that  $H_k = U_k^H A U_k$ . But this matrix is just the Rayleigh quotient with respect to the basis  $U_k$  (see Definition 2.7, Chapter 4). Thus the Arnoldi decomposition provides all that is needed to compute Ritz pairs.

### Reduced Arnoldi decompositions

In Definition 1.2 we have required that the Hessenberg part of an Arnoldi decomposition  $AU_k = U_{k+1}\hat{H}_k$  be unreduced. It is natural to ask under what conditions an Arnoldi-like relation can be satisfied with a reduced Hessenberg matrix. The following theorem answers that question.

**Theorem 1.4.** *Let the orthonormal matrix  $U_{k+1}$  satisfy*

$$AU_k = U_{k+1}\hat{H}_k,$$

where  $\hat{H}_k$  is Hessenberg. Then  $\hat{H}_k$  is reduced if and only if  $\mathcal{R}(U_k)$  contains an eigenspace of  $A$ .

**Proof.** First suppose that  $\hat{H}_k$  is reduced, say that  $h_{j+1,j} = 0$ . Let  $H$  be the leading principal submatrix of  $\hat{H}_k$  of order  $k$  and let  $U$  be the matrix consisting of the first  $j$  columns of  $U_{k+1}$ . Then it is easily verified that  $AU = UH$ , so that  $U$  is an eigenbasis of  $A$ .

Conversely, suppose that  $A$  has an eigenspace that is a subset of  $\mathcal{R}(U_k)$  and suppose that  $H_k$  is unreduced. Then  $\mathcal{R}(U_k)$  contains an eigenvector of  $A$ , which we can write in the form  $x = U_k w$  for some  $w$ . Let  $\lambda$  be the corresponding eigenvalue and let

$$\hat{H}_\lambda = \begin{pmatrix} H_k - \lambda I \\ h_{k+1,k}e_k^T \end{pmatrix}.$$

Then

$$0 = (A - \lambda I)U_k w = U_{k+1}\hat{H}_\lambda w.$$

Because  $\hat{H}_\lambda$  is unreduced, the matrix  $U_{k+1}\hat{H}_\lambda$  is of full column rank. It follows that  $w = 0$ , a contradiction. ■

This theorem, along with Theorem 3.5, Chapter 4, shows that the only way for an Arnoldi decomposition to be reduced is for the Krylov sequence to terminate prematurely. However, if  $h_{j+1,j} = 0$ , the sequence can be continued by setting  $u_{j+1}$  equal to any normalized vector in the orthogonal complement of  $U_j$ . If this procedure is repeated each time the sequence terminates, the result is a reduced Arnoldi decomposition.

### Computing Arnoldi decompositions

The Arnoldi decomposition suggests a method for computing the vector  $u_{k+1}$  from  $u_1, \dots, u_k$ . Write the  $k$ th column of (1.4) in the form

$$Au_k = U_k h_k + \beta_k u_{k+1}.$$

Then from the orthonormality of  $U_{k+1}$  we have

$$h_k = U_k^H Au_k.$$

Since  $\beta_k u_k = Au_k - U_k h_k$  and  $\|u_k\|_2 = 1$ , we must have

$$\beta_k = \|Au_k - U_k h_k\|_2$$

and

$$u_{k+1} = \beta_k^{-1}(Au_k - U_k h_k).$$

These considerations lead to the following algorithm for computing an expanding set of Arnoldi decompositions.

1. **for**  $k = 1, 2, \dots$
  2.      $h_k = U_k^H A u_k$
  3.      $v = Au_k - U_k h_k$
  4.      $\beta_k = h_{k+1,k} = \|v\|_2$
  5.      $u_{k+1} = v/\beta_k$
  6.      $\hat{H}_k = \begin{pmatrix} \hat{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{pmatrix}$
  7. **end for**  $k$
- (1.9)

We call this procedure the *Arnoldi process*. It is worthwhile to verify that this process is the same as the reduction to Hessenberg form described in §3.2, Chapter 2.

### Reorthogonalization

Although we derived (1.9) from the Arnoldi decomposition, the computation of  $u_{k+1}$  is actually a form of the well-known Gram–Schmidt algorithm. This algorithm has been treated extensively in §I:4.1.4, where it is shown that in the presence of inexact arithmetic cancellation in statement 3 can cause it to fail to produce orthogonal vectors. The cure is a process called reorthogonalization. It is illustrated in the following extension of (1.9).

1. **for**  $k = 1, 2, \dots$
2.      $h_k = U_k^H A u_k$
3.      $v = Au_k - U_k h_k$
4.      $w = U_k^H v$
5.      $h_k = h_k + w$
6.      $v = v - U_k w$
7.      $\beta_k = h_{k+1,k} = \|v\|_2$
8.      $u_{k+1} = v_k/\beta_k$
9.      $\hat{H}_k = \begin{pmatrix} \hat{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{pmatrix}$
10. **end for**  $k$

Since the vector  $v$  in statement 3 need not be orthogonal to the columns of  $U_k$ , the extended algorithm therefore orthogonalizes it again, accumulating any changes in  $h_k$ .

Given an Arnoldi decomposition

$$AU = \hat{U}\hat{H} = UH + \beta ue_k^T$$

of order  $k$ , *ExpandArn* extends it to an Arnoldi decomposition of order  $k + 1$ . The algorithm uses an auxiliary routine *gsreorthog* to perform the orthogonalization (Algorithm I:4.1.13).

1. *ExpandArn*( $\hat{U}$ ,  $\hat{H}$ )
2.  $v = A * \hat{U}[:, k+1]$
3. *gsreorthog*( $\hat{U}$ ,  $v$ ,  $u$ ,  $h$ ,  $\beta$ )
4.  $\hat{H} = \begin{pmatrix} \hat{H} & h \\ 0 & \beta \end{pmatrix}$
5.  $\hat{U} = (\hat{U} \ u)$
6. **end** *ExpandArn*

### Algorithm 1.1: The Arnoldi step

---

In general, this algorithm is quite satisfactory—usually no more than the one reorthogonalization performed in statements 4–6 is needed. However, there are some tricky special cases; e.g., what to do when  $v$  in statement 3 is zero. More important, we can often dispense with the reorthogonalization entirely. Hence, our final algorithm—Algorithm 1.1—uses a function *gsreorthog* (Algorithm I:4.1.3) that handles exceptional cases and tests whether to reorthogonalize. By applying Algorithm 1.1 repeatedly, one can compute an Arnoldi decomposition of any order.

### Practical considerations

The work in computing an Arnoldi step is divided between the computation of the vector-matrix product  $Au$  and the orthogonalization. The user must provide a method for computing the former, and its cost will vary. For more see Example 1.2, Chapter 2.

The cost of the orthogonalization will be  $2nkr$  flam, where  $r$  is the number of orthogonalization steps performed by *gsreorthog*. In general  $r$  will be one or two. Hence:

*The cost of an Arnoldi step ranges between  $2nk$  flam and  $4nk$  flam plus a matrix-vector multiplication.* (1.10)

To compute an Arnoldi decomposition of order  $k$  from scratch, one must apply the algorithm for orders  $i = 1, \dots, k-1$ . Thus:

*It requires between  $nk^2$  flam and  $2nk^2$  flam plus  $k$  matrix-vector multiplications to compute an Arnoldi decomposition of order  $k$ .* (1.11)

It requires  $nk$  floating-point words to store the matrix  $U$ . This fact, more than operations counts, restricts our ability to compute Arnoldi decompositions. If  $n$  is very large, then as  $k$  grows we will quickly run out of memory. Of course we can move the columns of  $U$  to a backing store—say a disk. But each Arnoldi step requires all the previous vectors, and transfers between backing store and main memory will overwhelm the calculation.

### Orthogonalization and shift-and-invert enhancement

We have seen in connection with the inverse power method (§1.2, Chapter 2) that replacing  $A$  by  $(A - \kappa I)^{-1}$  will make eigenvalues near  $\kappa$  large. If  $(A - \kappa I)^{-1}$  is used to form a Krylov sequence, the sequence will tend to produce good approximations to those large eigenvalues. Thus shift-and-invert enhancement can be used to steer a Krylov sequence to a region whose eigenvalues we are interested in.

For the inverse power method, where the concern is with one eigenpair, the closer the shift is to the eigenvalue the better the performance of the method. With Krylov sequences, where the concern is often with several eigenpairs, an excessively accurate approximation to one eigenvalue can cause the convergence to the other eigenpairs to stagnate. The problem, at least in part, is in the Arnoldi procedure, as the following example shows.

**Example 1.5.** Let

$$A_i = \text{diag}(10^i, 1, \dots, 10^{-5})$$

be of order 30 with the last 29 eigenvalues decaying geometrically. For  $i > 1$ , the Krylov subspaces  $\mathcal{K}_k(A_i, \mathbf{e})$  can be expected to quickly develop very good approximations to  $\mathbf{e}_1$ . The bound (3.12), Chapter 4 suggests that it will also produce approximations to  $\mathbf{e}_2$ , although more slowly. In fact, Figure 1.1 plots  $\sin \theta$ , where  $\theta = \angle[\mathbf{e}_2, \mathcal{K}_k(A_i, \mathbf{e})]$  versus  $k$  for several values of  $i$ . The convergence of  $\sin \theta$  toward zero proceeds typically up to a point and then levels off. The larger the value of  $i$ , the sooner the convergence stagnates.

In this example we have simulated a shift-and-invert transformation by giving  $A_i$  a large eigenvalue of  $10^i$ . Moreover, we have taken  $A_i$  to be diagonal, so that the effects of rounding error in the formation of the matrix-vector product  $A_i u_k$  are minimal. Thus the effects observed in the figure are due to the combination of the large eigenvalue and the orthogonalization step. No formal analysis of this phenomenon has appeared in the literature, but what seems to be happening is the following. In any Krylov sequence the Arnoldi vector  $u_{k+1}$  is orthogonal to  $\mathcal{K}_k$ . Hence if there is an accurate approximation to any vector in  $\mathcal{K}_k$ , then  $u_{k+1}$  must be almost orthogonal to it. For our example, this means that the first component of  $u_k$  will rapidly approach zero as good approximations to  $\mathbf{e}_1$  appear in  $\mathcal{K}_k$ . Similarly, as good approximations to  $\mathbf{e}_2$  appear, the second component of  $u_k$  will approach zero. Now in the presence of rounding error, the first component of  $u_k$  will not become exactly zero but will be reduced to

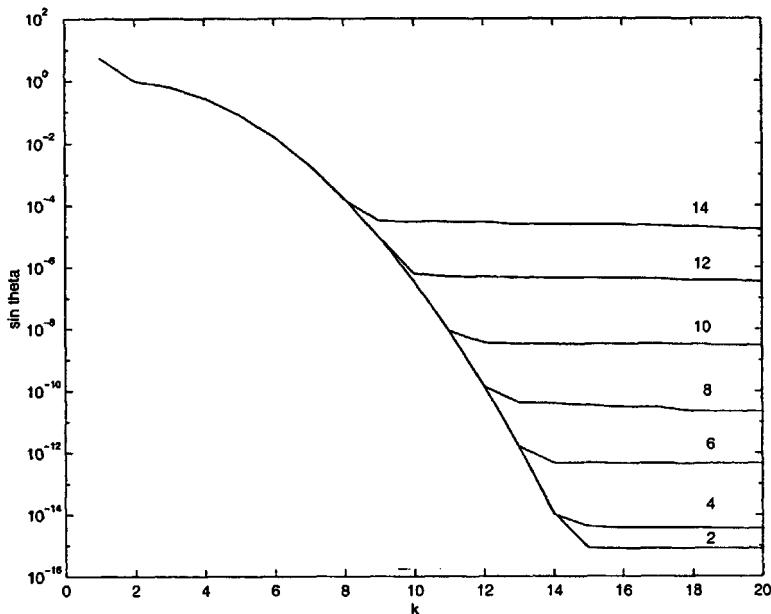


Figure 1.1: Stagnation in a Krylov sequence

the order of the rounding unit — say  $10^{-16}$ . But this nonzero first component is magnified by  $10^i$  when  $Au_k$  is formed. When  $Au_k$  is reorthogonalized, this spurious first component will introduce errors of order  $10^{i-16}$  in the other components. Thus we have effectively changed our rounding unit from  $10^{-16}$  to  $10^{i-16}$ , and we can expect no greater accuracy in the approximations to the other eigenvectors.

The moral of this example is that if you wish to use shift-and-invert enhancement to find eigenpairs in a region, do not let the shift fall too near any particular eigenvalue in the region.

## 1.2. LANCZOS DECOMPOSITIONS

### Lanczos decompositions

When  $A$  is Hermitian the Arnoldi decomposition and the associated orthogonalization process simplify. Specifically, let  $A$  be Hermitian and let

$$AU_k = U_k T_k + \beta_k u_{k+1} e_k^T \quad (1.12)$$

be an Arnoldi decomposition. Then the Rayleigh quotient  $T_k$  is upper triangular. But since  $T_k = U_k^H A U_k$ ,  $T_k$  is Hermitian. It follows that  $T_k$  is tridiagonal and can be

written in the form

$$T_k = \begin{pmatrix} \alpha_1 & \bar{\beta}_1 & & & \\ \beta_1 & \alpha_2 & \bar{\beta}_2 & & \\ & \beta_2 & \alpha_3 & \bar{\beta}_3 & \\ & & \ddots & \ddots & \\ & & & \beta_{k-2} & \alpha_{k-1} & \bar{\beta}_{k-1} \\ & & & & \beta_{k-1} & \alpha_k \end{pmatrix}. \quad (1.13)$$

We will call the decomposition (1.12) a *Lanczos decomposition*.

Like the Arnoldi, the Lanczos decomposition is essentially unique, provided that the matrix  $T_k$  is unreduced. This will always be true if the sequence does not terminate.

### The Lanczos recurrence

The fact that  $T_k$  is tridiagonal implies that the orthogonalization  $u_{k+1}$  depends only on  $u_k$  and  $u_{k-1}$ , in contrast to the Arnoldi procedure where  $u_k$  depends on  $u_1, \dots, u_k$ . To see this, note that the first column of the Lanczos decomposition (1.12) is

$$Au_1 = \alpha_1 u_1 + \beta_1 u_2,$$

or

$$u_2 = \frac{Au_1 - \alpha_1 u_1}{\beta_1}.$$

Moreover, from the orthonormality of  $u_1$  and  $u_2$  it follows that  $\alpha_1 = u_1^H Au_1$  and we may take  $\beta_1 = \|Au_1 - \alpha_1 u_1\|_2$ .

More generally, from the  $j$ th column of (1.12) we get the relation

$$u_{j+1} = \frac{Au_j - \alpha_j u_j - \bar{\beta}_{j-1} u_{j-1}}{\beta_j},$$

where

$$\alpha_j = u_j^H Au_j \quad \text{and} \quad \beta_j = \|Au_j - \alpha_j u_j - \bar{\beta}_{j-1} u_{j-1}\|_2. \quad (1.14)$$

This is the Lanczos three-term recurrence, which is implemented in Algorithm 1.2. Here are three comments.

- Since  $\beta_{j-1}$  is taken to be real, we do not have to conjugate it in statement 5.
- A frequently occurring alternative is to calculate

1.  $v = A*u_j - \beta_{j-1}*u_{j-1}$
2.  $\alpha_j = u_j^H * v$
3.  $v = v - \alpha_j*u_j$

This is the equivalent of the modified Gram–Schmidt algorithm (see §I:4.1.4). Both work well. For more on variants of the basic algorithm, see the notes and references.

- The body of the loop requires one multiplication by  $A$ , two inner products, and two subtractions of a constant times a vector. Hence:

Let  $u_1$  be given. This algorithm generates the Lanczos decomposition

$$AU_k = U_k T_k + \beta_k u_{k+1} e_k^T$$

where  $T_k$  has the form (1.13).

1.  $u_0 = 0; \beta_0 = 0;$
2. **for**  $j = 1$  **to**  $k$
3.      $v = A * u_j$
4.      $\alpha_j = u_j^H * u_{j+1}$
5.      $v = v - \alpha_j * u_j - \beta_{j-1} * u_{j-1}$
6.      $\beta_j = \|v\|_2$
7.      $u_{j+1} = v / \beta_j$
8. **end for**  $j$

### Algorithm 1.2: The Lanczos recurrence

---

*The operation count for Algorithm 1.2 is  $k$  matrix-vector multiplications and  $4nk$  flam.*

The above is less by a factor of about  $k$  than the corresponding count (1.11) for the Arnoldi process.

---

Although the simplicity and economy of the Lanczos recurrence is enticing, there are formidable obstacles to turning it into a working method. If only eigenvalues are desired, we can discard the  $u_j$  as they are generated and compute Ritz values from the Rayleigh quotient  $T_k$ . But if one wants to compute eigenvectors, one must retain all the vectors  $u_j$ , and for large problems it will be necessary to transfer them to a backing store.

A more serious problem is loss of orthogonality among the vectors  $u_j$ . In the Arnoldi process we were able to overcome this problem by reorthogonalization. However, if we reorthogonalize  $u_{j+1}$  in the Lanczos against its predecessors, we lose all the advantages of the Lanczos recurrence. We will return to this problem in §3, where we will treat the Lanczos algorithm in detail.

## 1.3. KRYLOV DECOMPOSITIONS

An Arnoldi (or Lanczos) decomposition can be regarded as a vehicle for combining Krylov sequences with Rayleigh–Ritz refinement. It exhibits an orthonormal basis for a Krylov subspace and its Rayleigh quotient—the wherewithal for computing Ritz pairs. As we shall see, it also permits the efficient computation of the residual norms of the Ritz pairs. However, because of its uniqueness, we can do little more with it. For

example, there is no unreduced Arnoldi form whose Rayleigh quotient is triangular. In this subsection we will show how to relax the definition of an Arnoldi decomposition to obtain a decomposition that is closed under similarity transformations of its Rayleigh quotient.

### Definition

A Krylov decomposition, like an Arnoldi decomposition, specifies a relation among a set of linearly independent vectors that guarantees that they span a Krylov subspace. Specifically, we make the following definition.

**Definition 1.6.** Let  $u_1, u_2, \dots, u_{k+1}$  be linearly independent and let

$$U_k = (u_1 \ \cdots \ u_k).$$

A KRYLOV DECOMPOSITION OF ORDER  $k$  is a relation of the form

$$AU_k = U_k B_k + u_{k+1} b_{k+1}^H. \quad (1.16)$$

Equivalently, we can write

$$AU_k = U_{k+1} \hat{B}_k,$$

where  $U_{k+1} = (U_k \ u_{k+1})$  and

$$\hat{B}_k = \begin{pmatrix} B_k \\ b_{k+1}^H \end{pmatrix}.$$

If  $U_{k+1}$  is orthonormal, we say that the Krylov decomposition is ORTHONORMAL.

We call the  $\mathcal{R}(U_{k+1})$  the SPACE SPANNED BY THE DECOMPOSITION and  $U_{k+1}$  the basis for the decomposition. Two Krylov decompositions spanning the same space are said to be EQUIVALENT.

Thus a Krylov decomposition has the general form as an Arnoldi decomposition, but the matrix  $\hat{B}_k$  is not assumed to be of any particular form. It is uniquely determined by the basis  $U_{k+1}$ . For if  $(V \ v)^H$  is any left inverse for  $U_{k+1}$ , then it follows from (1.16) that

$$B_k = V^H A U_k \quad \text{and} \quad b_{k+1}^H = v^H A U_k. \quad (1.17)$$

In particular,  $B_k$  is a Rayleigh quotient of  $A$ .

Here is a nontrivial example of a Krylov decomposition.

**Example 1.7.** Let

$$B = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

be of order  $k$ . If the columns of  $K_{k+1}(A, u_1)$  are independent, then

$$AK_k = K_k B_k + A^k u_1 \mathbf{e}_k^T$$

is a Krylov decomposition whose basis is the Krylov basis.

Every Arnoldi decomposition is a Krylov decomposition, but not every Krylov decomposition is an Arnoldi decomposition, as the above example shows. However, every Krylov decomposition is equivalent to a (possibly unreduced) Arnoldi decomposition. If the Arnoldi decomposition is unreduced, then it is essentially unique, and the subspace of the original Krylov decomposition must be a nondegenerate Krylov subspace. To establish these facts we will introduce two classes of transformations that transform a Krylov decomposition into an equivalent Krylov decomposition. They are called similarity transformations and translations.

### Similarity transformations

Let

$$AU_k = U_k B_k + u_{k+1} b_{k+1}^H$$

be a Krylov decomposition and let  $Q$  be nonsingular. Then on postmultiplying by  $Q$  we obtain the equivalent Krylov decomposition

$$A(U_k Q) = (U_k Q)(Q^{-1} B Q) + u_{k+1}(b_{k+1}^H Q). \quad (1.18)$$

We will say that the two Krylov decompositions are *similar*.

We have already met with a similarity transformation in the proof of the uniqueness of the Arnoldi decomposition [see (1.6)]. In that case, however, the transforming matrix  $Q$  necessarily satisfied  $|Q| = I$  so that the two similar decompositions were essentially the same. With a general Krylov decomposition, however, we are free to reduce the Rayleigh quotient  $B$  to any convenient form. For example, we could reduce  $B$  to Jordan form, in which case we will say that the decomposition is a Krylov–Jordan decomposition. In this terminology an Arnoldi decomposition is an orthonormal Krylov–Hessenberg decomposition and a Lanczos decomposition is an orthonormal Krylov–tridiagonal decomposition. There is also a Krylov–Schur decomposition, which will play an important role in our treatment of the Arnoldi method (see §2.2).

### Translation

The similarity transformation (1.18) leaves the vector  $u_{k+1}$  unaltered. We can transform this vector as follows. In the Krylov decomposition

$$AU_k = U_k B_k + u_{k+1} b_{k+1}^H$$

let

$$\gamma \tilde{u}_{k+1} = u_{k+1} - U_k a,$$

where by the independence of the vectors  $u_j$  we must have  $\gamma \neq 0$ . Then it follows that

$$AU_k = U_k(B_k + ab_{k+1}^H) + \tilde{u}_{k+1}(\gamma b_k^H).$$

Since  $\mathcal{R}[(U_k \ u_{k+1})] = \mathcal{R}[(U_k \ \tilde{u}_{k+1})]$ , this Krylov decomposition is equivalent to the original. We say that it was obtained from the original decomposition by *translation*.

Note that translation cannot produce an arbitrary vector  $\tilde{u}_{k+1}$ . Instead,  $\tilde{u}_{k+1}$  is constrained to have components along the original vector  $u_{k+1}$ , so that the two decompositions span the same space.

### Equivalence to an Arnoldi decomposition

We will now use the transformations introduced above to prove the following theorem.

**Theorem 1.8.** *Every Krylov decomposition is equivalent to a (possibly reduced) Arnoldi decomposition.*

**Proof.** We begin with the Krylov decomposition

$$AU = UB + ub^H,$$

where for convenience we have dropped the subscripts in  $k$  and  $k+1$ . Let  $U = \dot{U}R$  be the QR factorization of  $U$ . Then

$$A\dot{U} \equiv A(UR^{-1}) = (UR^{-1})(RBR^{-1}) + u(b^HR^{-1}) \equiv \dot{U}\dot{B} + ub^H$$

is an equivalent decomposition, in which the matrix  $\dot{U}$  is orthonormal. Next let

$$\dot{u} = \gamma^{-1}(u - Ua)$$

be a vector of norm one such that  $U^H\dot{u} = 0$ . Then the translated decomposition

$$A\dot{U} = \dot{U}(B + ab^H) + \ddot{u}(\gamma b^H) \equiv \dot{U}\ddot{B} + \ddot{u}b^H$$

is an equivalent orthonormal Krylov decomposition. Finally, let  $Q$  be a unitary matrix such that  $b^HQ = \|b\|_2 e_k^T$  and  $Q^H \ddot{B} Q$  is upper Hessenberg. Then the equivalent decomposition

$$A\tilde{U} \equiv A(\dot{U}Q) = (\dot{U}Q)(Q^H \ddot{B} Q) + \dot{u}(\ddot{b}^HQ) \equiv \tilde{U}\ddot{B} + \tilde{u}\tilde{b}^H e_k^T$$

is a possibly reduced Arnoldi decomposition.

There are two points to be made about this theorem.

- If the resulting Arnoldi decomposition is unreduced, then by Theorem 1.3 it is essentially unique, and by Theorem 1.4  $\mathcal{R}(U)$  contains no invariant subspaces of  $A$ .
- The proof of Theorem 1.8 is constructive. Given the original decomposition, each step in the proof has an algorithmic realization. If the original basis  $U_{k+1}$  is reasonably well conditioned, these algorithms will be reasonably stable.

---

To summarize:

$$\begin{array}{c}
 \left( \begin{array}{cccccc} b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \end{array} \right) \xrightarrow{H_2} \left( \begin{array}{cccccc} b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ 0 & 0 & 0 & b & b \end{array} \right) \xrightarrow{H_3} \left( \begin{array}{cccccc} b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \end{array} \right) \\
 \xrightarrow{H_4} \left( \begin{array}{cccccc} b & b & b & b & b \\ b & b & b & b & b \\ 0 & b & b & b & b \\ 0 & 0 & b & b & b \\ 0 & 0 & 0 & b & b \end{array} \right)
 \end{array}$$

Figure 1.2: Rowwise reduction to Hessenberg form

Each Krylov sequence that does not terminate at  $k$  corresponds to the class of Krylov decompositions that are equivalent to its Arnoldi decomposition.

### Reduction to Arnoldi form

As we shall see, it is sometimes necessary to pass from an orthonormal Krylov sequence to an Arnoldi sequence. This can be done by Householder transformations as follows.

Let the Krylov decomposition in question be

$$AU = UB + u^H b,$$

where  $B$  is of order  $k$ . Let  $H_1$  be a Householder transformation such that  $b^H H_1 = \beta e_k$ . We now reduce  $H_1 B H_1$  to Hessenberg form by a variant of the reduction described in §2.3, Chapter 2, in which the matrix is reduced rowwise beginning with the last row. The course of the reduction is illustrated in Figure 1.2.

Now let  $Q = H_1 H_2 \cdots H_{k-1}$ . Then the Krylov decomposition

$$A(UQ) = (UQ)(Q^H H Q) + b^H Q \quad (1.19)$$

has a Hessenberg Rayleigh quotient. Because the last rows of  $H_2, \dots, H_{k-1}$  are  $e_1$ ,

$$b^H Q = (b^H H_1)(H_2 \cdots H_{k-1}) = \beta e_k^T (H_2 \cdots H_{k-1}) = \beta e_k^T$$

so that (1.19) is an Arnoldi decomposition.

Algorithm 1.3 implements this scheme. It uses a variant of *housegen* that zeros the initial components of a vector.

This algorithm illustrates an important fact about algorithms that transform Krylov decompositions. If  $n$  is large, transformations on the Rayleigh quotient are negligible

Let

$$AU = UB + ub^H$$

be an orthogonal Krylov decomposition of order  $k$ . This algorithm reduces it to an Arnoldi decomposition.

$$AU = UH + \beta ue_k^T$$

The algorithm uses the routine *housegen2*( $x, u, \nu$ ), which takes an  $n$ -vector  $x$  and returns a Householder transformation  $H = I - uu^H$  such that  $Hx = \nu e_n$  [see (2.20), Chapter 2].

1.  $H = B$
2.  $Q = I$
3. **for**  $j = k$  **to** 2 **By**  $-1$
4.     **if** ( $j = k$ )
5.         *housegen2*( $b^H, r^H, \beta$ )
6.     **else**
7.         *housegen2*( $H[j+1, 1:j], r^H, H[j+1, j]$ )
8.     **end if**
9.      $s = H[1:j, 1:j]*r$
10.     $H[1:j, 1:j] = H[1:j, 1:j] - s*r^H$
11.    **if** ( $j \neq k$ )  $H[j+1, 1:j-1] = 0$  **fi**
12.     $s^H = r^H * H[1:j, 1:k]$
13.     $H[1:j, 1:k] = H[1:j, 1:k] - r*s^H$
14.     $s = Q[:, 1:j]*r$
15.     $Q[:, 1:j] = Q[:, 1:j] - s*r^H$
16. **end for**  $j$
17.  $U = U*Q$

**Algorithm 1.3:** Krylov decomposition to Arnoldi decomposition



compared to the manipulation of  $U$ . Thus the bulk of the work in Algorithm 1.3 is found in the very last statement, where  $U$  is overwritten by  $U*Q$ . This process requires  $nk^2$  flam. We could have accumulated the Householder transformations directly into  $U$ , which would give the same operation count. But the possibilities for optimizing cache utilization are better if we calculate  $U*Q$  as above (see §I:2.3).

#### 1.4. COMPUTATION OF REFINED AND HARMONIC RITZ VECTORS

In §§4.3 and 4.4, Chapter 4, we introduced refined and harmonic Ritz vectors. In general, both are somewhat expensive to compute. However, if the defining subspace for these vectors is a Krylov subspace, their computation simplifies. The purpose of this subsection is to give the details. In what follows, we will assume that

$$AU = UB + ub^H \quad (1.20)$$

is a orthonormal Krylov decomposition.

##### Refined Ritz vectors

If  $\mu$  is a Ritz value then the refined Ritz vector associated with  $\mu$  is the right singular vector of  $(A - \mu I)U$  whose singular value is smallest [see (4.16), Chapter 4]. From (1.20) we have

$$(A - \mu I)U = U(B - \mu I) + ub^H.$$

If we set

$$\hat{B}_\mu = \begin{pmatrix} B - \mu I \\ b^H \end{pmatrix}$$

we have

$$(A - \mu I)U = (U \ u)\hat{B}_\mu.$$

Since  $(U \ u)$  is orthonormal, the right singular vectors of  $(A - \mu I)U$  are the same as the right singular vectors of  $\hat{B}_\mu$ . Thus the computation of a refined Ritz vector can be reduced to computing the singular value decomposition of the low-order matrix  $\hat{B}_\mu$ .

##### Harmonic Ritz vectors

By Definition 4.11, Chapter 4, the pair  $(\kappa + \delta, Uw)$  is a harmonic Ritz pair if

$$U^H(A - \kappa I)^H(A - \kappa I)Uw = \delta U^H(A - \kappa I)^H Uw.$$

From (1.4) we have  $(A - \kappa I)U = U(B - \kappa I) + ub^H$ . Hence

$$U^H(A - \kappa I)^H(A - \kappa I)U = (B - \kappa I)^H(B - \kappa I) + bb^H$$

and

$$U^H(A - \kappa I)^H U = (B - \kappa I)^H.$$

It follows that

$$[(B - \kappa I)^H(B - \kappa I) + bb^H]w = \delta(B - \kappa I)^H w.$$

Thus we have reduced the problem of computing  $w$  to that of solving a small generalized eigenvalue problem, which can be done by the QZ algorithm (see §4.3, Chapter 2).

If we are willing to assume that  $(B - \kappa I)^H$  is well conditioned, we can multiply by  $(B - \kappa I)^{-H}$  to get the ordinary eigenvalue problem

$$[(B - \kappa I) + (B - \kappa I)^{-H}bb^H]w = \delta^H w.$$

## 1.5. NOTES AND REFERENCES

### Lanczos and Arnoldi

Contrary to our order of presentation, the Lanczos recurrence preceded the Arnoldi procedure. It is due to Cornelius Lanczos (pronounced Lantzosh), whose algorithm [157, 1950] has had a profound influence on the solution of large eigenproblems. Lanczos did not compute approximations to the eigenvalues from the matrix  $T_k$  (which he never explicitly formed), but instead used the recurrence to generate the characteristic polynomial of  $T_k$ , from which he computed eigenvalues. As indicated above (p. 279), he was aware that accurate approximations to certain eigenvalues could emerge early in the procedure. He called his method the *method of minimized iterations* because orthogonalizing  $Au$  against  $u_1, \dots, u_k$  minimizes the norm of the result. Lanczos also gave a biorthogonalization algorithm for non-Hermitian matrices involving right and left Krylov sequences.

The matrix form (1.12) of the Lanczos decomposition is due to Paige [195], whose ground-breaking thesis helped turn the Lanczos algorithm from a computational toy with promise into a viable algorithm for large symmetric eigenproblems.

As might be expected, there are several variants of the basic three-term recursion, of which (1.15) is an example. Paige [197] has given a rounding-error analysis of these variants. The two described here emerge with a clean bill of health, but the others may not perform well.

Arnoldi's chief contribution [5, 1951] seems to have been to generalize the one-sided Lanczos process to non-Hermitian matrices, noting that the Rayleigh quotient is Hessenberg.

What we have been calling an Arnoldi decomposition is often called an Arnoldi factorization, and indeed the relation (1.2) is a factorization of  $AU_k$ . However, the factorization is concealed by the second form (1.4), which is by far the most widely used. Consequently, we have opted for the slightly vaguer word "decomposition."

### Shift-and-invert enhancement and orthogonalization

The fact that a shift that closely approximates an eigenvalue can degrade convergence to other eigenpairs does not seem to be noted in the literature. One possible reason is that one does not generally have very accurate shifts. Moreover, as Example 1.5 shows, the shift has to be extremely accurate for the degradation to be significant.

### Krylov decompositions

The idea of relaxing the definition of an Arnoldi decomposition is due to Stewart [267], as are the systematic transformations of the resulting Krylov decompositions. However, some of these ideas are implicit in some algorithms for large eigenproblems and large linear systems. Most notably, Sorensen's implicitly restarted Arnoldi algorithm [246], to be treated in the next section, proceeds via an orthonormal Krylov decomposition, which is then truncated to yield an Arnoldi decomposition. For more, see §2.5.

## 2. THE RESTARTED ARNOLDI METHOD

Let

$$AU_k = U_k H_k + \beta_k u_{k+1} e_k^T$$

be an Arnoldi decomposition. We have already noted that the Ritz pairs associated with  $U_k$  can be computed from the Rayleigh quotient  $H_k$ . As  $k$  increases, some of these Ritz pairs will (we hope) approach eigenpairs of  $A$ . In principle, then, we can use Algorithm 1.1 to keep expanding the Arnoldi decomposition until the Ritz pairs we want have converged.

Unfortunately, for large problems our ability to expand the Arnoldi decomposition is limited by the amount of memory we can devote to the storage of  $U_k$ . For example, if  $A$  is of order 10,000,  $U_{100}$  requires almost 10 megabytes of memory to store in IEEE double precision. A possible cure is to move the vectors  $u_k$  to a backing store — say a disk — as they are generated. However, in order to orthogonalize  $Au_k$  against the previous vectors we will have to bring them back into main memory. Since disk transfers are slow, the algorithm will spend most of its time moving numbers back and forth between the disk and main memory.

An alternative is to restart the Arnoldi process once  $k$  becomes so large that we cannot store  $U_k$ . This, in fact, is the way the Arnoldi method is used in practice. In this chapter we will introduce two restarting methods: the implicit restarting method of Sorensen and a new method based on a Krylov–Schur decomposition. These methods are the subjects of §§2.1 and 2.2. We then turn to practical considerations: stability, convergence, and deflation.

## 2.1. THE IMPLICITLY RESTARTED ARNOLDI METHOD

The process of restarting the Arnoldi method can be regarded as choosing a new starting vector for the underlying Krylov sequence. A natural choice would be a linear combination of Ritz vectors that we are interested in. This, it turns out, is a special case of a general approach in which we apply a *filter polynomial* to the original starting vector. We will begin by motivating this process and then go on to show how the QR algorithm can be used to implement it.

### Filter polynomials

For the sake of argument let us assume that  $A$  has a complete system of eigenpairs  $(\lambda_i, x_i)$ , and suppose that we are interested in calculating the first  $k$  of these eigenpairs. Let  $u_1$  be a fixed vector. Then  $u_1$  can be expanded in the form

$$u_1 = \sum_{i=1}^k \gamma_i x_i + \sum_{i=k+1}^n \gamma_i x_i.$$

If  $p$  is any polynomial, we have

$$p(A)u_1 = \sum_{i=1}^k \gamma_i p(\lambda_i) x_i + \sum_{i=k+1}^n \gamma_i p(\lambda_i) x_i.$$

If we can choose  $p$  so that the values  $p(\lambda_i)$  ( $i = k+1, \dots, n$ ) are small compared to the values  $p(\lambda_i)$  ( $i = 1, \dots, k$ ), then  $u$  will be rich in the components of the  $x_i$  that we want and deficient in the ones that we do not want. Such a polynomial is called a *filter polynomial*.

There are many possible choices for a filter polynomial, but two suggest themselves immediately.

1. If the eigenvalues  $\lambda_{k+1}, \dots, \lambda_n$  lie in an interval  $[a, b]$  and the eigenvalues  $\lambda_1, \dots, \lambda_k$  lie without, we can take  $p$  to be a Chebyshev polynomial on  $[a, b]$ . By the properties of Chebyshev polynomials (see Theorem 3.7, Chapter 4), this polynomial has values on  $[a, b]$  that are small compared to values sufficiently outside  $[a, b]$ .
2. Suppose we have at hand a set of Ritz values  $\mu_1, \dots, \mu_m$ , and suppose that  $\mu_{k+1}, \dots, \mu_m$  correspond to the part of the spectrum we are not interested in. Then take  $p(t) = (t - \mu_{k+1}) \cdots (t - \mu_m)$ .

It should be observed that both choices accentuate the negative. The emphasis is on filtering out what we do not want, while what we want is left to fend for itself. For example, desired eigenvalues near the interval  $[a, b]$  in the first of the above alternatives will not be much enhanced.

### Implicitly restarted Arnoldi

Whatever the (considerable) attractions of filter polynomials, they are expensive to apply directly. If, for example,  $p$  is of degree  $m-k$ , then  $p(A)u_1$  requires  $m-k$  matrix-vector multiplications to compute. However, if  $u_1$  is the starting vector of an Arnoldi decomposition of order  $m$ , we can use shifted QR steps to compute the Krylov decomposition of  $\mathcal{K}_k[p(A)u_1]$  without any matrix-vector multiplications. This technique is called *implicit restarting*.

Before describing how implicit restarting is implemented, let us look at how it will be used. We begin with an Arnoldi decomposition

$$AU_k = U_k H_k + \beta_k u_{k+1} \mathbf{e}_k^T.$$

We then use Algorithm 1.1 to expand this decomposition to a decomposition

$$AU_m = U_m H_m + \beta_m u_{m+1} \mathbf{e}_m^T$$

of order  $m$ . The integer  $m$  is generally chosen so that the decomposition can fit into the available memory. At this point a filter polynomial  $p$  of degree  $m-k$  is chosen, and the implicit restarting process is used to reduce the decomposition to a decomposition

$$A\tilde{U}_k = \tilde{U}_k \tilde{H}_k + \tilde{\beta}_k \tilde{u}_{k+1} \mathbf{e}_k^T$$

of order  $k$  with starting vector  $p(A)u_1$ . This process of expanding and contracting can be repeated until the desired Ritz values have converged.

The contraction phase works with a factored form of the filter polynomial. Specifically, let

$$p(t) = (t - \kappa_1)(t - \kappa_2) \cdots (t - \kappa_{m-k}).$$

We incorporate the factor  $t - \kappa_1$  into the decomposition using an analogue of the basic QR step [see (2.1), Chapter 2]. Starting with the shifted decomposition

$$(A - \kappa_1 I)U_m = U_m(H_m - \kappa_1 I) + \beta_m u_{m+1} \mathbf{e}_m^T,$$

we write

$$(A - \kappa_1 I)U_m = U_m Q_1 R_1 + \beta_m u_{m+1} \mathbf{e}_m^T, \quad (2.1)$$

where  $Q_1 R_1$  is the QR factorization of  $H_m - \kappa_1 I$ . Postmultiplying by  $Q_1$ , we get

$$(A - \kappa_1 I)(U_m Q_1) = (U_m Q_1)(R_1 Q_1) + \beta_m u_{m+1} (\mathbf{e}_m^T Q_1).$$

Finally, we have

$$AU_m^{(1)} = U_m^{(1)} H_m^{(1)} + \beta_m u_{m+1} b_{m+1}^{(1)H}, \quad (2.2)$$

where

$$U_m^{(1)} = U_m Q_1, \quad H_m^{(1)} = R_1 Q_1 + \kappa_1 I, \quad \text{and} \quad b_{m+1}^{(1)T} = \mathbf{e}_m^T Q_1.$$

Note that  $H_k^{(1)}$  is the matrix that results from one QR step with shift  $\kappa_1$  applied to  $H_m$ .

There are five points to note about this procedure.

1. The matrix  $U_m^{(1)}$  is orthonormal.
2. The matrix  $H_m^{(1)}$  is Hessenberg, and at most its last subdiagonal element is zero. This follows from the proof of the preservation of Hessenberg form by the QR algorithm.
3. The matrix  $Q_1$  is upper Hessenberg. This follows from the fact that it can be written in the form

$$Q_1 = P_{12}^H P_{23}^H \cdots P_{m-1,m}^H,$$

where  $P_{j-1,j}$  is a rotation in the  $(j-1, j)$ -plane (see Figure 2.2, Chapter 2).

4. The vector  $b_{m+1}^{(1)H} = \mathbf{e}_m^T Q_1$  is the last row of  $Q$  and has the form

$$b_{m+1}^{(1)H} = (0 \ \cdots \ 0 \ q_{m-1,m}^{(1)} \ q_{m,m}^{(1)}) ;$$

i.e., only the last two components of  $b_{m+1}^{(1)}$  are nonzero.

5. The first column of  $U_m^{(1)} = U_m Q_1$  is a multiple of  $(A - \kappa_1 I)$ . For on post-multiplying (2.1) by  $\mathbf{e}_1$  and remembering that  $R_1$  is upper triangular, we get

$$(A - \kappa_1 I)u_1 = r_{11}^{(1)}u_1^{(1)}.$$

Since  $H_m$  is unreduced,  $r_{11}^{(1)}$  is nonzero, which establishes the result.

We may now repeat this process with  $\kappa_2, \dots, \kappa_{m-k}$ . The result will be a Krylov decomposition

$$AU_m^{(m-k)} = U_m^{(m-k)}H_m^{(m-k)} + \beta_m u_{m+1} b_{m+1}^{(m-k)H} \quad (2.3)$$

with the following properties.

1.  $U_m^{(m-k)}$  is orthonormal.
2.  $H_m^{(m-k)}$  is upper Hessenberg. At most its last  $m-k$  subdiagonal elements are zero.
3.  $Q = Q_1 \cdots Q_{m-k}$  is zero below its  $(m-k)$ th subdiagonal. (2.4)
4.  $b_{m+1}^{(m-k)H}$  is the last row of  $Q$ , and hence its first  $k-1$  components are zero.
5. The first column of  $U_m^{(m-k)}$  is a multiple of  $(A - \kappa_1 I) \cdots (A - \kappa_{m-k} I)u_1$ .

Thus the decomposition (2.3) has the Wilkinson diagram presented below for  $k = 3$  and  $m = 6$ :

$$A(u \ u \ u | u \ u \ u) = (u \ u \ u | u \ u \ u) \left( \begin{array}{ccc|ccc} h & h & h & h & h & h \\ h & h & h & h & h & h \\ 0 & h & h & h & h & h \\ \hline 0 & 0 & h & h & h & h \\ 0 & 0 & 0 & h & h & h \\ 0 & 0 & 0 & 0 & h & h \end{array} \right) + u(0 \ 0 \ q | q \ q \ q),$$

where the  $q$ 's are from the last row of  $Q$ . From this diagram it is easy to see that the first  $k$  columns of the decomposition can be written in the form

$$AU_k^{(m-k)} = U_k^{(m-k)} H_{kk}^{(m-k)} + h_{k+1,k} u_{k+1}^{(m-k)} \mathbf{e}_k^T + \beta_k q_{mk} u_{m+1} \mathbf{e}_k^T,$$

where  $U_k^{(m-k)}$  consists of the first  $k$  columns of  $U_m^{(m-k)}$ ,  $H_{kk}^{(m-k)}$  is the leading principal submatrix of order  $k$  of  $H_m^{(m-k)}$ , and  $q_{k,m}$  is from the matrix  $Q = Q_1 \cdots Q_{m-k}$ . Hence if we set

$$\begin{aligned}\tilde{U}_k &= U_k^{(m-k)}, \\ \tilde{H}_k &= H_{kk}^{(m-k)}, \\ \tilde{\beta}_k &= \|h_{k+1,k} u_{k+1}^{(m-k)} + \beta_k q_{mk} u_{m+1}\|_2, \\ \tilde{u}_{k+1} &= \tilde{\beta}_k^{-1} (h_{k+1,k} u_{k+1}^{(m-k)} + \beta_k q_{mk} u_{m+1}),\end{aligned}$$

then

$$A\tilde{U}_k = \tilde{U}_k \tilde{H}_k + \tilde{\beta}_k \tilde{u}_{k+1} \mathbf{e}_k^T$$

is an Arnoldi decomposition whose starting vector is proportional to  $(A - \kappa_1 I) \cdots (A - \kappa_{m-k} I) u_1$ . We have thus restarted the Arnoldi process with a decomposition of order  $k$  whose starting vector has been filtered by  $p(A)$ .

The elegance and economy of this procedure is hard to overstate. Not only do we avoid any matrix-vector multiplications in forming the new starting vector, but we also get its Arnoldi decomposition of order  $k$  for free. For large  $n$  the major cost will be in computing  $UQ$ .

A bit of terminology will help in our description of the algorithm. After we have applied the shifts  $\kappa_1, \dots, \kappa_j$ , we can truncate the decomposition as above to yield a decomposition of order  $m-j$ . We will call the integer  $m-j$  the *truncation index* for the current problem.

## Implementation

We have described the contraction algorithm for the implicitly restarted Arnoldi method in terms of the explicitly shifted QR algorithm. For single shifts there is no reason not to use this procedure. However, when  $A$  is real and the shift is complex, it is natural to use the implicit double shift procedure. The only new feature is that a double shift adds an additional two subdiagonals to the matrix  $Q$  and hence moves the truncation index back by two.

For simplicity, we will only describe the implicit single step method. For convenience we separate the final truncation from the implicit filtering. Algorithm 2.1 is an implementation of one step of filtering. Here are some comments.

- The algorithm is a straightforward implementation of the single implicit QR step.
- The algorithm requires  $O(m^2)$  work. A precise operation count is not necessary, since when  $n$  is large the applications of *FilterArn* will account for an insignificant part

Let

$$AU = UH + ub^H$$

be an Arnoldi decomposition of order  $m$ . Given a shift  $\kappa$ , this algorithm reduces the truncation index by one. The transformations generated by the algorithm are accumulated in the  $m \times m$  matrix  $Q$ .

1. *FilterArn*( $H, m, \kappa, Q$ )
2.  $h11mk = H[1, 1] - \kappa$
3.  $h21 = H[2, 1]$
4. *rotgen*( $h11mk, h21, c, s$ )
5. *rotapp*( $c, s, H[1, 1:n], H[2, 1:n]$ )
6. *rotapp*( $c, \bar{s}, H[1:3, 1], H[1:3, 2]$ )
7. *rotapp*( $c, \bar{s}, Q[1:m, 1], Q[1:m, 2]$ )
8. **for**  $j = 2$  **to**  $m-1$
9.     *rotgen*( $H[j, j-1], H[j+1, j-1], c, s$ )
10.    *rotapp*( $c, s, H[j, j:m], H[j+1, j:m]$ )
11.     $j2 = \min\{j+2, m\}$
12.    *rotapp*( $c, \bar{s}, H[1:j2, j], H[1:j2, j+1]$ )
13.    *rotapp*( $c, \bar{s}, Q[1:m, j], H[1:m, j+1]$ )
14. **end for**  $j$
15. **end FilterArn**

**Algorithm 2.1:** Filtering an Arnoldi decomposition

Given an Arnoldi decomposition

$$AU = \hat{U}\hat{H} = UH + \beta u\mathbf{e}_k^T$$

and an integer  $m > k$ , this algorithm expands the decomposition to one of order  $m$  and then contracts it back to one of order  $k$ .

1. **for**  $j = k+1$  **to**  $m$
2.     ***ExpandArn***( $\hat{U}$ ,  $\hat{H}$ )
3. **end for**  $k$
4. Determine shifts  $\kappa_1, \dots, \kappa_{m-k}$
5.  $H = \hat{H}[1:m, 1:m]$
6.  $\beta = \hat{H}[m+1, m]$
7.  $Q = I_m$
8. **for**  $j = 1$  **to**  $m-k$
9.     ***FilterArn***( $H$ ,  $m$ ,  $\kappa_i$ ,  $Q$ )
10. **end for**  $j$
11.  $u = H[k+1, k]*\hat{U}[:, k:m]*Q[:, k+1] + \beta*Q[m, k]*\hat{U}[:, m+1]$
12.  $\beta = \|u\|_2$
13.  $u = \beta^{-1}u$
14.  $U = (U*Q[:, 1:k] u)$
15.  $\hat{H} = \begin{pmatrix} H \\ \beta*\mathbf{e}_k^T \end{pmatrix}$

### Algorithm 2.2: The restarted Arnoldi cycle

---

of the work in the overall Arnoldi algorithm. In fact, we have ignored the opportunity to reduce the count by taking advantage of the special structure of  $Q$ .

- Transformations are not accumulated in  $U$ . It is much cheaper to accumulate them in the smaller matrix  $Q$  and apply  $Q$  to  $U$  at the end of the contraction process.

### The restarted Arnoldi cycle

We are now in a position to combine the expansion phase with the contraction phase. This is done in Algorithm 2.2.

When  $n$  is large, the bulk of the work in this algorithm is concentrated in the expansion phase and in the computation of  $U*Q[1:k]$ . The expansion phase requires  $m-k$  matrix-vector multiplications. From (1.10) we know that the Arnoldi step requires between  $2nj$  flam and  $4nj$  flam, so that the expansion phase requires between  $n(m^2-k^2)$  flam and  $2n(m^2-k^2)$  flam.

If we implement the product  $U*Q[:, 1:k]$  naively, the operation count is  $nmk$  flam. However, the elements of  $Q$  below the  $m-k$  subdiagonal are zero. If this is taken into

consideration the count can be reduced to  $n(km - \frac{1}{2}k^2)$ .

These counts are not easy to interpret unless we know the relative sizes of  $m$  and  $k$ . There is no general analysis that specifies an optimal relation, but the conventional wisdom is to take  $m$  to be  $2k$  or greater. Thus if we take  $m = 2k$ , we end up with the following table (in which RO stands for reorthogonalization).

|               | No RO              | Full RO            | (2.5) |
|---------------|--------------------|--------------------|-------|
| Naive $U*Q$   | $5nk^2$            | $8nk^2$            |       |
| Careful $U*Q$ | $4\frac{1}{2}nk^2$ | $7\frac{1}{2}nk^2$ |       |

The main conclusion to be drawn from this table is that it does not much matter how  $U*Q[:, 1:k]$  is formed. In fact, the naive scheme permits more blocking, so that in spite of the operation counts it may be faster (see §I:2.3).

For purposes of exposition we have assumed that  $m$  and  $k$  are fixed. However, they can be varied from cycle to cycle, as long as  $m$  is not so large that storage limits are exceeded.

The implicitly restarted Arnoldi method can be quite effective, as the following example shows.

**Example 2.1.** Let

$$A = \text{diag}(1, .95, .95^2, .95^3, \dots, .95^{99})$$

(the matrix of Example 3.1, Chapter 4). The implicitly restarted Arnoldi method was run with  $k = 5$ ,  $m = 10$ , and starting vector  $\mathbf{e}$ . The results are summarized in the following table.

| $A \times u$ | A       | IRA     | RV      |
|--------------|---------|---------|---------|
| 10           | 5.7e-02 | 9.8e-02 | 1.2e-01 |
| 15           | 2.0e-03 | 4.9e-03 | 5.4e-03 |
| 20           | 1.9e-05 | 8.7e-05 | 8.8e-05 |
| 25           | 4.8e-08 | 8.0e-07 | 8.0e-07 |

The columns contain the following information:

1. the number of matrix-vector multiplications,
2.  $\tan \angle(\mathbf{e}_1, \text{the full Krylov subspace})$ ,
3.  $\tan \angle(\mathbf{e}_1, \text{the restarted Arnoldi subspace})$ ,
4.  $\tan \angle(\mathbf{e}_1, \text{the Ritz vector from the restarted subspace})$ .

The implicitly restarted Arnoldi method is almost as good as the full Krylov sequence. Yet the latter requires 2,500 words of floating-point storage, whereas the former requires only 1,000.

It would be rash to take one offhand example as characteristic of behavior of the implicitly restarted Arnoldi method. But experience has shown it to be the best general method for non-Hermitian matrices when storage is limited.

### Exact shifts

We have already mentioned the possibility of choosing Ritz values—the eigenvalues of  $H$ —as shifts for the implicitly restarted Arnoldi method. When this is done the Rayleigh quotient assumes a special form in which the discarded Ritz values appear on the diagonal of the discarded half of the Rayleigh quotient. This fact is a consequence of the following theorem.

**Theorem 2.2.** *Let  $H$  be an unreduced Hessenberg matrix, and let  $\mu$  be an eigenvalue of  $H$ . If one QR step with shift  $\mu$  is performed on  $H$ , then the resulting matrix  $\tilde{H}$  has the form*

$$\tilde{H} = \begin{pmatrix} \tilde{H}_{11} & \tilde{h}_{12} \\ 0 & \mu \end{pmatrix}, \quad (2.6)$$

where  $H_{11}$  is unreduced.

**Proof.** The first part of the QR step consists of the reduction

$$Q^T(H - \mu I) = R \quad (2.7)$$

of  $(H - \mu I)$  to triangular form. This reduction is illustrated in Figure 2.2, Chapter 2, from which it is evident that all but the last of the diagonal elements of  $R$  are nonzero. But  $H - \mu I$  is not of full rank. Hence the last diagonal element of  $R$  must be zero. It follows that the last row of  $RQ$  is zero. Moreover, it follows from the representation of the formation of this product in Figure 2.3, Chapter 2, that all but the last of the subdiagonal elements of  $RQ$  are nonzero. Hence  $\tilde{H} = RQ + \mu I$  has the form (2.6), where  $\tilde{H}_{11}$  is unreduced. ■

By repeated application of Theorem 2.2 we get the following result.

**Corollary 2.3.** *Let  $\mu_m, \dots, \mu_{k+1}$  be eigenvalues of  $H_k$ . If the implicitly restarted QR step is performed with shifts  $\mu_m, \dots, \mu_{k+1}$ , then the matrix  $H_m^{(m-k)}$  in (2.3) has the form*

$$H_m^{(m-k)} = \begin{pmatrix} \tilde{H}_k & H_{12}^{(m-k)} \\ 0 & T^{(m-k)} \end{pmatrix}, \quad (2.8)$$

where  $T^{(m-k)}$  is an upper triangular matrix with Ritz values  $\mu_{k+1}, \dots, \mu_m$  on its diagonal.

The matrix  $\tilde{H}_k$  in (2.8) is the Rayleigh quotient for the implicitly restarted Arnoldi algorithm. Because  $H_m^{(m-k)}$  is block triangular,  $\tilde{H}_k$  must contain the Ritz values that were not used in the restarting process. This means that we can choose to keep certain Ritz values and discard others. Specifically, at the end of the expansion phase the Ritz values—i.e., the eigenvalues of  $H_m$ —are calculated and divided into two classes:

those we wish to retain and those we wish to discard. The latter are used as shifts in the restarting algorithm.

In this way the restarting process can be used to steer the Arnoldi algorithm to regions of interest. For example, if we are using shift-and-invert enhancement, we will be interested in the largest eigenvalues, and it is natural to retain the largest Ritz values. In stability computations, where an eigenvalue with a positive real part indicates instability, we might retain the Ritz values having the largest real part.

A consequence of Corollary 2.3 is that in exact arithmetic the quantity  $H[k+1, k]$  in statement 11 of Algorithm 2.2 is zero, so that the computation of  $\beta$  and  $u$  can be simplified. In the presence of rounding error, however,  $H[k+1, k]$  can be far from zero. We now turn to a discussion of this phenomenon.

### Forward instability

Theorem 2.2 is a classic example of the limitations of mathematics in the face of rounding error. Given an exact shift, the matrix that one computes with rounding error can fail to deflate — i.e., its  $(m, m-1)$ -element can be far from zero. This phenomenon, which is called forward instability of the QR step, does not mean that we cannot use Algorithm 2.2. In fact, as we shall see later (Theorem 2.5), the Arnoldi relation  $AU = UH + \beta u$  will continue to hold to working accuracy. However, if a particular step is forward unstable, we cannot guarantee that the undesirable Ritz values have been purged from the decomposition. In fact, there is a danger of an unwanted Ritz value becoming permanently locked into the decomposition.

An important algorithmic implication of forward instability is that we cannot deflate the decomposition after the application of each exact shift. We must continue to work with the entire matrix  $H$  as in Algorithm 2.1. Fortunately, when  $n$  is large, the work saved by deflation is negligible.

Although forward instability is a real phenomenon, it should be kept in perspective. It cannot generate instability in the Arnoldi process itself. At worst it can only retard its progress. That is why ARPACK, the definitive implementation of the implicitly restarted Arnoldi method, has been and is used with great success to solve large eigenproblems.

## 2.2. KRYLOV–SCHUR RESTARTING

The implicitly restarted Arnoldi algorithm with exact shifts is a technique for moving Ritz values out of an Arnoldi decomposition. As we have seen, forward instability in the QR step can compromise the progress of the algorithm (though not its stability). In this subsection we describe a more reliable method for purging Ritz values. It is based on two observations. First, it is easier to move eigenvalues around in a triangular matrix than in a Hessenberg matrix. Second, if a Krylov decomposition can be partitioned in the form

$$A(U_1 \ U_2) = (U_1 \ U_2) \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix} + u(b_1^H \ b_2^H),$$

then  $AU_1 = U_1B_{11} + ub_1^H$  is also a Krylov decomposition. In other words a Krylov decomposition splits at any point where its Rayleigh quotient is block triangular. Our restarting algorithm then is to compute the Schur decomposition of the Rayleigh quotient, move the desired eigenvalues to the beginning, and throw away the rest of the decomposition. We will call the process *Krylov–Schur restarting*.

### Exchanging eigenvalues and eigenblocks

Since the Krylov–Schur method requires us to be able to move eigenvalues around in a triangular matrix, we will begin with a sketch of how this is done. The first thing to note is that to move an eigenvalue from one place to another it is sufficient to be able to exchange it with either one of its neighboring eigenvalues. More precisely, let a triangular matrix be partitioned in the form

$$\begin{pmatrix} A & B & C \\ 0 & S & D \\ 0 & 0 & E \end{pmatrix},$$

where

$$S = \begin{pmatrix} s_{11} & s_{12} \\ 0 & s_{22} \end{pmatrix}.$$

If we can find a unitary transformation  $Q$  such that

$$Q^H S Q = \begin{pmatrix} s_{22} & \hat{s}_{12} \\ 0 & s_{11} \end{pmatrix},$$

then the eigenvalues  $s_{11}$  and  $s_{22}$  in the matrix

$$\begin{pmatrix} A & BQ & C \\ 0 & Q^H S Q & Q^H D \\ 0 & 0 & E \end{pmatrix}$$

will have traded places. If we have an algorithm to perform such exchanges, then we can move any eigenvalue on the diagonal of a triangular matrix to another position on the diagonal by a sequence of exchanges.

However, it is not enough to be able to exchange just eigenvalues. If the eigenproblem in question is real, we will generally work with real Schur forms. Thus we must not only be able to exchange eigenvalues but also the  $2 \times 2$  blocks containing complex eigenvalues.

To derive our algorithms we will consider the real case. Let

$$S = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix},$$

where  $S_{ii}$  is of order  $n_i$  ( $i = 1, 2$ ). There are four cases to consider.

1.  $n_1 = 1, n_2 = 1.$
2.  $n_1 = 2, n_2 = 1.$
3.  $n_1 = 1, n_2 = 2.$
4.  $n_1 = 2, n_2 = 2.$

We will now sketch algorithms for each case.

The first two cases can be treated by what amounts to the first step in a reduction to Schur form (see the proof of Theorem 1.12, Chapter 1). Let the matrix in question be

$$S = \begin{pmatrix} S_{11} & s_{12} \\ 0 & s_{22} \end{pmatrix},$$

where  $S_{11}$  is of order one or two. Let  $x$  be a normalized eigenvector corresponding to  $s_{22}$  and let  $Q = (x \ Y)$  be orthogonal. Then it is easily verified that

$$Q^T S Q = \begin{pmatrix} s_{22} & \hat{s}_{12}^T \\ 0 & \hat{S}_{11} \end{pmatrix}.$$

Note that although  $\hat{S}_{11}$  need not be equal  $S_{11}$ , the two matrices must have the same eigenvalues (in fact, they are similar).

The third case can be treated by the variant of Schur reduction introduced in connection with the QR algorithm [see (2.2), Chapter 2]. Let

$$S = \begin{pmatrix} s_{11} & s_{12}^T \\ 0 & S_{22} \end{pmatrix}.$$

Let  $y$  be a normalized left eigenvector corresponding to  $s_{11}$  and let  $Q = (X \ y)$  be orthogonal. Then

$$Q^T S Q = \begin{pmatrix} \hat{S}_{22} & \hat{s}_{12} \\ 0 & s_{11} \end{pmatrix}.$$

The last case,  $n_1 = n_2 = 2$ , is more complicated. Let

$$S = \begin{pmatrix} S_{11} & S_{12}^T \\ 0 & S_{22} \end{pmatrix}.$$

Let  $(S_{22}, X)$  be an orthonormal eigenpair, and let  $Q = (X \ Y)$  be orthogonal. Then by Theorem 1.6, Chapter 4,

$$Q^T S Q = \begin{pmatrix} \hat{S}_{22} & \hat{S}_{12} \\ 0 & \hat{S}_{11} \end{pmatrix}.$$

Since  $\hat{S}_{22}$  is the Rayleigh quotient  $X^T S X$  it must have the same eigenvalues as  $S_{22}$ , and similarly for  $\hat{S}_{11}$ .

The problem, then, is to compute the eigenbasis  $X$ . The procedure is to compute a nonorthonormal eigenbasis and then orthogonalize it. Specifically let the eigenbasis be  $(P^T \ I)^T$ , where  $P$  is to be determined. Then

$$\begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} \begin{pmatrix} P \\ I \end{pmatrix} = \begin{pmatrix} P \\ I \end{pmatrix} S_{22}.$$

From the first row of this equation, we get the Sylvester equation

$$S_{11}P - PS_{22} = -S_{12}, \quad (2.9)$$

which can be solved for  $P$ . In this case we do not use Algorithm 1.1, Chapter 1, to compute  $P$  but instead note that (2.9) is a set of four linear equations in four unknowns, which can be solved by any stable method. Once we have computed  $P$ , we can compute the QR decomposition

$$\begin{pmatrix} I \\ P \end{pmatrix} = (X \ Y) \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

In particular,  $Q = (X \ Y)$  can be represented as the product of two Householder transformations.

The algorithms for the first three cases have stable implementations. For the first case, the eigenvector can be computed by inspection. For the other two, they can be computed stably by solving a  $2 \times 2$  system of equations. It is possible for this system to be singular, but replacing zero pivot elements with a suitably small number will allow the computation to continue (see Algorithm 2.9, Chapter 2). Once the eigenvector has been computed, the matrix  $Q$  can be taken to be a Householder transformation (or a plane rotation when  $n_1 = n_2 = 1$ ), and it can be applied to the matrix in which  $S$  is embedded. These methods are stable in the usual sense.

The algorithm for the fourth case is not provably stable, although it works well in practice and its failure can be detected. Specifically, after the transformation the matrix  $Q^T S Q$  will have the form

$$\begin{pmatrix} \hat{S}_{22} & \hat{S}_{12} \\ E & \hat{S}_{11} \end{pmatrix}.$$

To complete the algorithm, the matrix  $E$  must be set to zero. Consequently, if  $E$  is too large — say  $\|E\|_\infty \geq 10\|S\|_\infty \epsilon_M$  — then the algorithm is deemed to have failed.

For more details on these algorithms see the LAPACK subroutine `xLAEXC`.

### The Krylov–Schur cycle

We will now describe the basic cycle of the Krylov–Schur method. To keep the exposition simple, we will assume that the matrix in question is complex, so that we do not have to be concerned with real Schur forms.

The Krylov–Schur cycle begins with a Krylov–Schur decomposition

$$AU_k = U_{k+1}\hat{S}_k = U_k S_k + u_{k+1} b_{k+1}^H,$$

where  $U_{k+1}$  is orthonormal and  $S_k$  is upper triangular. To get such a decomposition initially, one chooses a starting vector, expands it to an Arnoldi decomposition, and transforms it by a unitary similarity to triangular form.

The expansion phase is exactly like the expansion phase for the Arnoldi method. In fact, one can use *ExpandArn* (Algorithm 1.1) to implement it. However, the expanded Rayleigh quotient now has the form illustrated below for  $k = 4$  and  $m = 8$ :

$$\begin{pmatrix} s & s & s & s & t & t & t & t \\ 0 & s & s & s & t & t & t & t \\ 0 & 0 & s & s & t & t & t & t \\ 0 & 0 & 0 & s & t & t & t & t \\ b & b & b & b & t & t & t & t \\ 0 & 0 & 0 & 0 & b & t & t & t \\ 0 & 0 & 0 & 0 & 0 & b & t & t \\ 0 & 0 & 0 & 0 & 0 & 0 & b & t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & b \end{pmatrix}. \quad (2.10)$$

Writing the corresponding Krylov decomposition in the form

$$AU_m = U_m T_m + \beta_m u_{m+1} e_m^T,$$

we now determine an orthogonal matrix  $Q$  such that  $\tilde{S}_m = Q^H T_m Q$  is upper triangular and transform the decomposition to the form

$$A\hat{U}_m = \hat{U}_m S_m + u_{m+1} b_{m+1}.$$

Finally, we use the exchange algorithms to transform the eigenvalues we want to keep to the beginning of  $S_m$  and truncate the decomposition.

Algorithm 2.3 implements this scheme. Here are two comments.

- The work involved in the expansion phase is the same as for the Arnoldi process. For large  $n$  the work in the contraction phase is concentrated in the formation of the product  $U * Q[:, 1:k]$ . As (2.5) shows this is about the same as for implicit restarting. Thus the implicitly restarted Arnoldi cycle and the Krylov–Schur cycle are essentially equivalent in the amount of work they perform.
- In the computation of the Schur decomposition in statement 7, minor savings can be had in the initial reduction of  $S$  to Hessenberg form, since by (2.10)  $S$  is already partially reduced. When  $n$  is large, the savings are not worth the trouble.

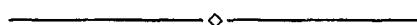
Given a Krylov–Schur decomposition

$$AU = \hat{U}\hat{S} = US + u_{k+1}b_{k+1}^T$$

of order  $k$  and an integer  $m > k$ , this algorithm expands the decomposition to one of order  $m$  and then contracts it back to one of order  $k$ .

1.  $\hat{T} = \hat{S}$
2. **for**  $j = k+1$  **to**  $m$
3.     *ExpandArn*( $\hat{U}$ ,  $\hat{T}$ )
4. **end for**  $k$
5.  $T = \hat{T}[1:m, 1:m]$
6.  $\beta = \hat{T}[m+1, m]$
7. Determine  $Q$  such that  $S = Q^H T Q$  is triangular
8. Select  $m-k$  Ritz values and move them to the end of  $S$ , accumulating the transformations in  $Q$
9.  $S = S[1:k, 1:k]$
10.  $b^H = \beta * Q[m, 1:k]$
11.  $U = U * Q[:, 1:k]$

**Algorithm 2.3:** The Krylov–Schur cycle



### The equivalence of Arnoldi and Krylov–Schur

Given that the implicitly restarted Arnoldi method with exact shifts and the Krylov–Schur method both proceed by moving Ritz values out of their respective decompositions, it is natural to conclude that if we start from equivalent decompositions and remove the same Ritz values then we will end up with equivalent decompositions. That is indeed the case, although it requires some proving.

**Theorem 2.4.** *Let*

$$\mathbb{P} := AU = UH + \beta ue_k^T$$

*be an unreduced Arnoldi decomposition and let*

$$\mathbb{Q} := AV = VS + ub^H$$

*be an equivalent Krylov–Schur form. Suppose that an implicitly restarted Arnoldi cycle is performed on  $\mathbb{P}$  and a Krylov–Schur cycle is performed on  $\mathbb{Q}$ . If the same Ritz values are discarded in both and those Ritz values are distinct from the other Ritz values, then the resulting decompositions are equivalent.*

**Proof.** We must show that the subspaces associated with the final results are the same. First note that the expansion phase results in equivalent decompositions. In fact, since  $\mathcal{R}(U) = \mathcal{R}(V)$  and in both cases we are orthogonalizing the vectors  $Au, A^2u, \dots$ , the vectors  $u_{k+2} \dots u_{m+1}$  and  $v_{k+2}, \dots, v_{m+1}$  are the same up to multiples of modulus one.

Now assume that both algorithms have gone through the expansion phase and have moved the unwanted Ritz values to the end of the decomposition. At this point denote the first decomposition by

$$\hat{\mathbb{P}} := A\hat{U} = U\hat{H} + \hat{\beta}\hat{u}\hat{e}_m^T$$

and the second by

$$\hat{\mathbb{Q}} := A\hat{V} = \hat{V}\hat{S} + \hat{u}\hat{b}^H.$$

Note that for both methods the final truncation leaves the vector  $\hat{u}$  unaltered. Since  $\hat{V} = \hat{U}W$  for some unitary  $W$ , we have

$$\hat{S} = \hat{V}^H A\hat{V} = W^H \hat{U}^H A\hat{U}W = W^H \hat{H}W.$$

Thus  $\hat{H}$  and  $\hat{S}$  are similar and have the same Ritz values. Thus it makes sense to say that both methods reject the same Ritz values.

Let  $P$  be the unitary transformation applied to the Rayleigh quotient in  $\hat{\mathbb{P}}$  and let  $Q$  be the one applied to the Rayleigh quotient of  $\hat{\mathbb{Q}}$ . Then we must show that the subspaces spanned by  $\hat{U}P[:, 1:k]$  and  $\hat{V}Q[:, 1:k]$  are the same (see statement 14 in Algorithm 2.2 and statement 11 in Algorithm 2.3). For brevity, set  $P_k = P[:, 1:k]$  and  $Q_k = Q[:, 1:k]$ .

By construction  $\mathcal{R}(P_k)$  spans the eigenspace  $\mathcal{P}$  of Schur vectors of  $\hat{H}$  corresponding to the retained Ritz values. Likewise,  $\mathcal{R}(Q_k)$  spans the eigenspace  $\mathcal{Q}$  of Schur vectors of  $\hat{S}$  corresponding to the retained Ritz values. By hypothesis each of these eigenspaces are simple and hence are unique (page 246). Since  $W^H \hat{S} W = \hat{H}$ , the matrix  $W^H \hat{P}_k$  spans  $\mathcal{Q}$ . Hence there is a unitary matrix  $R$  such that  $Q_k = W^H P_k R$ . We then have

$$\hat{V} Q_k = \hat{U} W W^H P_k R = \hat{U} P_k R.$$

It follows that  $\hat{V} Q_k$  and  $\hat{U} P_k$  span the same subspace. ■

This theorem says that the implicitly restarted Arnoldi method with exact shifts and the Krylov–Schur method with the same shifts are doing the same thing. In particular, the Krylov–Schur method is restarting with a vector filtered by a polynomial whose roots are the rejected Ritz values. This fact is not at all obvious from the algorithmic description of the Krylov–Schur method.

### Comparison of the methods

The implicitly restarted Arnoldi method and the Krylov–Schur method each have their place. The former can restart with an arbitrary filter polynomial, something the latter cannot do. On the other hand, when it comes to exact shifts the Krylov–Schur method is to be preferred because exchanging eigenvalues in a Schur form is a more reliable process than using implicit QR steps to deflate.

The Krylov–Schur method can be used as an adjunct to the implicitly restarted Arnoldi method. For example, if the latter produces a persistent unwanted Ritz value, the decomposition can be reduced to Krylov–Schur form, the offending eigenvalue swapped out, and Algorithm 1.3 used to return to an Arnoldi decomposition. Although this seems like a lot of work, if  $n$  is large and we delay multiplying transformations into the decomposition until the end of the process, it effectively costs no more than a contraction step. We will see in the next subsection that a similar procedure can be used to deflate converged Ritz pairs.

## 2.3. STABILITY, CONVERGENCE, AND DEFLATION

In this subsection we will consider some of the practicalities that arise in the implementation of the Arnoldi method. We will first treat the behavior of the methods in the presence of rounding error. We will then discuss tests for the convergence of approximate eigenpairs. Finally, we will show how to deflate converged eigenpairs from a Krylov decomposition. Since the Arnoldi and Krylov–Schur decompositions are both orthonormal Krylov decompositions, we will present our results in the more general notation of the latter.

### Stability

Provided we take due care to reorthogonalize, the algorithms of this section are stable in the usual sense.

**Theorem 2.5.** *Let*

$$AU_k \cong U_{k+1}\hat{B}_k$$

*be an orthogonal Krylov decomposition computed by the implicitly restarted Arnoldi method or the Krylov–Schur method. Then there is an orthonormal matrix  $\tilde{U}_{k+1}$  satisfying*

$$\|\tilde{U}_{k+1} - U_{k+1}\|_2 \leq \gamma \epsilon_M \quad (2.11)$$

*and a matrix  $E$  satisfying*

$$\|E\| \leq \gamma \|A\|_2 \epsilon_M \quad (2.12)$$

*such that*

$$(A + E)\tilde{U}_k = \tilde{U}_{k+1}\hat{B}_k. \quad (2.13)$$

*Here  $\gamma$  is a generic constant—in general different in (2.11) and (2.13)—that depends on the size of the problem and the number of cycles performed by the algorithm.*

*Alternatively, there is a matrix  $F$  satisfying*

$$\|F\| \leq \gamma \|A\|_2 \epsilon_M$$

*such that*

$$AU_k = U_{k+1}\hat{B}_k + F.$$

For a sketch of the proof, see the notes and references.

This theorem has three implications.

- The generic constant  $\gamma$  grows only slowly as the algorithm in question progresses. Thus the algorithm can be continued for many iterations without fear of instability.
  - Since  $B = \tilde{U}^H A \tilde{U}$  is a Rayleigh quotient of  $A$ , the eigenpairs  $(\mu_i, w_i)$  of  $B$  are primitive Ritz pairs of  $A + E$ .
  - The vectors  $U_k w_i$  are near the Ritz vectors  $\tilde{U}_k w_k$ .
- 

It follows that unless the Ritz pairs are very sensitive to perturbations in  $A$ , they will be accurately computed. Thus we can take the output of our algorithms at face value.

### Stability and shift-and-invert enhancement

When shift-and-invert enhancement with shift  $\kappa$  is combined with the Arnoldi process, the vector  $v = (A - \kappa I)^{-1}u$  will be computed by solving the system

$$A_\kappa v = u,$$

where  $A_\kappa = A - \kappa I$ , and we must take into account the errors generated in solving this system. In most applications  $A_\kappa$  will be large and sparse, and the system must be solved by a direct sparse method or by an iterative method. The treatment of each of these methods is worth a volume in itself. However, we can say that the computed solution will satisfy

$$(A_\kappa + F)v = u, \quad (2.14)$$

where

$$\|F\| \leq \gamma \|A_\kappa\| \epsilon_M.$$

Depending on the stability properties of the direct solver or the convergence criterion used by the iterative method, it is possible for the constant  $\gamma$  to be large.

To relate the backward error in (2.14) to  $A_\kappa^{-1}$ , which replaces  $A$  in (2.12), we must show how  $F$  in (2.14) propagates to  $A_\kappa^{-1}$ . From the first-order expansion

$$(A_\kappa + F)^{-1} \cong A_\kappa^{-1} - A_\kappa^{-1} F A_\kappa^{-1} \equiv A_\kappa^{-1} - G,$$

we see that the backward error in  $A_\kappa^{-1}$  is approximately bounded by

$$\|G\| \leq (\gamma \|A_\kappa\| \|A_\kappa^{-1}\|) \|A_\kappa^{-1}\| \epsilon_M. \quad (2.15)$$

This bound compares unfavorably with (2.12) in two ways. First, the constant  $\gamma$  can be larger than its counterpart in (2.12), as we have already pointed out. Second, if  $\kappa$  is very near an eigenvalue of  $A$ , then the *condition number*  $\|A_\kappa\| \|A_\kappa^{-1}\|$  will be large and will magnify the multiplier  $(\gamma \|A_\kappa\| \|A_\kappa^{-1}\|)$  in (2.15).

To put things less formally, instability in a direct solver or a lax convergence criterion in an iterative method can result in a comparatively large backward error in solving the system  $A_\kappa v = u$ . Ill conditioning of  $A_\kappa$ , due to the shift being near an eigenvalue of  $A$ , can further magnify the error.

All this suggests that one must be careful in using shifts very near an eigenvalue of  $A$ , something we have already noted in Example 1.5. However, the above analysis is quite crude, and in practice accurate shifts may be less damaging than it suggests. Unfortunately, there is little in the literature to guide us.

### Convergence criteria

We have already stressed in connection with the power method (p. 61) that an appropriate criterion for accepting a purported eigenpair  $(\mu, z)$  is that the residual  $r = Az - \mu z$

be sufficiently small. More generally, if  $(M, Z)$  is an approximate orthonormal eigenpair with a small residual

$$R = AZ - ZM, \quad (2.16)$$

then by Theorem 2.8, Chapter 4, there is a matrix  $E$  satisfying

$$\|E\|_p = \|R\|_p, \quad p = 2, F, \quad (2.17)$$

such that  $(M, Z)$  is an exact eigenpair of  $A + E$ . Thus if

$$\|R\|_F \leq \text{tol} \cdot \|A\|_F \quad (2.18)$$

for some tolerance tol, then  $(M, Z)$  is an exact eigenpair of  $\tilde{A}$ , where the (normwise) relative error in  $\tilde{A}$  is tol. Thus, (2.18) provides a convergence criterion for determining if eigenpairs produced by an algorithm have converged.

At first glance, the residual (2.16) would appear to be expensive to compute. For if  $Z$  has  $p$  columns, the computation requires  $p$  matrix-vector multiplications, not to mention the arithmetic operations involved in computing  $ZM$ . However, if  $Z = UW$ , where  $U$  is from an orthonormal Krylov decomposition, the residual is cheap to compute, as the following theorem shows.

**Theorem 2.6.** Let

$$AU = UB + ub^H$$

be an orthonormal Krylov decomposition and let  $(M, Z) = (M, UW)$  be an orthonormal pair. Then

$$\|AZ - ZM\|_F^2 = \|BW - WM\|_F^2 + \|b^H W\|_F^2. \quad (2.19)$$

**Proof.** It is easily verified that

$$AZ - ZM = AUW - UWM = U(BW - WM) + ub^H W. \quad (2.20)$$

Since  $u^H U = 0$ ,

$$\begin{aligned} \|AZ - ZM\|_F^2 &= \text{trace}[(AZ - ZM)^H (AZ - ZM)] \\ &= \text{trace}[(BW - WM)^H (BW - WM)] \\ &\quad + \text{trace}[(b^H W)^H (b^H W)] \\ &= \|BW - WM\|_F^2 + \|b^H W\|_F^2. \quad \blacksquare \end{aligned}$$

There are three comments to be made about this theorem.

- We have chosen to work with eigenspaces because that is what is produced by complex eigenvalues in a real Schur form. But for the most part, we will be concerned with ordinary eigenvalues and their eigenvectors.
- If  $(M, W)$  is a primitive Ritz pair, then  $BW - WM = 0$ , and the expression (2.19) reduces to  $\|b^H W\|_2$ . If the decomposition is an Arnoldi decomposition, so that  $b^H = \beta e_K^H$ , then  $\|R\|_F$  is just the norm of the last row of  $W$ . Moreover:

If  $(\mu, z) = (\mu, Uw)$  is a primitive Ritz pair of the Arnoldi decomposition  $AU = UH + \beta e_k^T$ , then

$$\|Az - \mu z\|_2 = |\beta||\omega_k|,$$

where  $\omega_k$  is the last component of  $w$ .

- If  $(M, W)$  is not a Ritz pair, then the formula (2.19) cannot be simplified. However, by Theorem 2.6, Chapter 4, it will be minimized when  $M$  is the Rayleigh quotient  $W^H B W$ . In particular, when checking the convergence of refined or harmonic Ritz vectors, one should use the Rayleigh quotient, not the refined or harmonic Ritz value.

In practice, we may not be able to compute  $\|A\|_F$ —especially if we are using shift-and-invert enhancement. However, since we are generally concerned with exterior eigenvalues, the norm of the Ritz block  $M$  is often a ballpark approximation to  $\|A\|_F$ . Thus we might say that a Ritz pair  $(M, UW)$  has converged if

$$\|b^H W\|_2 \leq_F \text{tol} \cdot \|M\|_F.$$

When the concern is with eigenvectors rather than eigenspaces, the criterion becomes

$$\|b^H w\|_2 \leq_F \text{tol} \cdot |\mu|. \quad (2.21)$$

### Convergence with shift-and-invert enhancement

As in the case of stability, when shift-and-invert enhancement is used—say with shift  $\kappa$ —the residual  $R$  in (2.17) relates to the shifted matrix. Specifically, if we are concerned with the approximate eigenpair  $(\mu, z)$ , then

$$r = (A - \kappa I)^{-1}z - \mu z. \quad (2.22)$$

In terms of the original matrix  $A$ , the approximate eigenvalue is  $\kappa + \mu^{-1}$ , and from (2.22) we have the residual

$$Az - (\kappa + \mu^{-1})z = -\mu^{-1}(A - \kappa I)r.$$

It follows that for the relative backward error in  $A$  not to be greater than  $\text{tol}$ , we must have

$$\frac{\|A - \kappa I\|_F}{\|A\|_F} \frac{\|r\|_2}{|\mu|} \leq \text{tol}.$$

Now it is reasonable to assume that  $\|A - \kappa I\|_F$  and  $\|A\|_F$  are of the same order of magnitude. Hence our convergence test becomes

$$\|r\|_2 \leq \text{tol} \cdot |\mu|.$$

This is exactly the same as the criterion (2.21) for Ritz pairs.

### Deflation

We say a Krylov decomposition has been deflated if it can be partitioned in the form

$$A(U_1 \ U_2) = (U_1 \ U_2) \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix} + u(0 \ b_2^H).$$

When this happens, we have  $AU_1 = U_1B_{11}$ , so that  $U_{11}$  spans an eigenspace of  $A$ .

There are two advantages to deflating a converged eigenspace. First by freezing it at the beginning of the Krylov decomposition we insure that the remaining space of the decomposition remains orthogonal to it. In particular, this gives algorithms the opportunity to compute more than one independent eigenvector corresponding to a multiple eigenvalue.

The second advantage of the deflated decomposition is that we can save operations in the contraction phase of an Arnoldi or Krylov–Schur cycle. The expansion phase does not change, and we end up with a decomposition of the form

$$A(U_1 \ U_2 \ U_3) = (U_1 \ U_2 \ U_3) \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ 0 & B_{22} & B_{23} \\ 0 & B_{23} & B_{33} \end{pmatrix} + \beta u e_m^T.$$

Now since  $B_{11}$  is uncoupled from the rest of the Rayleigh quotient, we can apply all subsequent transformations exclusively to the eastern part the Rayleigh quotient and to  $(U_2 \ U_3)$ . If the order of  $B_{11}$  is small, the savings will be marginal; but as its size increases during the course of the algorithm, the savings become significant.

Of course, we will never have an exact eigenspace in our decompositions. The following theorem relates the residual norm of an approximate subspace to the quantities we must set to zero to deflate.

**Theorem 2.7.** *Let*

$$AU = UB + ub^H$$

*be an orthonormal Krylov decomposition, and let  $(M, Z) = (M, UW)$  be an orthonormal pair. Let  $(W \ W_\perp)$  be unitary, and set*

$$\tilde{B} = \begin{pmatrix} W^H \\ W_\perp^H \end{pmatrix} B(W \ W_\perp) = \begin{pmatrix} \tilde{B}_{11} & \tilde{B}_{12} \\ \tilde{B}_{21} & \tilde{B}_{22} \end{pmatrix}$$

*and*

$$\tilde{b}^H = b^H(W \ W_\perp) = (\tilde{b}_1^H \ \tilde{b}_2^H).$$

*Then*

$$\|AZ - ZM\|_F^2 = \|\tilde{B}_{21}\|_F^2 + \|\tilde{b}_1\|_F^2 + \|\tilde{B}_{11} - M\|_F^2. \quad (2.23)$$

**Proof.** Let

$$(\tilde{U}_1 \ \tilde{U}_2) = U(W \ W_\perp).$$

From (2.20), we have

$$\begin{aligned} AZ - ZM &= U(W \ W_\perp) \left[ \begin{pmatrix} W^H \\ W_\perp^H \end{pmatrix} B(W \ W_\perp) \begin{pmatrix} I \\ 0 \end{pmatrix} - \begin{pmatrix} I \\ 0 \end{pmatrix} M \right] \\ &\quad + ub^H(W \ W_\perp) \begin{pmatrix} I \\ 0 \end{pmatrix} \\ &= (\tilde{U}_1 \ \tilde{U}_2) \begin{pmatrix} \tilde{B}_{11} - M \\ \tilde{B}_{21} \end{pmatrix} + u\tilde{b}_1^H. \end{aligned}$$

The theorem now follows on taking norms. ■

To see the consequences of this theorem, suppose that  $AZ - ZM$  is small, and, using  $(W \ W_\perp)$ , we transform the Krylov decomposition  $AU - UB = ub^H$  to the form

$$A(\tilde{U}_1 \ \tilde{U}_2) = (\tilde{U}_1 \ \tilde{U}_2 \ u) \begin{pmatrix} \tilde{B}_{11} & \tilde{B}_{12} \\ \tilde{B}_{21} & \tilde{B}_{22} \\ \tilde{b}_1^H & \tilde{b}_2^H \end{pmatrix}. \quad (2.24)$$

Then by (2.23)

$$\left\| \begin{pmatrix} \tilde{B}_{21} \\ \tilde{b}_1^H \end{pmatrix} \right\|_F \leq \|AZ - ZM\|_F, \quad (2.25)$$

with equality if and only if  $M$  is the Rayleigh quotient  $W^H B W$ . Now if the residual norm  $\|AZ - ZM\|_F$  is sufficiently small, we may set  $\tilde{B}_{21}$  and  $\tilde{b}_1$  to zero to get the approximate decomposition

$$A(\tilde{U}_1 \ \tilde{U}_2) \cong (\tilde{U}_1 \ \tilde{U}_2) \begin{pmatrix} \tilde{B}_{11} & \tilde{B}_{12} \\ 0 & \tilde{B}_{22} \end{pmatrix} + u(0 \ b_2^H). \quad (2.26)$$

Thus we have deflated our problem by making small changes in the Arnoldi decomposition.

### Stability of deflation

The deflation procedure that leads to (2.26) is backward stable. If we restore the quantities that were zeroed in forming (2.26), we get the following relation:

$$A(\tilde{U}_1 \ \tilde{U}_2) = (\tilde{U}_1 \ \tilde{U}_2) \begin{pmatrix} \tilde{B}_{11} & \tilde{B}_{12} \\ 0 & \tilde{B}_{22} \end{pmatrix} + u(0 \ b_2^H) + \tilde{U}_2 \tilde{B}_{21} + u\tilde{b}_1^H.$$

If we write this decomposition in the form

$$A\tilde{U} = \tilde{U}\check{B} + u\check{b}^H + R, \quad \text{where } R = \tilde{U}_2\tilde{B}_{21} + u\tilde{b}_1^H,$$

then

$$\|R\|_F \leq \|AZ - ZM\|_F.$$

If we set  $E = -R\tilde{U}^H$ , then  $(A+E)\tilde{U} = \tilde{U}\check{B} + u\check{b}^H$ . We may summarize these results in the following theorem.

**Theorem 2.8.** *Under the hypotheses of Theorem 2.7, write the deflated decomposition (2.26) in the form*

$$A\tilde{U} \cong \tilde{U}\check{B} + u\check{b}^H.$$

*Then there is an  $E$  satisfying*

$$\|E\|_F \leq \|AZ - ZM\|_F \tag{2.27}$$

*such that*

$$(A+E)\tilde{U} = \tilde{U}\check{B} + u\check{b}^H.$$

*Equality holds in (2.27) if and only if  $M$  is the Rayleigh quotient  $Z^H AZ = W^H BW$ .*

This theorem says that the deflation technique is backward stable. Deflating an approximate eigenspace with residual norm  $\rho$  corresponds to introducing an error of no more than  $\rho$  in  $A$ .

### Nonorthogonal bases

In practice we will seldom encounter a converging subspace unless it is a two-dimensional subspace corresponding to an imaginary eigenvalue in a real Schur decomposition. Instead we will be confronted with converged, normalized Ritz pairs  $(\mu_i, \hat{z}_i)$  ( $i = 1, \dots, p$ ) of one kind or another, and the vectors in these pairs cannot be guaranteed to be orthogonal. If we arrange the vectors in a matrix  $\hat{Z}$  and set  $\hat{M} = \text{diag}(\mu_1, \dots, \mu_p)$ , the residual  $\hat{R} = A\hat{Z} - \hat{Z}\hat{M}$  must be small because the individual residuals are small.

Since the deflation procedure only requires an orthonormal basis for the approximate eigenspace in question, we can compute a QR factorization

$$\hat{Z} = ZS \tag{2.28}$$

of  $\hat{Z}$  and use the orthonormal basis  $Z$  in the deflation procedure. Unfortunately, the residual for  $Z$  becomes

$$R = \hat{R}S^{-1} = A\hat{Z}S^{-1} - Z\hat{M}S^{-1} = AZ - ZM,$$

where  $M = S\hat{M}S^{-1}$ . If the columns of  $\hat{Z}$  are nearly dependent,  $\|S\|_2$  will be large, and the residual may be magnified — perhaps to the point where the deflation cannot be performed safely.

It may seem paradoxical that we could have, say, two vectors each of which we can deflate, but which taken together cannot be deflated. The resolution of this paradox is to remember that we are not deflating two vectors but the subspace spanned by them. If the vectors are nearly dependent, they must be very accurate to determine their common subspace accurately.

### Deflation in the Krylov–Schur method

The deflation procedure is not confined to eigenpairs calculated by a Rayleigh–Ritz procedure. For example, it can be used to deflate harmonic Ritz vectors or refined Ritz vectors. However, if Ritz vectors are the concern, there is an easy way to deflate Ritz vectors in the Krylov–Schur method. After a cycle of the algorithm, let the current decomposition have the form

$$A(U_1 \ U_1) = (U_1 \ U_1) \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} + u(0 \ b_2^H).$$

Here  $U_1$  represents a subspace that has already been deflated, and  $S_{22}$  is the Schur form that remains after the contraction phase.

To deflate the the Ritz pair corresponding to the  $(1, 1)$ -element of  $S_{22}$  we must set the first component of  $b_2$  to zero. Consequently, all we have to do to deflate is to verify that that component satisfies our deflation criterion.

If some other diagonal element of  $S_{22}$  is a candidate for deflation, we can exchange it into the  $(1, 1)$ -position of  $S_{22}$ , and test as above. This procedure can be repeated as often as desired. Algorithm 2.4 implements this scheme. If, as usual, we accumulate the transformations in an auxiliary matrix  $Q$  and apply them to  $U$  only at the end of the deflation process, this is an inexpensive algorithm.

### Deflation in an Arnoldi decomposition

Deflating an Arnoldi decomposition proceeds by reducing it to a Krylov–Schur decomposition. Specifically, at the end of the contraction phase the Arnoldi decomposition will have the form

$$A(U_1 \ U_1) = (U_1 \ U_1) \begin{pmatrix} B_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix} + u(0 \ b_2^H),$$

where  $H_{22}$  is upper Hessenberg. The procedure is to reduce  $H_{22}$  to Schur form, deflate as in the Krylov–Schur form, and use Algorithm 1.3 to bring the decomposition back to an Arnoldi form. Although the procedure is roundabout, if  $n$  is large, it is not expensive.

Let the Krylov–Schur  $AU = US + ub^H$  have the form

$$A(U_1 \ U_1) = (U_1 \ U_1) \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} + u(0 \ b_2^H),$$

where  $S_{11}$  is of order  $j$ . For some  $i > j$ , let the Ritz value  $s_{ii}$  be a candidate for deflation. This algorithm determines if the value can be deflated and if so produces a deflated decomposition.

1. Exchange  $S[i, i]$  into  $S[j+1, j+1]$ , accumulating the transformations
2. **if** ( $|b[j+1]| \leq$  the deflation tolerance)
3.      $j = j+1$
4.      $b[j+1] = 0$
5. **end if**

#### Algorithm 2.4: Deflating in a Krylov–Schur decomposition

---

#### Choice of tolerance

Although algorithms for dense eigenproblems are necessarily iterative, their implementations seldom use an explicit parameter to determine convergence. Instead they iterate until the limits set by finite precision arithmetic are reached. The reason is that it is comparatively inexpensive to do so.

Algorithms for large eigenproblems, on the other hand, usually allow the user to furnish a tolerance to decide convergence. The reason is that the computations are potentially so voluminous that it is unreasonable to continue them beyond the accuracy desired by the user. It is therefore necessary to say how such a criterion should be chosen.

The criterion (2.18) suggests stopping when the residual norm of the eigenpair in question is less than  $\text{tol} \cdot \alpha$ , where  $\alpha$  is some convenient estimate of the size of  $A$ . In that case, the eigenpair comes from a perturbed matrix with relative error  $\text{tol}$ . There are two possibilities for choosing  $\text{tol}$ .

1. Choose  $\text{tol}$  to be a modest multiple of the rounding unit. There is no point in choosing  $\text{tol}$  smaller, since the operations in the algorithms already impose a backward error of the same order (see Theorem 2.5).
2. Choose  $\text{tol}$  to be something that gives a backward error that one can live with. The advantage of this choice is that one may save calculations by not demanding too much. The difficulty is that the choice is highly problem dependent and must be left to the user.

Good packages, such as ARPACK, allow both a user-specified option and a default op-

tion based on the rounding unit.

Since deflation is backward stable (see Theorem 2.8), the tolerance for deflation should of the same order of magnitude as the tolerance for convergence. However, it should be made somewhat larger to allow multiple vectors to be deflated. See the discussion on page 339.

## 2.4. RATIONAL KRYLOV TRANSFORMATIONS

The principle use of shift-and-invert transformations in Arnoldi's method is to focus the algorithm on the eigenvalues near the shift  $\kappa$ . However, when the desired part of the spectrum is spread out, it may necessary to use more than one shift. One way of doing this is to restart with a new shift and a new vector. However, when the new shift is not far removed from the old, the Arnoldi decomposition may still contain useful information about the eigenvectors near the new shift—information that should be kept around. It is a surprising fact that we can change a Krylov decomposition from one in  $(A - \kappa_1 I)^{-1}$  to one in  $(A - \kappa_2 I)^{-1}$  without having to perform an inversion. This subsection is devoted to describing how this is done.

### Krylov sequences, inverses, and shifts

Although the computational algorithm is easy to derive, it is interesting to see what lies behind it. Suppose we have a Krylov sequence

$$u, (A - \kappa_1 I)^{-1}u, (A - \kappa_1 I)^{-2}u, \dots, (A - \kappa_1 I)^{1-k}u.$$

If we set  $v = (A - \kappa_1 I)^{1-k}u$ , then the sequence with its terms in reverse order is

$$v, (A - \kappa_1 I)v, \dots, (A - \kappa_1 I)^{k-1}v,$$

so that

$$\mathcal{K}_k[(A - \kappa_1 I)^{-1}, u] = \mathcal{K}_k(A - \kappa_1 I, v).$$

Moreover by the shift invariance of a Krylov sequence (Theorem 3.3, Chapter 4)

$$\mathcal{K}_k(A - \kappa_1 I, v) = \mathcal{K}_k(A - \kappa_2 I, v).$$

If we then set  $w = (A - \kappa_2 I)^{k-1}v$ , we have, as above,

$$\mathcal{K}_k(A - \kappa_2 I, v) = \mathcal{K}_k[(A - \kappa_2 I)^{-1}, w].$$

It follows that

$$\mathcal{K}_k[(A - \kappa_1 I)^{-1}, u] = \mathcal{K}_k[(A - \kappa_2 I)^{-1}, w]. \quad (2.29)$$

Equation (2.29) says that the Krylov subspace in  $(A - \kappa_1 I)^{-1}$  with starting vector  $u$  is exactly the same as the Krylov subspace in  $(A - \kappa_2 I)^{-1}$  with a different starting vector  $w$ . Thus the problem posed above amounts to restarting the Krylov sequence with the new vector  $w$ . We will now describe an algorithm for doing this.

### The rational Krylov method

Let us begin with an orthonormal Arnoldi decomposition

$$(A - \kappa_1 I)^{-1} U = \hat{U} \hat{H}.$$

Then

$$\begin{aligned} U &= (A - \kappa_1 I) \hat{U} \hat{H} \\ &= (A - \kappa_2 I) \hat{U} \hat{H} - (\kappa_1 - \kappa_2) \hat{U} \hat{H}. \end{aligned}$$

Hence

$$(A - \kappa_2 I)^{-1} \hat{U} [\hat{I} + (\kappa_1 - \kappa_2) \hat{H}] = \hat{U} \hat{H},$$

where

$$\hat{I} = (I \ 0)^T.$$

To reduce this relation to a Krylov decomposition, let

$$\hat{I} + (\kappa_1 - \kappa_2) \hat{H} = Q(\hat{I}R)$$

be a QR decomposition of  $\hat{I} + (\kappa_1 - \kappa_2) \hat{H}$ . Then

$$\begin{aligned} (A - \kappa_2 I)^{-1} (\hat{U} Q \hat{I}) &= \hat{U} \hat{H} R^{-1} \\ &= (\hat{U} Q)(Q^H \hat{H} R^{-1}). \end{aligned}$$

If we set  $\hat{V} = \hat{U} Q$ ,  $V = \hat{V} \hat{I}$ , and  $\hat{B} = Q^H \hat{H} R^{-1}$ , then

$$(A - \kappa_2 I)^{-1} V = \hat{V} \hat{B}.$$

This is a Krylov decomposition, which can be reduced to an Arnoldi decomposition by Algorithm 1.3.

Algorithm 2.5 implements this procedure. Here are some comments.

- The algorithm is quite economical involving no manipulations of  $A$ . As usual, the transformations to  $U$  can be saved and applied at the end of the algorithm.
- Since  $\hat{I} + (\kappa_1 - \kappa_2) \hat{H}$  is upper Hessenberg, the  $Q$  in the factorization in statement 1 can be computed as the product of  $k$  plane rotations. In particular, this structure makes it easy to preserve deflated vectors in the initial part of the decomposition.
- The algorithm is not unconditionally stable. If  $R$  is ill conditioned, the application of  $R^{-1}$  can magnify the residual error in the decompositions.

---

At first sight it might appear that the rational Krylov transformation is a good way of getting something for nothing. As one gets better approximations to an eigenvalue

Given an Arnoldi decomposition

$$(A - \kappa_1 I)^{-1} U = \hat{U} \hat{H}$$

of order  $k$  and a shift  $\kappa_2$ , this algorithm computes an Arnoldi decomposition

$$(A - \kappa_2 I)^{-1} W = \hat{W} \hat{H}.$$

1. Compute the QR decomposition  $\hat{I} + (\kappa_1 - \kappa_2) \hat{H} = Q(\hat{I}R)$ , where  $\hat{I} = (I \ 0)$
2.  $\hat{V} = \hat{U}Q$
3.  $V = \hat{V}\hat{I}$
4.  $\hat{B} = Q^H \hat{H} R^{-1}$
5. Use Algorithm 1.3 to reduce the decomposition  $(A - \kappa_2 I)^{-1} V = \hat{V} \hat{B}$  to an Arnoldi decomposition

### Algorithm 2.5: The rational Krylov transformation

---

of  $A$ , the argument goes, one can use them as shifts and get a sequence of Krylov decompositions with increasingly effective shift-and-invert enhancers. The flaw in this argument is that the rational Krylov transformation does not change the Krylov subspace. Consequently, the accuracy of eigenvector approximations is not affected. To put it another way, the closer the new shift is to an eigenvalue, the poorer the new starting vector.

Nor does the method save the trouble of computing a factorization. If one wants to continue the sequence, one must be able to apply  $(A - \kappa_2 I)^{-1}$  to a vector.

## 2.5. NOTES AND REFERENCES

### Arnoldi's method, restarting, and filter polynomials

Although the Arnoldi method is indeed due to Arnoldi [5], it was Saad [230, 1980] who first presented it as an effective algorithm for computing eigenpairs of large non-Hermitian matrices.

It is important to distinguish between restarting the Arnoldi process to compute a single eigenpair and to compute several eigenpairs simultaneously. The former problem is relatively easy: just restart with the best available approximation to the desired eigenvector. Once the eigenvector has been found it can be removed from the problem by various deflating techniques, which have been surveyed in [300, pp. 584–602] and [233, §IV.2]. Equivalent block methods for several eigenvalues exist, provided the number of eigenvalues is less than the block size. See [45] and [93].

The problem is far different when it is desired to restart the Arnoldi process while several eigenvectors are converging. To restart with a single approximate eigenvector would destroy the information accumulated about the others. In 1982, Saad [230] proposed restarting with a linear combination of approximate eigenvectors from the Krylov subspace, but in 1990 [233, p. 180] he pointed out that a good combination is hard to find. In an elegant paper Morgan [180, 1996] showed why.

The idea of following a sequence of Arnoldi steps with the application of a filter polynomial to purify the restart vector is due to Saad [232, 1984]. The problem with this procedure is that once again one must choose a vector to purify.

The breakthrough came with Sorensen's 1992 paper [246] entitled "Implicit Application of Polynomial Filters in a  $k$ -Step Arnoldi Method." In it he described the implicitly restarted Arnoldi algorithm as presented here, including exact and inexact shifts. (By the way, the "implicit" in implicit restarting does not refer to its use of implicit QR steps but to the fact that the starting vector is chosen implicitly by the shifts in contraction procedure.) He also gives convergence results for the case of a fixed filter polynomial and the case of exact shifts for a symmetric polynomial. (For other convergence results see [160].)

## ARPACK

New algorithms for large eigenproblems are usually implemented in MATLAB for testing but seldom end up as quality mathematical software. A happy exception is the implicitly restarted Arnoldi algorithm. Lehoucq, Sorensen, and Yang, building on a design of Phuong Vu, have produced ARPACK [162], a well-coded and well-documented package that bids fair to become the standard for large non-Hermitian eigenproblems.

### Reverse communication

All routines for solving large eigenvalue problems require that some operation be performed involving the matrix in question and a vector. The simplest example is the computation of  $Ax$ ; however, a more complicated operation may be required—e.g., for shift-and-invert enhancement. The problem for a general-purpose routine is how to get the caller to perform the operations in question.

One solution is for the caller to specify a subroutine to perform the operation in the calling sequence. However, once this subroutine is called, it must get data from the original program, which can only be done with messy global variable structures. An alternative—called *reverse communication*—is for the general-purpose routine to return to the calling program with a flag specifying the desired operation. The calling program, which knows its own data, performs the operation and returns the results via another call to the general-purpose routine.

### Exchanging eigenvalues

Stewart [256] produced the first program for exchanging eigenvalues in a real Schur form. The method is to iterate QR steps with an exact shift until the process converges.

The method is stable but can fail to converge when the separation of the eigenvalues in the blocks is poor. This lead to attempts to formulate noniterative algorithms. The algorithm described here is due to Bai and Demmel [11]. It too can fail, but the failures do not seem to occur in practice. It has been implemented in the LAPACK routine `xTREXC`.

### Forward instability

The forward instability of the QR step has been investigated by Parlett and Le [207] for symmetric tridiagonal matrices and by Watkins [289] for Hessenberg matrices. To deal with the problems it poses for the iteratively restarted Arnoldi method, Lehoucq and Sorensen [158] propose a procedure for modifying the Krylov subspace so that it remains orthogonal to the left eigenvector of a converged, unwanted Ritz pair.

### The Krylov–Schur method

The iteratively restarted Arnoldi procedure has two defects, both resulting from its commitment to maintaining a strict Arnoldi decomposition. First, an unwanted Ritz pair can fail to be purged from the decomposition. Second, it is awkward to deflate converged Ritz pairs. The introduction of general orthonormal Krylov decompositions, due to Stewart [267], solves both these problems by expanding the class of transformations that can be performed on the underlying decomposition. Since the method is centered around a Schur decomposition of the Rayleigh quotient, it is called the Krylov–Schur method.

Although the method is new, precursors of various aspects can be found in the literature. In fact, the contraction step of the iteratively restarted Arnoldi method produces an orthonormal Krylov decomposition, which is then truncated to become an Arnoldi decomposition. Regarding the use of Schur vectors, Fokkema, Sleijpen, and van der Vorst [74] explicitly use Schur vectors to restart the Jacobi–Davidson algorithm. Stathopoulos, Saad, and Wu [248] point out that since the unpreconditioned Jacobi–Davidson algorithm is equivalent to the Arnoldi algorithm, Schur vectors can also be used to restart the latter. Lehoucq [personal communication] has written code that uses Schur vectors in deflation. Also see [181]. Closer to home, for symmetric matrices Wu and Simon [306] exhibit what might be called a Lanczos-spectral decomposition, a special case of our Arnoldi–Schur decomposition. What distinguishes the Krylov–Schur approach is the explicit introduction and systematic manipulation of Krylov decompositions to derive and analyze new algorithms. The manipulations are justified by Theorem 1.8, which shows that they can never yield a decomposition that is not based on a Krylov sequence.

### Stability

Theorem 2.5 has not appeared in the literature. But it is easily proved. From the properties of the Gram–Schmidt algorithm with reorthogonalization [51, 117] and the general rounding-error analysis of unitary transformations [300, Ch. 3], it is easy to show

that the residual  $AU - UB - ub^H$  of the computed decomposition is of the order of the rounding error, and that the departure of  $U$  from orthonormality is of the same order. From this one constructs  $\tilde{U}$ , absorbs its error in the residual, and throws the error back on  $A$  as in Theorem 2.8, Chapter 4.

### Shift-and-invert enhancement and the Rayleigh quotient

Nour-Omid, Parlett, Ericsson, and Jensen [184] discuss the effect of accurate shifts on the Rayleigh quotient and conclude that the graded structure of the Rayleigh quotient makes it a nonproblem.

### Convergence

Convergence criteria for the Arnoldi processes are discussed in [161]. The analysis of convergence criteria for shift-and-invert enhancement is new.

### Deflation

Deflating converged Ritz values from an Arnoldi decomposition is a complicated affair [158], and it is not clear that the procedure can be extended to arbitrary vectors, such as refined and harmonic Ritz vectors. In fact, this is the problem that lead me to orthonormal Krylov decompositions. The fact that it may be impossible to deflate a group of several individually deflatable eigenvectors appears to be known but has not been analyzed before. The deflation technique for Arnoldi–Schur decompositions is closely related to Saad’s method of implicit deflation [233, §VI.2.3].

### The rational Krylov transformation

The rational Krylov method is due to Ruhe [221], who described it for the generalized eigenvalue problem.

## 3. THE LANCZOS ALGORITHM

We have seen that if  $A$  is symmetric and

$$AU_k = U_k T_k + \beta_k u_{k+1} e_k^T$$

is an Arnoldi decomposition, then the Rayleigh quotient  $T_k$  is a tridiagonal matrix of the form

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} \\ & & & & \beta_{k-1} & \alpha_k \end{pmatrix}. \quad (3.1)$$

This implies that the vectors  $u_k$  can be generated by a three-term recurrence of the form

$$\beta_j u_{j+1} = Au_j - \alpha_j u_j - \beta_{j-1} u_{j-1},$$

where

$$\alpha_j = u_j^H A u_j \quad \text{and} \quad \beta_j = \|Au_j - \alpha_j u_j - \beta_{j-1} u_{j-1}\|_2.$$

When we combine this recursion with the Rayleigh–Ritz method, we get the Lanczos method. The purpose of this section is to describe the practical implementations of this method.

Mathematically, the Lanczos vectors  $u_j$  computed by the Lanczos algorithm must be orthogonal. In practice, however, they can lose orthogonality, and the chief problem in implementing a Lanczos method is to maintain a satisfactory degree of orthogonality among them. There are two solutions. The first is to reorthogonalize the vectors at each step and restart when it becomes impossible to store the Lanczos vectors in main memory. The alternative is to save the Lanczos vectors in a backing store and move them into main memory to reorthogonalize the new Lanczos vectors—presumably only when absolutely necessary. In both cases the result of the orthogonalization is something that is not, strictly speaking, a Lanczos decomposition, since the reorthogonalization causes the Rayleigh quotient to become upper Hessenberg instead of tridiagonal. This fact is one of the main complications in dealing with the Lanczos algorithm in inexact arithmetic.

We will begin by describing a general form of the algorithm that encompasses most of the variants that have appeared in the literature. We will then treat the restarted Lanczos algorithm in which orthogonalization is performed at each step. In §3.3 we will consider semiorthogonal algorithms, in which the orthogonality of the Lanczos basis is allowed to deteriorate up to a point, after which orthogonalization is performed.

A key to deriving Lanczos-based algorithms is knowing when quantities can be neglected, i.e., when they are of the order of the rounding unit compared with  $A$ . To avoid cluttering our expressions with factors of  $\|A\|$ , we will assume throughout this section that:

$$A \text{ is a symmetric matrix with } \|A\|_2 = 1.$$

However, we reserve the right to insert a factor of  $\|A\|$  in our bounds whenever it seems appropriate.

### 3.1. REORTHOGONALIZATION AND THE LANCZOS ALGORITHM

#### Lanczos with reorthogonalization

In finite precision arithmetic the vectors  $u_k$  produced by the Lanczos process will deviate from orthogonality. As in the Arnoldi algorithm, we can reorthogonalize a straying Lanczos vector  $u_k$  against its predecessors. There are two problems associated with

Given a starting vector  $u_1$  this algorithm computes a sequence of suitably orthogonalized Lanczos vectors.

```

1. $U_1 = u_1$
2. for $k = 1, 2, \dots$
3. $v = A*u_k$
4. if ($k \neq 1$) $v = v - \beta_{k-1}*u_{k-1}$ fi
5. $\alpha_k = u_k^T * v$
6. $v = v - \alpha_k*u_k$
7. if ($k \neq 1$) $v = v - (u_{k-1}^T v)*u_{k-1}$ fi
8. $v = v - (u_k^T v)*u_k$
9. if (v is not sufficiently orthogonal)
10. Reorthogonalize
11. end if
12. $\beta_k = \|v\|_2$
13. $u_{k+1} = v/\beta_k$
14. $U_{k+1} = (U_k \ u_{k+1})$
15. end for k
```

### Algorithm 3.1: Lanczos procedure with orthogonalization

---

this procedure. First, the preceding vectors must be stored and then brought back into the fast memory of the machine. The actual storing of these vectors is not a problem, since we have to save them anyway to compute Ritz vectors. But if a reorthogonalization is performed at each Lanczos step, the time taken to transfer Lanczos vectors from the backing store will dominate the calculation. To avoid this problem, we can make two compromises. First, we can agree to tolerate a prescribed degree of nonorthogonality so that we do not have to reorthogonalize at every step. Second, we can reorthogonalize  $u_k$  only against those vectors to which it is actually nonorthogonal.

Algorithm 3.1 sketches a general procedure that encompasses most of the variants of the Lanczos algorithm. The following comments should be read carefully. They are a useful guide to what follows.

- Although we have written a nonterminating loop, it obviously will have to stop at some point—e.g., to restart or to deflate converged Ritz vectors.
- Depending on how the decision is made to reorthogonalize and the on details of the reorthogonalization process, this code encompasses five common variants of the Lanczos method.

1. No reorthogonalization:  $v$  is accepted as is at each iteration.
2. Full reorthogonalization:  $v$  is always reorthogonalized against all its predecessors.

3. Periodic reorthogonalization:  $v$  is reorthogonalized against all its predecessors, but only when it fails a test for orthogonality.
4. Partial reorthogonalization:  $v$  is reorthogonalized against a subset of its predecessors when it fails a test for orthogonality.
5. Selective reorthogonalization:  $v$  is reorthogonalized against a subspace of the span of its predecessors determined by the current Ritz vectors.

In the next two subsections we will treat full and periodic reorthogonalization. For the other procedures see the notes and references.

- It may happen that  $\beta_k$  becomes so small that it can be neglected. In that case  $U_k$  effectively spans an eigenspace of  $A$ . If eigenpairs outside this eigenspace are desired, the Krylov sequence must be restarted. In what follows we will tacitly assume that this is done whenever necessary.
- The loss of orthogonality in the Lanczos procedure is of two types: local and global. Statements 7–8 attempt to enforce *local orthogonality* of  $v$  against  $u_k$  and  $u_{k-1}$ . It turns out that the vector  $v$  in statement 6 can fail to be orthogonal to  $u_k$  and  $u_{k-1}$  to working accuracy. The cure is to reorthogonalize, which is done in statements 7–8. Note that the coefficients  $\alpha_k$  and  $\beta_{k-1}$  are not updated during this process. This is because in spite of the fact that reorthogonalization restores orthogonality, it does not affect the coefficients significantly: the loss of orthogonality corresponds to errors in  $\alpha_k$  and  $\beta_{k-1}$  that are below the rounding threshold.
- Statements 9–11 attempt to enforce *global orthogonality*—that is, the orthogonality of  $v$  against  $U_{k-1}$ . There are three problems here.

The first problem is to determine when global reorthogonalization is needed. We do this by keeping a running estimate of the inner products  $U_k^T U_k$  (see Algorithm 3.2).

The second problem is how to reorthogonalize. If we are performing full reorthogonalization, we simply have to reorthogonalize  $u_k$  against  $U_{k-1}$ . Other variants require that we also reorthogonalize  $u_{k-1}$  against  $U_{k-2}$ . Thus we use the term “reorthogonalize” in statement 10 in a general sense—to be specified in the context of the variant in question.

The third problem is that reorthogonalization contaminates the Lanczos decomposition. Specifically, the Rayleigh quotient becomes upper Hessenberg, so that instead of a Lanczos decomposition we have the relation

$$AU_k = U_k H_k + \beta_k u_{k+1} e_k^T, \quad (3.2)$$

where  $H_k$  is upper Hessenberg. This is not even an Arnoldi decomposition, since with periodic, partial, and selective reorthogonalization the matrix  $U_{k+1}$  is not required to be orthogonal to working accuracy. Nonetheless, there are effective work-arounds for this problem. See §3.3.

---

It will be useful to fix our nomenclature and notation at this point. The matrix  $H_k$  is a Rayleigh quotient in the general sense of Definition 2.7, Chapter 4. To distinguish

it from the ordinary Rayleigh quotient  $T_k$  we will call it the *adjusted Rayleigh quotient*. Note that it is truly a Rayleigh quotient in the sense of Definition 2.7, Chapter 4, since if  $((V \ v)^T)$  is a left inverse of  $(U_k \ u_{k+1})$ , then  $H_k = V^T A U_k$ .

In what follows the deviation of  $H_k$  from the tridiagonal matrix (3.1) generated by Algorithm 3.1 will be of paramount importance. We will denote this deviation by  $S_k$ , so that

$$H_k = T_k + S_k. \quad (3.3)$$

### 3.2. FULL REORTHOGONALIZATION AND RESTARTING

When the Lanczos algorithm is run in the presence of rounding error with full reorthogonalization the results are stable in the usual sense. This implies that the elements above the superdiagonal in the Rayleigh quotient are negligible and can be ignored. To see this, postmultiply the relation (2.13) in Theorem 2.5 by  $\tilde{U}_k$  to get

$$B_k = \tilde{U}_k A \tilde{U}_k + \tilde{U}_k E \tilde{U}_k,$$

where  $B_k$  is now the matrix  $H_k$  in (3.2). Hence  $H_k$  is a small perturbation of a symmetric matrix. Since the elements of  $H_k$  below the first subdiagonal are zero, the elements above the first superdiagonal must be small.

Thus the main difference between the Lanczos algorithm with full orthogonalization and the Arnoldi algorithm is that the Rayleigh quotient for the former is symmetric (to working accuracy). In this subsection we will briefly sketch the resulting simplifications. The reader is assumed to be familiar with the contents of §2.

#### Implicit restarting

In its larger details implicitly restarting a Lanczos decomposition is the same as restarting an Arnoldi decomposition (see §2.1). Given the Lanczos decomposition

$$AU_m = U_m T_m - \beta_m u_{m+1} e_m^T$$

and  $m-k$  shifts  $\kappa_1, \dots, \kappa_{m-k}$ , one applies  $m-k$  implicit QR steps with shifts  $\kappa_i$  to give the transformed decomposition

$$A(U_m Q) = (U_m Q)(Q^T T_m Q) + \beta_m u_{m+1} e_m^T Q \equiv \check{U}_m \check{T}_m - \beta_m u_{m+1} e_m^T Q,$$

where  $Q$  is the matrix of the accumulated QR transformations. If we set

$$\begin{aligned} \tilde{T}_k &= \check{T}_m[1:k, 1:k], \\ \tilde{U}_k &= U_m Q[:, 1:k], \\ \tilde{\beta}_k &= \|\check{T}_m[k+1, k] \check{U}[:, k+1] + \beta_k Q[k, m] u_{m+1}\|_2, \\ \tilde{u}_{k+1} &= \tilde{\beta}_k^{-1} (\check{T}_m[k+1, k] \check{U}[:, k+1] + \beta_k Q[k, m] u_{m+1}\|_2), \end{aligned}$$

then the restarted decomposition is

$$A \tilde{U}_k = \tilde{U}_k \tilde{T}_k + \tilde{\beta}_k \tilde{u}_{k+1} e_k^T.$$

The major difference between implicit restarts for the Lanczos and the Arnoldi decompositions is that implicit symmetric tridiagonal QR steps are used in the former (see Algorithm 1.3, Chapter 3). For large  $n$  this makes no difference in efficiency.

Both general and exact shifts can be used in restarting the Lanczos algorithm. But because the spectrum of  $A$  is real, one can be more sophisticated about choosing filter polynomials than with the Arnoldi method. For more see the notes and references.

### Krylov-spectral restarting

The comments on the forward instability of the QR algorithm for Hessenberg matrices (p. 325) apply to the tridiagonal Rayleigh quotient of the Lanczos algorithm. Consequently, when exact shifts are to be used in restarting, it will pay to use the analogue of the Krylov–Schur method. Since the Schur decomposition of a symmetric matrix is its spectral decomposition, we will call this analogue the *Krylov-spectral* method.

The chief difference between the Krylov–Schur and the Krylov-spectral methods is in the treatment of the Rayleigh quotient. The reduction to diagonal form is done by the symmetric tridiagonal QR algorithm. Since the Rayleigh quotient is diagonal, eigenvalues can be moved around by simple permutations — i.e., there is no need for special routines to exchange eigenvalues or eigenblocks.

### Convergence and deflation

If

$$AU = UM + ub^T, \quad M = \text{diag}(\mu_1, \dots, \mu_k),$$

is the current Krylov-spectral decomposition, then the primitive Ritz vectors are the unit vectors  $e_i$  ( $i = 1, \dots, m$ ). Consequently the residual norm for the  $i$ th Ritz vector is the magnitude of the  $i$ th component  $\beta_i$  of  $b$ . This number can be used as in §2.3 to determine the convergence of the  $i$ th Ritz pair. If this pair is moved to the beginning of the Rayleigh quotient, its component in  $b$  follows it. Consequently, if this component satisfies the criterion for deflation, it can be set to zero and the decomposition deflated.

A particularly satisfying aspect of the Krylov-spectral decomposition is that the deflation of Ritz pairs is uncoupled. The fact that one Ritz pair has been deflated does not affect the later deflation of other Ritz pairs. Contrast this with the discussion on page 339.

### 3.3. SEMIORTHOGONAL METHODS

We turn now to methods that allow the orthogonality of the Lanczos method to deteriorate. We cannot, of course, allow orthogonality to be completely lost. However, if we suitably restrict the loss we can compute accurate Ritz values and partially accurate Ritz vectors. A method that performs reorthogonalization only when there is danger of the loss becoming too great is called a *semiorthogonal method*. The first half of this subsection is devoted to the definition and consequences of semiorthogonality. The second shows how to maintain semiorthogonality by timely reorthogonalizations.

### Semiorthogonal bases

We begin with a definition.

**Definition 3.1.** Let  $U \in \mathbb{R}^{n \times k}$  have columns of 2-norm one, and let  $C = U^T U$ . Then  $U$  is SEMIORTHOGONAL if

$$|\gamma_{ij}| \leq \gamma_{\max} \leq \sqrt{\frac{\epsilon_M}{k}}, \quad i \neq j.$$

Three comments on this definition.

- Since the diagonal elements of  $C$  are one, the definition says that  $U$  is semiorthogonal if the elements of  $I - C$  are within roughly the square root of the rounding unit. Specifically, from the elementary properties of the 2-norm, we have

$$\|I - C\|_2 \leq \sqrt{\epsilon_M}.$$

If  $\|I - C\|_2$  were of the order of the rounding unit, we would say that  $U$  is orthogonal to working accuracy. Thus semiorthogonality requires orthogonality to only half working accuracy.

- Our definition of semiorthogonality is meant to be applied to the bases generated by Algorithm 3.1. Owing to rounding error, the members of these bases will not have norm exactly one. However, because they are normalized explicitly (statements 12 and 13), they will be normal to working accuracy. Since this small deviation from normality makes no difference in what follows, we will tacitly assume that all our bases are normalized.
- Choosing  $\gamma_{\max}$  to be about the square root of the rounding unit has the convenience that certain quantities in the Lanczos process will have negligible errors on the order of the rounding unit. However, there is nothing to keep us from using a larger value of  $\gamma_{\max}$  — say, to reduce the number of reorthogonalizations. The price we pay is that the errors that were formerly at the level of the rounding unit are now of the order  $\gamma_{\max}^2$ , and this number, not the rounding unit, will limit the attainable accuracy of our results.

### The QR factorization of a semiorthogonal basis

Let  $U$  be semiorthogonal and let

$$U = QR$$

be the QR factorization  $U$ . Then

$$C = U^T U = R^T R.$$

If we write  $R = I + W$ , where  $W$  is strictly upper triangular, then

$$C = (I + W)^T(I + W) \cong I + W^T + W, \tag{3.4}$$

where the approximation on the right is to working accuracy. It follows that up to terms of  $O(\epsilon_M)$  the matrix  $W$  is the strictly upper triangular part of  $C$  and hence that

$$\|R - I\|_2 = \|W\|_2 \lesssim \sqrt{\epsilon_M}. \quad (3.5)$$

In particular

$$\|R\|_2 \leq \|I\|_2 + \|W\|_2 = 1 + O(\sqrt{\epsilon_M}).$$

We will use these facts in the sequel.

### The size of the reorthogonalization coefficients

We have observed [see (3.2)] that reorthogonalization in the Lanczos algorithm introduces elements in the upper triangular part of the Rayleigh quotient, destroying its tridiagonal structure. For full reorthogonalization, these elements are insignificant and can be ignored. We will now show that if our basis is semiorthogonal, these elements may be too large to ignore.

Let us suppose that we must reorthogonalize at step  $k$ . Let  $\beta_k u_{k+1} = Au_k - \alpha_k u_k - \beta_{k-1} u_{k-1}$  be the relation obtained after the Lanczos step but before the reorthogonalization. Let  $U_k$  be semiorthogonal and let  $U_k = QR$  be its QR factorization. Let

$$u_{k+1} = \gamma q + \tau q_\perp,$$

where  $q \in \mathcal{R}(Q)$  and  $q_\perp \in \mathcal{R}(Q)^\perp$  have norm one. Then the vector  $d$  of reorthogonalization coefficients is

$$\begin{aligned} d &= \beta_k U_k^T u_{k+1} \\ &= \beta_k R^T Q^T (\gamma q + \tau q_\perp) \\ &= \beta_k \gamma R^T Q^T q. \end{aligned}$$

Since  $\|Q^T q\|_2 = 1$  and  $R^T \cong I$ , we have

$$\|d\|_2 \cong |\beta_k| |\gamma|. \quad (3.6)$$

To get some notion of what this approximation means, note that  $\gamma$  is the norm or the projection of  $u_{k+1}$  onto the column space of  $U_k$  and hence measures how  $u_{k+1}$  deviates from orthogonality to  $\mathcal{R}(U_k)$ . Now  $\gamma$  must be larger than  $\gamma_{\max}$ , since a loss of semiorthogonality is just what triggers a reorthogonalization. In the Lanczos process it is rare to encounter an extremely small  $\beta_k$ . Consequently, we can expect the elements of  $d$  to be on the order of  $\sqrt{\epsilon_M}$ . Since the elements of the difference  $S_k$  of the adjusted Rayleigh quotient  $H_k$  and  $T_k$  depend on the components of  $d$  [see (3.19)] we can expect it to be of the same order of magnitude.

### The effects on the Rayleigh quotient

We have observed [see (3.2)] that the Rayleigh quotient  $H_k = T_k + S_k$  is not the same as the tridiagonal matrix  $T_k$  computed by Algorithm 3.1. Nonetheless, it can be used to compute accurate Ritz values, as the following remarkable theorem shows. Its proof is beyond the scope of this volume.

**Theorem 3.2.** *Let*

$$AU_k = U_k(T_k + S_k) + \beta_{k+1}u_{k+1}\mathbf{e}_k^T \quad (3.7)$$

*be the decomposition computed by Algorithm 3.1 and assume that  $U_k$  is semiorthogonal. Let  $U_k = Q_kR_k$  be the QR factorization of  $U_k$ . Then*

$$Q_k^T A Q_k = T_k + O(\epsilon_M).$$

The theorem says that up to rounding error,  $T_k$  is a Rayleigh quotient associated with the space spanned by  $U_k$ . Thus we can expect the Ritz values from  $T_k$  to be as accurate as those we would obtain by full reorthogonalization.

### The effects on the Ritz vectors

Although the effects of semiorthogonality on the Rayleigh quotient and its Ritz values are benign, they have the potential to alter the corresponding Ritz vectors. The problem is that although a primitive Ritz vector  $w$  will be accurately determined by  $T_k$  (provided it is well conditioned) we cannot compute the corresponding Ritz vector  $Q_k w$ . Instead we must compute the vector  $U_k w$ . Since

$$\|U_k w - Q w\|_2 = \|(U_k - Q)w\|_2 = \|(Q R - Q)w\|_2 = \|(R - I)w\|_2,$$

by (3.5) we have

$$\|U_k w - Q w\|_2 \lesssim \sqrt{\epsilon_M}.$$

At first glance this result appears to be discouraging, since it says that the errors in the Ritz vector can be as large as  $\sqrt{\epsilon_M}$ . However, there are mitigating factors. Although  $w$  has 2-norm one, its elements decrease as the vector converges, so that most of its elements are much less than one. Moreover, only some of the elements of  $R - I$  are of order  $\sqrt{\epsilon_M}$ . Thus the product of  $(R - I)w$  may be considerably less than  $\sqrt{\epsilon_M}$ .

Perhaps more important is the effect of semiorthogonality on the residual norms, since it is these numbers that we use to decide on convergence. Now if  $(\mu, w)$  is a primitive Ritz pair, we see from (3.7) that

$$AU_k w - \mu U_k w = U_k S_k w + \beta_k \mathbf{e}_k^T w = U_{k+1} \begin{pmatrix} S_k \\ \beta_k \mathbf{e}_k^T \end{pmatrix} w.$$

Since  $U_{k+1}$  is semiorthogonal, it has a QR factorization  $Q R$  with  $R \cong I$ . Hence the residual norm is

$$\|AU_k w - \mu U_k w\|_2 = \left\| QR \begin{pmatrix} S_k \\ \beta_k e_k^T \end{pmatrix} w \right\|_2 \cong \left\| \begin{pmatrix} S_k \\ \beta_k e_k^T \end{pmatrix} w \right\|_2. \quad (3.8)$$

Thus the semilarge values in  $S_k$  can inflate the residual norm. However, since it is the product  $S_k w$  that determines the inflation, the comments about eigenvectors apply here also. The residual does not have to stagnate once it reaches  $\sqrt{\epsilon_M}$ .

Be that as it may, the above analysis shows clearly that we cannot rely on the usual estimate  $|\beta_k \omega_k|$  to decide on convergence, since it may be smaller than the right-hand side of (3.8). Instead one must carry along the matrix  $S_k$  and use it in computing residual norms. Another advantage of saving this matrix is that if a particular Ritz pair stops converging, we can use the inverse power method on  $H_k = T_k + S_k$  to correct the vector. Such a process is cheap, since  $H_k$  is tridiagonal with a few spikes above the diagonal in the columns corresponding to reorthogonalization steps.

Although it is not expensive to compute and store  $S_k$  (or equivalently  $H_k$ ), the formulas for the columns of  $S_k$  are not easy to derive. For the sake of continuity, we will leave this derivation to the end of the section and assume that  $S_k$  is available whenever we need it.

### Estimating the loss of orthogonality

Algorithms based on semiorthogonality allow the orthogonality to deteriorate until a vector  $u_{k+1}$  violates one of the semiorthogonality conditions

$$|u_j^T u_{k+1}| \leq \gamma_{\max}, \quad j = 1, \dots, k,$$

at which point  $u_{k+1}$  is reorthogonalized. Unfortunately, it is almost as expensive to compute the inner products  $u_j^T u_{k+1}$  as to perform a full reorthogonalization. We will now show how to estimate the loss of orthogonality as the Lanczos algorithm progresses.

Let  $U_k$  be the Lanczos basis computed by Algorithm 3.1 and let

$$C_k = U_k^H U_k.$$

Then the size of the off-diagonal elements of  $C_k$ , which are the inner products  $\gamma_{ij} = u_i^T u_j$ , measure the departure from orthogonality of  $u_i$  and  $u_j$ . It turns out that we can use the relation (3.7) to derive a recurrence for the  $\gamma_{ij}$ .

Suppose that we have computed  $C_k$  and we want to compute  $C_{k+1}$ . Write the  $j$ th column of (3.7) in the form

$$\beta_j u_{j+1} = A u_j - \alpha_j u_j - \beta_{j-1} u_{j-1} - U_j s_j + h_j, \quad (3.9)$$

where  $h_j$  is a quantity of order  $\|A\| \epsilon_M$  that accounts for the effects of rounding error in the relation. Also write

$$\beta_k u_{k+1} = A u_k - \alpha_k u_k - \beta_{k-1} u_{k-1} - U_k s_k + h_k. \quad (3.10)$$

Given the output of Algorithm 3.1, this algorithm computes estimates of the inner products  $\gamma_{k+1,j} = u_{k+1}^T u_j$ . It requires an estimate  $\alpha$  of  $\|A\|_2$ .

```

1. $\rho = 4*\alpha*\epsilon_M$
2. for $j = 1$ to $k-1$
3. $\gamma_{k+1,j} = \beta_j * \gamma_{k,j+1} - (\alpha_k - \alpha_j) * \gamma_{k,j} - \beta_{k-1} * \gamma_{k-1,j}$
4. if ($j \neq 1$)
5. $\gamma_{k+1,j} = \gamma_{k+1,j} + \beta_{j-1} * \gamma_{k,j-1}$
6. end if
7. if ($\gamma_{k+1,j} \geq 0$)
8. $\tau = \rho$
9. else
10. $\tau = -\rho$
11. end if
12. $\gamma_{k+1,j} = (\gamma_{k+1,k} + \tau) / \beta_k$
13. end for j
14. $\gamma_{k+1,k} = \epsilon_M$
```

### Algorithm 3.2: Estimation of $u_{k+1}^T u_j$

---

◊

Multiplying (3.9) by  $u_k^T$  and (3.10) by  $u_j^T$ , subtracting, and simplifying, we obtain

$$\begin{aligned} \beta_k \gamma_{k+1,j} &= \beta_j \gamma_{k,j+1} - (\alpha_k - \alpha_j) \gamma_{k,j} - \beta_{k-1} \gamma_{k-1,j} + \beta_{j-1} \gamma_{k,j-1} \\ &\quad - u_j^T U_k s_k + u_k^T U_j s_j - u_k^T h_k + u_j^T h_j. \end{aligned} \quad (3.11)$$

Letting  $j = 2, \dots, k-2$  we can compute all but the first and last elements of row  $k+1$  of  $C_k$ . The first element can be computed by leaving out the term  $\beta_{j-1} \gamma_{k,j-1}$  in (3.11). Because we maintain local orthogonality, the quantity  $\gamma_{k+1,k} = u_{k+1}^T u_k$  is on the order of the rounding unit.

Unfortunately, the quantities  $q_j^T h_j$  and  $q_k^T h_k$  are unknown. However, experience indicates that  $\|h_i\|_2 \leq \|A\|_2 \epsilon_M$ . Consequently, we can replace  $q_j^T h_j$  and  $q_k^T h_k$  with  $\alpha \epsilon_M$ , where  $\alpha$  is a suitable estimate of  $\|A\|_2$ . We adjust the sign of this replacement so that it always increases  $\gamma_{k+1,j}$ . The quantities  $u_k^T U_j s_j$  and  $u_j^T U_k s_k$  are in principle computable. But because we are maintaining semiorthogonality, they are also on the order of  $\alpha \epsilon_M$ . Consequently, we can absorb them into the corrections for  $q_j^T h_j$  and  $q_k^T h_k$ .

Algorithm 3.2 implements this scheme. Two comments.

- By taking absolute values in (3.11) we could obtain upper bounds on the loss of orthogonality. But these bounds grow too quickly to be useful: they will trigger re-orthogonalizations before they are needed. What we need are estimates of the  $\gamma_{ij}$  themselves. In exact arithmetic with accurate values of  $q_j^T h_k$ ,  $q_k^T h_j$ ,  $u_k^T U_k s_j$ , and

$u_j^T U_k s_k$ , the  $\gamma_{ij}$  would be exact. The substitution of upper bounds on these quantities turns the  $\gamma_{ij}$  into an estimate, driven upward by the choice of sign of  $\tau$ . Nonetheless, experiments show that the  $\gamma_{ij}$  calculated by this algorithm track the true values rather well. The reason seems to be that as the loss of orthogonality grows, the terms of order  $\alpha\epsilon_M$  has little effect on the recursion.

- In the basic recurrence, the row  $k+1$  of  $C$  depends only on rows  $k$ th and  $k-1$ . Moreover, we only need the row  $k+1$  to decide when to reorthogonalize  $u_{k+1}$ . Consequently, at any time we need only store two rows — the current row and the preceding row.
- In Algorithm 3.1 we do not test the vector  $u_{k+1}$  but its unnormalized counterpart  $v$ . This means we must replace  $\beta_k$  in the algorithm by  $\|v\|_2$ .

### Periodic reorthogonalization

The ability to estimate the deviation of the Lanczos vectors from orthogonality enables us to be precise about the test in statement 9 in Algorithm 3.1. Specifically, if for some  $j$  the estimate  $\gamma_{k+1,j}$  has a value greater than  $\gamma_{\max}$ , we must perform a reorthogonalization step. Because we have allowed our basis to deteriorate, the process is more complicated than a full reorthogonalization step. There are three differences.

1. We must reorthogonalize both  $u_k$  and  $v$  against  $U_{k-1}$ .
2. We must reorthogonalize  $v$  against  $u_k$ .
3. We may have to repeat the orthogonalization of  $v$  against  $U_k$ .

We will treat these items in turn.

The fact that we have to reorthogonalize both  $u_k$  and  $v$  follows from the three-term recurrence

$$\beta_{k+1} u_{k+2} = A u_{k+1} - \alpha_{k+1} u_{k+1} - \beta_k u_k.$$

If either  $u_k$  or  $u_{k+1}$  fails to be orthogonal to  $U_{k-1}$ , the lack of orthogonality will propagate to  $u_{k+2}$  and hence to the subsequent Lanczos vectors. This problem is also reflected by the presence of  $\gamma_{k,j-1}$  in the recursion (3.11) for the orthogonality estimates.

The process of reorthogonalizing  $v$  and  $u_k$  against  $U_{k-1}$  may affect the orthogonality of these two vectors. Hence the lost orthogonality must be restored.

The fact that we may have to repeat the reorthogonalization of  $v$  is a consequence of the fact that we are using a semiorthogonal basis to perform the reorthogonalization. To see what is going on, let  $U_k = QR$  be the QR factorization of  $U_k$ . Recall that we can write  $R = I + W$ , where to working accuracy  $W$  is the strictly upper triangular part of  $C_k = U_k^T U_k$  [see (3.4)]. Suppose we are trying to orthogonalize a vector of the form  $u = \gamma q + \tau q_\perp$ , where  $q \in \mathcal{R}(U_k)$ ,  $q_\perp \in \mathcal{R}(U_k)^\perp$ , and  $\|u\|_2 = \|q\|_2 = \|q_\perp\|_2 = 1$ . The orthogonalized vector is  $\tilde{u} = (I - U U^T)u$ . Hence the norm of the component of

$\tilde{u}$  lying in  $\mathcal{R}(U_k)$  is the norm of

$$\begin{aligned} Q^T \tilde{u} &= Q^T(I - QRR^TQ^T)(\gamma q + \tau q_{\perp}) \\ &= \gamma(I - RR^T)Q^Tq \\ &\cong -\gamma(W + W^T)Q^Tq \\ &\cong -\gamma(I - C_k)Q^Tq. \end{aligned} \tag{3.12}$$

Since  $\|I - C_k\|_2 \leq \sqrt{k}\gamma_{\max}$  and  $\|Q^T q\|_2 = 1$ , we have

$$\|Q^T \tilde{u}\|_2 \lesssim \sqrt{k}|\gamma|\gamma_{\max} \leq |\gamma|\sqrt{\epsilon_M}.$$

Now  $\|Q^T \tilde{u}\|_2$  is the size of the component of  $\tilde{u}$  in  $\mathcal{R}(U_k)$ , and  $|\gamma|$  is the size of the corresponding component of  $u$ . It follows that the reorthogonalization step will tend to reduce the component by a factor of  $\sqrt{\epsilon_M}$ . But we only perform a reorthogonalization if  $\gamma$  is near  $\sqrt{\epsilon_M}$  or larger. Thus, it is possible for one reorthogonalization to fail to purge the components along  $\mathcal{R}(U_k)$ . This is not a problem for  $u_k$ , since in this case by semiorthogonality  $\gamma$  is well below  $\sqrt{\epsilon_M}$ . But since  $v$  is not semiorthogonal, it may require another orthogonalization.

It is worth noting that although we perform reorthogonalizations in the Arnoldi method, our reasons for doing so are different. In the Arnoldi method, where the vectors are orthogonal to working accuracy, the danger is that cancellation will magnify otherwise negligible rounding errors, and the cure is to repeat the orthogonalization. In the semiorthogonal Lanczos method, the basis vectors are not orthogonal, and, as the above analysis shows, they have a limited capacity to reduce unwanted components—even when there is no rounding error involved.

When should we reorthogonalize? The size of the component of  $v$  in  $\mathcal{R}(U_k)$  is  $\|Q^T v\|_2$ . Hence if  $\|Q^T v\|_2 > \sqrt{\epsilon_M}$ , the first reorthogonalization step may have failed to eliminate the component. Now the vector  $v$  of reorthogonalization coefficients is  $x = U_k^T v = RQ^T U_{k+1}$ , and since  $R$  is near the identity matrix, we have  $\|x\|_2 \cong \|Q^T v\|$ . Thus we can use  $x/\|v\|_2$  to decide whether to reorthogonalize again.

Reorthogonalization also solves the problem of how to restart the recursion for the  $\gamma_{ij}$ . Since reorthogonalization leaves us with a  $u_k$  and  $u_{k+1}$  that are orthogonal to the columns of  $U_{k-1}$  to working accuracy, after the process the elements  $\gamma_{kj}$  and  $\gamma_{k+1,j}$  should be some multiple of the rounding unit—say  $\sqrt{n}\epsilon_M$ .

Algorithms 3.3 and 3.4 pull things together. Owing to limitations of the page size, we have been forced to separate the outer Lanczos loop from the reorthogonalization. But they are essentially one algorithm. Here are some comments.

- The reorthogonalization, which is done by the modified Gram–Schmidt algorithm, assumes that the previous Lanczos vectors are available. The details will depend on the size of the problem and the choice of backing store.
- Note that the algorithm does not renormalize  $u_k$ . The reason is that the reorthogonalization does not significantly change its norm. To see this, let [in the notation of

This algorithm performs Lanczos steps with periodic reorthogonalization. It is assumed the last two rows of the matrix  $C_k$  estimating  $U_{k+1}^T U_{k+1}$  are available. The reorthogonalization is done in Algorithm 3.4. The algorithm also updates the Rayleigh quotient  $H_k = T_k + S_k$  [see (3.7)].

```

1. for $k = 1, 2, \dots$
2. $v = A*u_k$
3. if ($k \neq 1$)
4. $v = v - \beta_{k-1}*u_{k-1}$
5. $H[k-1, k] = H[k, k-1] = \beta_{k-1}$
6. end if
7. $\alpha_k = u_k^T * v$
8. $H[k, k] = \alpha_k$
9. $v = v - \alpha_k*u_k$
10. if ($k \neq 1$) $v = v - (u_{k-1}^T v)*u_{k-1}$ fi
11. $v = v - (u_k^T v)*u_k$
12. if ($\max\{|\gamma_{k+1,j}|, j = 1, \dots, k\} > \gamma_{\max}$)
13. Orthogonalize u_k and v . Update H and C .
14. end if
15. $\beta_k = \|v\|_2$
16. $u_{k+1} = v/\beta_k$
17. end for i
```

**Algorithm 3.3:** Periodic orthogonalization I: Outer loop



This algorithm replaces statements 12–14 in Algorithm 3.3

```

1. if ($\max\{|\gamma_{k+1,j}|, j = 1, \dots, k\} > \gamma_{\max}$)
2. $\rho = \|v\|_2$
 Reorthogonalize u_k and v
3. for $j = 1$ to $k-1$
4. $w[j] = u_j^T * u_k$
5. $u_k = u_k - w[j]*u_j$
6. $x[j] = u_j^T * v$
7. $v = v - x[j]*u_j$
8. end for j
9. $\eta = u_k^T * v$
10. $v = v - \eta*u_k$

 If necessary orthogonalize v again
11. if $\|x\|_2 \geq \sqrt{n\epsilon_M}*\rho$
12. for $j = 1$ to $k-1$
13. $t = u_j^T * v$
14. $v = v - t*u_j$
15. $x[j] = x[j] + t$
16. end for j
17. $t = u_k^T * v$
18. $v = v - t*u_k$
19. $\eta = \eta + t$
20. end if

 Adjust H
21. $H[1:k-1, k-1] = H[1:k-1, k-1] + H(k-1, k)*w$
22. $H[1:k-1, k] = H[1:k-1, k] - H[1:k-1, 1:k-1]*w$
 $(w[k-1] - H[k, k])*w + x$
23. $H[k, k] = H[k, k] - \beta_{k-1}*w[k-1] + \eta$
24. end if
```

**Algorithm 3.4:** Periodic reorthogonalization II: Orthogonalization step

---

(3.12)]  $u_k = \gamma q + \tau q_{\perp}$ . Since  $\|u_k\|_2 = 1$ , we have  $\gamma^2 + \tau^2 = 1$ . By semiorthogonality,  $\gamma^2 \leq \epsilon_m$ . Hence  $\tau = 1$  to working accuracy. Since the reorthogonalized vector is  $\tau q_{\perp}$ , it has norm one to working accuracy.

- We have chosen to compute the Rayleigh quotient  $H_k$  instead of  $S_k$ . Since by definition  $S_k = H_k - T_k$ , we can obtain  $S_k$  by subtracting the  $\alpha_i$  and  $\beta_i$  from the diagonal and the super- and subdiagonals of  $H_k$ .
- Since reorthogonalization is an exceptional event, most columns of  $H_k$  will be zero above the first superdiagonal. Instead of allocating storage to retain the zero elements, one can store only the nonzero elements in a packed data structure. It is not at all clear that this is worthwhile, since the storage is no more than that required for a set of primitive Ritz vectors, which must be computed to test for convergence.

### Updating $H_k$

We must now derive the formulas for updating  $H_k$  after a reorthogonalization. Before the reorthogonalization we have the relation

$$AU_k = U_k H_k + v e_k^T, \quad (3.13)$$

which holds to working accuracy. The results of the reorthogonalization replace  $u_k$  with

$$\tilde{u}_k = u_k - U_{k-1} w \quad (3.14)$$

and  $v$  with

$$\tilde{v} = v - U_{k-1} x - \eta \tilde{u}_k. \quad (3.15)$$

Solving (3.14) and (3.15) for  $u_k$  and  $v$  and substituting the results in (3.13), we get

$$\begin{aligned} A(U_{k-1} \tilde{u}_k) + A(0 \ U_{k-1} w) &= (U_{k-1} \tilde{u}_k) H_k + (0 \ U_{k-1} w) H_k \\ &\quad + \tilde{v} e_k^T + (U_k x + \eta \tilde{u}_k) e_k^T. \end{aligned} \quad (3.16)$$

Now since  $AU_{k-1} = U H_{k-1} + \beta_{k-1} u_k e_k^T$ , we have from (3.14)

$$A(0 \ U_{k-1} w) = U_{k-1} (H_{k-1} + w e_{k-1}^T) w + \beta_{k-1} \tilde{u}_k e_{k-1}^T w. \quad (3.17)$$

Also

$$(0 \ U_{k-1} w) H_k = U_{k-1} w e_{k-1}^T H_k = h_{k-1,k} U_{k-1} w e_{k-1}^T + h_{kk} U_{k-1} w e_k^T. \quad (3.18)$$

If we substitute (3.17) and (3.18) into (3.16) and define  $\tilde{H}_k$  by

1.  $\tilde{H}[1:k-1, k-1] = H[1:k-1, k-1] + h_{k-1,k} w,$
2.  $\tilde{H}[1:k-1, k] = H[1:k-1, k] - H_{k-1} w - w_{k-1} w + h_{kk} w + x,$
3.  $\tilde{H}[k, k] = H[k, k] - \beta_{k-1} w[k-1] + \eta,$

then it is straightforward to verify that

$$A(U_{k-1} \tilde{u}_k) = (U_{k-1} \tilde{u}_k) \tilde{H}_k + \tilde{v} \mathbf{e}_k^T,$$

which is a Krylov decomposition involving the new vectors  $\tilde{u}_k$  and  $\tilde{v}$ . These adjustments have been implemented in statements 21–23 of Algorithm 3.4.

### Computing residual norms

A knowledge of  $H_k$  makes the computation of residual norms easy. Specifically, suppose that  $(\mu, U_k w)$  is an approximate eigenpair in the space spanned by  $U_k$ . We make no assumptions about how this pair was obtained—in most cases  $(\mu, w)$  will be an eigenpair of  $T_k$ , but the following analysis does not require it.

From the relation

$$AU_k = U_k H_k + \beta_k u_{k+1} \mathbf{e}_k^T,$$

we have that

$$AU_k w - \mu U_k w = U_k r + w_k \beta_k u_{k+1},$$

where

$$r = H_k w - \mu w.$$

It follows that the residual norm  $\rho = \|AU_k w - \mu U_k w\|_2$  satisfies

$$\begin{aligned} \rho^2 &= r^T U^T U r + \beta_k^2 w_k^2 + 2\beta_k w_k u_{k+1}^T U_k r \\ &= r^T (I + S) r + \beta_k^2 w_k^2 + 2\beta_k w_k s^T r, \end{aligned}$$

where by semiorthogonality

$$\|S\|_2, \|s\|_2 \leq O(\sqrt{\epsilon_M}).$$

Thus

$$|\rho^2 - \|r\|_2^2 - \beta_k^2 w_k^2| \leq \|r\|_2 |\beta_k w_k| O(\sqrt{\epsilon_M}).$$

Since  $\|r\|_2 |\beta_k w_k| \leq \|r\|_2^2 + \beta_k^2 w_k^2$ , we see that the quantity

$$\tilde{\rho} = \sqrt{\|r\|_2^2 + \beta_k^2 w_k^2} \tag{3.20}$$

estimates  $\rho$  to at least half the number of significant figures carried in the calculation. Thus a knowledge of the adjusted Rayleigh quotient enables us to compute residual norms without having to multiply by  $A$ .

If we do not require very accurate eigenvectors, we can avoid accumulating  $H_k$ . Specifically, the corrections in (3.19) consist of coefficients from the reorthogonalization step, which by semiorthogonality are bounded by  $O(\sqrt{\epsilon_M})$ . Hence the difference between the adjusted Rayleigh quotient  $H_k$  and the Lanczos matrix  $T_k$  is also  $O(\sqrt{\epsilon_M})$ . Consequently, until the residual falls below  $\sqrt{\epsilon_M}$ , it does not matter if we use  $T_k$  or  $H_k$  to estimate its norm. But for smaller residuals, we should use  $H_k$ .

### An example

We conclude this subsection with an example that illustrates the behavior of periodic reorthogonalization in the Lanczos algorithm.

**Example 3.3.** In this example  $A$  is a diagonal matrix of order  $n = 500$  whose first eigenvalue  $\lambda_1 = 1$  and whose other eigenvalues are given by the recurrence

$$\lambda_i = \frac{\lambda_{i-1}}{1 + 1/i^2}, \quad i = 2, \dots, n.$$

The periodically reorthogonalized Lanczos algorithm was run on this matrix with a random starting vector and  $\gamma_{\max} = \sqrt{10^{-16}/n}$ . Orthogonalizations took place for  $k = 11, 17, 22, 28, 33, 38$ . Various statistics are summarized in the following table, in which we are concerned with the approximate eigenpairs  $(\mu_5^{(k)}, U_k w_5^{(k)})$ , where the pair  $(\mu_5^{(k)}, w_5^{(k)})$  is the fifth eigenpair of the Lanczos matrix  $T_k$ .

| $k$ | c1      | c2      | c3      | c4      | c5      |
|-----|---------|---------|---------|---------|---------|
| 10  | 3.9e-15 | 3.9e-15 | 1.9e-02 | 1.9e-02 | 1.9e-02 |
| 20  | 5.3e-15 | 3.9e-15 | 2.5e-07 | 2.5e-07 | 2.5e-07 |
| 30  | 6.5e-15 | 3.9e-15 | 2.8e-16 | 1.4e-12 | 1.4e-12 |
| 40  | 7.0e-15 | 3.9e-15 | 8.9e-19 | 1.4e-12 | 1.4e-12 |

The columns of the table contain the following quantities.

c1:  $\|T_k - Q^T A Q\|_2$ , where  $Q$  is the  $Q$ -factor of  $U_k$ .

c2: The global residual norm

$$\|AU_k - U_k H_k - \beta_k u_{k+1} e_k^T\|_2.$$

c3: The classical estimate  $|\beta_k e_k^T w_5^{(k)}|$  for the residual norm.

c4: The true residual norm.

c5: The residual norm estimated by (3.20).

The numbers in the first column illustrate the results of Theorem 3.2. The tridiagonal Lanczos matrix  $T_k$  is to working accuracy the Rayleigh quotient formed from the  $Q$ -factor of  $U_k$ . This means that in spite of the contamination of  $T_k$  by the reorthogonalization process, its eigenvalues converge to eigenvalues of  $A$ .

The second column shows that the corrections in Algorithm 3.4 produce a Krylov decomposition whose residual is of the order of the rounding unit.

The third and fourth columns show that the classic estimate for the residual norm deviates from the true residual norm. In particular the latter stagnates at about  $10^{-12}$ , whereas the former continues to decrease. But the two values diverge only after the true residual norm falls below  $\sqrt{\epsilon_M}$ , as noted above. The fifth column shows that the approximate residual norm (3.20) is a reliable estimate of the residual norm.

Perhaps the most important point of this example is that Ritz vectors computed from the matrix  $T_k$  can fail to converge. When this happens, there is no cure except to compute Ritz vectors from the adjusted Rayleigh quotient  $H_k$ . Fortunately, the simple form of  $H_k$  makes it easy to apply a step of the inverse power method to the eigenvectors of  $T_k$  to obtain the primitive Ritz vectors of  $H_k$ .

### 3.4. NOTES AND REFERENCES

#### **The Lanczos algorithm**

As was noted in §3.4, Chapter 4, Lanczos knew that his method was in some sense iterative, since his vectors could be combined to produce good approximations to eigenvectors before the algorithm ran to completion. But for some time the method was chiefly regarded as an exact method for tridiagonalizing a symmetric matrix. For example, it is so viewed in an early paper of Wilkinson [296, 1959], which, incidentally, was the first to come to grips with the need for reorthogonalization. The acceptance of the Lanczos method as an iterative method is due to Kaniel [145, 1966] and especially Paige, who in his thesis [196, 1971] and other papers [195, 197, 198, 199] laid the foundation for much of the work that was to follow.

#### **Full reorthogonalization**

The need for full reorthogonalization has been generally cited as a defect of the Lanczos method, and a good deal of research has been devoted to finding a way around this problem. Only when Sorensen showed how to restart the Arnoldi process (see §2) did the fully reorthogonalized method become a reasonable option. For an error analysis of full reorthogonalization, see [195].

Krylov-spectral restarting follows from our general treatment of Krylov decompositions. However, Wu and Simon [306] introduce it impromptu under the name of thick restarting.

#### **Filter polynomials**

For Arnoldi's method exact shifts are generally used in restarting because the alternatives are cumbersome. In the Lanczos method, where the eigenvalues in question lie on the real line, inexact shifts are a real possibility. For example, one could use the zeros of a Chebyshev polynomial to suppress the spectrum on a given interval. Another possibility is to use Leja points or their approximations. For more on this topic, see [6] and [7].

#### **Semiorthogonal methods**

The semiorthogonal variants of Lanczos's algorithm all appeared between 1979 and 1984. Selective reorthogonalization is due to Parlett and Scott [209]; periodic reorthogonalization to Grcar [97]; and partial reorthogonalization to Simon [238]. Special mention must be made of an elegant paper by Simon [237], which unifies these methods.

Periodic reorthogonalization is the most straightforward of the methods. Theorem 3.2 is due to Simon [237], who shows that it applies to any basis generated by a semiorthogonal method. The recurrence (3.11) is implicit in Paige's thesis [196]; however, its explicit formulation and computational use are due to Simon [237, 238], who also gives empirical grounds for believing that the quantities calculated in Algorithm 3.2 actually track the loss of orthogonality. For another recurrence see [143]. The fact that semiorthogonality can cause the convergence of Ritz vectors to stagnate does not appear to have been noticed before (although in the context of linear systems the possible untoward effects are hinted at in [237] and [205]). The formulas (3.19) are new. A package LANSO implementing periodic reorthogonalization is available from Netlib.

Partial reorthogonalization is based on the observation that one only needs to reorthogonalize  $v$  against those vectors  $u_i$  for which the bounds computed by Algorithm 3.2 are above  $\gamma_{\max}$ . Although the idea is simple, the implementation is complicated, and the reader is referred to the treatment of Simon [238]. A package LANZ implementing partial reorthogonalization is available from Netlib.

Selective reorthogonalization is based on a celebrated result of Paige [196]. Let  $z_j$  ( $j = 1, \dots, k$ ) be the Ritz vectors computed at step  $k$  of the Lanczos process and let  $w_j$  be the corresponding primitive Ritz vectors. Then there are constants  $\gamma_j$  of order of the rounding unit such that

$$z_j^T u_{k+1} = \frac{\gamma_j}{\beta_k e_k^T w_j}.$$

Now by (3.20),  $|\beta_k e_k^T w_j|$  is the residual norm for the Ritz vector [remember that for Ritz vectors  $r$  in (3.20) is zero]. It follows that  $u_{k+1}$  can have large components in  $\mathcal{R}(U_k)$  along only Ritz vectors that have almost converged. Broadly speaking, selective reorthogonalization consists of saving converged and nearly converged Ritz vectors and orthogonalizing the Lanczos vectors against them. But as is usual with Lanczos methods, the devil is in the details, and the reader should look for his presence in the above references.

### No reorthogonalization

It is possible to use the Lanczos algorithm to good effect with no reorthogonalization. What happens is that after a loss of orthogonality previously converged Ritz values may reappear and spurious Ritz values may be generated. Cullum and Willoughby [46, 48] have shown how to detect the spurious Ritz values. A package LANCZOS is available from Netlib.

### The singular value decomposition

The singular values and vectors  $(\sigma_i, u_i, v_i)$  of a matrix  $X$  can be calculated by applying the Lanczos algorithm to the cross-product matrix  $X^T X$ . This works well for the dominant singular values, but as the analysis in §3.2, Chapter 3, shows it may fail to

give sufficiently accurate results for the smaller singular values and their vectors. An alternative is to apply the method to the matrix

$$\begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$$

whose eigenpairs are

$$\left[ \pm\sigma_k, \begin{pmatrix} v \\ \pm u \end{pmatrix} \right].$$

However, convergence to the smaller eigenvalues can be expected to be slow, since they are interior eigenvalues.

For more see [21, 47, 89]. Software is available in SVDPACK at Netlib.

### Block Lanczos algorithms

In §3.3, Chapter 4, we pointed out that a Krylov sequence cannot see more than one copy of a multiple eigenvalue and indicated that one way around the problem is to work with block Krylov sequences. It turns out that if one orthogonalizes a block Krylov sequence one gets a block tridiagonal Rayleigh quotient. The resulting generalization of the Lanczos algorithm is called a *block Lanczos algorithm*. Such algorithms have been proposed by Cullum and Donath [45], Golub and Underwood [92, 280], and Ruhe [219]. For the use of shift-and-invert enhancement in a block Lanczos algorithm see the paper of Grimes, Lewis, and Simon [98], which is particularly strong on implementation issues.

### The bi-Lanczos algorithm

The original Lanczos algorithm [157] was actually a method for general non-Hermitian matrices. The idea was to generate two Krylov sequences, one in  $A$  and the other in  $A^T$ . If we denote bases for these sequences by  $U_k$  and  $V_k$  and require that they be biorthogonal in the sense that  $V_k^H U_k = I$ , then the Rayleigh quotient  $T_k = V_k^H A V_k$  is tridiagonal, and it follows that the columns of  $U_k$  and  $V_k$  satisfy three-term recurrence relations. When  $A$  is symmetric and  $u_1 = v_1$ , this reduces to the usual Lanczos method.

Gutknecht [103, 104] gives a full treatment of the theory of the bi-Lanczos method. The method shares with the Lanczos method the problem of loss of orthogonality (or, strictly speaking, biorthogonality). But it can fail in other, more serious ways. The cure is a look-ahead Lanczos recurrence, proposed by Parlett, Taylor, and Liu [210], that gets around the failure by allowing the Rayleigh quotient to deviate locally from tridiagonality. For more see [79, 105, 275].

## 4. THE GENERALIZED EIGENVALUE PROBLEM

In this section we will consider the use of Krylov methods to solve the generalized eigenvalue problem  $Ax = \lambda Bx$ . The basic idea is to transform the generalized ei-

genvalue problem into an ordinary eigenvalue problem and then apply some variant of the Arnoldi or Lanczos method. An important issue here is the nature of the transformation. A second issue is how to determine convergence, since the usual residual criterion concerns the transformed problem, not the original. In §4.1 we will discuss these issues. In many applications the matrix  $B$  is positive definite. In §4.2 we will show how we can replace orthogonality in the Arnoldi and Lanczos algorithms with  $B$ -orthogonality to obtain new algorithms that are specially tailored to this form of the generalized eigenproblem.

## 4.1. TRANSFORMATION AND CONVERGENCE

### Transformation to an ordinary eigenproblem

The first step in applying Krylov sequence methods to a generalized eigenproblem  $Ax = \lambda Bx$  is to find an equivalent eigenproblem  $Cx = \mu x$  with the same right eigenvectors. Unfortunately, all transformations require that a linear system be solved to compute the products  $Cu$  required by Krylov methods. For example, if  $B$  is non-singular we can take  $C = B^{-1}A$ . The product  $v = Cu$  can then be computed as follows.

1.  $w = Au$
2. Solve the system  $Bv = w$

To solve the linear system, we need to factor a matrix —  $B$  in the above example. (The alternative is to use an iterative method; but that is another story.) When the matrix in question is easy to factor, this presents no problems. But when factorizations are expensive, we need to make each one count. As it turns out, if we define  $C$  properly, we can transform to an ordinary eigenproblem and get the benefits of shift-and-invert enhancement for the cost of a single factorization.

To see how this is done let  $\kappa$  be a shift, and write the problem in the form

$$(A - \kappa B)x = (\lambda - \kappa)Bx.$$

Then

$$(A - \kappa B)^{-1}Bx = (\lambda - \kappa)^{-1}x.$$

Hence if we set

$$C = (A - \kappa B)^{-1}B \quad \text{and} \quad \mu = (\lambda - \kappa)^{-1},$$

then

$$Cx = \mu x.$$

Because the transformation from  $\lambda$  to  $\mu$  is the same as for the ordinary shift-and-invert method, we will call  $C$  the *generalized shift-and-invert transform*.

The eigenvalues of the pencil  $(A, B)$  that are near  $\kappa$  become large, so that they will tend to be found first by Krylov methods. If the matrix  $A - \kappa B$  can be factored, the product  $v = Cu$  can be efficiently computed by the following algorithm.

1.  $w = Bu$
2. Solve the system  $(A - \kappa B)v = w$

Thus the generalized shift-and-invert transformation requires only a single factorization.

### Residuals and backward error

We turn now to the problem of convergence criteria. We will proceed in two stages. First we will establish a backward perturbation theorem for the generalized eigenproblem  $Ax = \lambda Bx$  in terms of residual  $r = Ax - \lambda Bx$ . We will then show how to use the residual  $Cx - \lambda x$  to bound the norm of  $r$ .

The backward error theorem is an analogue of Theorem 1.3, Chapter 2.

**Theorem 4.1.** *Let  $(\lambda, x)$  be an approximate eigenpair of the pencil  $A - \lambda B$  and let*

$$r = Ax - \lambda Bx.$$

*Let  $\|\cdot\|$  denote the Frobenius or 2-norm, and let*

$$E = \frac{\|A\|}{(\|A\| + |\lambda| \|B\|) \|x\|} rx^H \quad \text{and} \quad F = \text{sign}(\lambda) \frac{\|B\|}{(\|A\| + |\lambda| \|B\|) \|x\|} rx^H,$$

*where*

$$\text{sign}(\lambda) = \begin{cases} \bar{\lambda}/|\lambda|, & \lambda \neq 0, \\ 0, & \lambda = 0. \end{cases}$$

*Then*

$$\frac{\|E\|}{\|A\|}, \frac{\|F\|}{\|B\|} = \frac{\|r\|}{(\|A\| + |\lambda| \|B\|) \|x\|}$$

*and*

$$(A + E)x = \lambda(B + F)x.$$

**Proof.** Direct verification. ■

The bounds are optimal in the sense that there are no matrices  $\hat{E}$  and  $\hat{F}$  satisfying  $(A + E)x = \lambda(B + F)x$  for which

$$\frac{\|\hat{E}\|}{\|A\|}, \frac{\|\hat{F}\|}{\|B\|} < \frac{\|r\|}{(\|A\| + |\lambda| \|B\|) \|x\|},$$

where at least one of the inequalities is strict.

Since the residual is in general computable, we can use this theorem to determine convergence. Namely, stop when the backward error is sufficiently small — say some multiple of the rounding unit. For more on the use of backward error to determine convergence, see the comments following Theorem 1.3, Chapter 2.

### Bounding the residual

When we use, say, the Arnoldi method with a generalized shift-and-invert transformation  $C = (A - \kappa B)^{-1}B$ , the residuals we calculate are of the form  $s = Cx - \mu x$ . What we really want is a bound on the norm of the residual  $r = Ax - \lambda Bx$ , where  $\mu = (\lambda - \kappa)^{-1}$ . To get such a bound multiply  $s$  by  $A - \kappa B$  to get

$$(A - \kappa B)s = Bx - \mu(A - \kappa B)x.$$

Dividing by  $\mu$ , and simplifying, we get

$$\mu^{-1}(A - \kappa B)s = -r.$$

Hence

$$\|r\| \leq |\mu^{-1}|(\|A\| + |\kappa|\|B\|)\|s\|.$$

It follows from Theorem 4.1 that  $(\lambda, x)$  is an exact eigenpair of pair  $(\tilde{A}, \tilde{B})$ , where the normwise relative error in  $\tilde{A}$  and  $\tilde{B}$  is bounded by

$$\rho = |\mu|^{-1} \frac{\|A\| + |\kappa|\|B\|}{\|A\| + |\lambda|\|B\|} \frac{\|s\|}{\|x\|}.$$

Consequently, if  $\rho$  is less than some tolerance, we can declare the pair  $(\lambda, x)$  converged.

This criterion has the drawback that we need to know the norms of  $A$  and  $B$ , which may not be available. However, in practice we will be looking for eigenvalues near the shift  $\kappa$ . Hence, we will have  $\|A\| + |\kappa|\|B\| \cong \|A\| + |\lambda|\|B\|$ . Consequently, we may take

$$\hat{\rho} = |\mu|^{-1} \frac{\|s\|}{\|x\|} \tag{4.1}$$

as an approximate bound on the backward error.

### 4.2. POSITIVE DEFINITE $B$

A difficulty with applying the generalized shift-and-invert transformation to the S/PD pencil  $(A, B)$  is that the matrix  $C = (A - \kappa B)^{-1}B$  is no longer symmetric and it appears we are forced to use the Arnoldi method rather than the more efficient Lanczos method to solve it. We could, of course, compute a Cholesky factorization  $R^T R$  of  $B$  and apply the Lanczos algorithm to the symmetric matrix  $R(A - \kappa B)^{-1}R^T$ , which is similar to  $C$ . Unfortunately, this procedure requires two factorizations — one of  $B$  and the other of  $A - \kappa B$ . Fortunately, we can restore symmetry of a sort by working with a different inner product generated by  $B$ .

We will begin with a brief treatment of the  $B$  inner product. We will then introduce the Arnoldi and Lanczos algorithms with respect to the  $B$  inner product. Finally, we will indicate some of the things that can happen when  $B$  is singular or ill conditioned.

### The $B$ inner product

We begin with some definitions.

**Definition 4.2.** Let  $B$  be a positive definite matrix. Then the  $B$  INNER PRODUCT is defined by

$$(x, y)_B = y^T B x. \quad (4.2)$$

The  $B$ -NORM is

$$\|x\|_B = \sqrt{(x, x)_B} = \sqrt{x^T B x}.$$

A matrix  $C$  is  $B$ -SYMMETRIC if for all  $x$  and  $y$ ,

$$(Cx, y)_B = (x, Cy)_B.$$

Here are three comments on this definition.

- The reader should prove that the  $B$  inner product is actually an inner product — i.e., that it is a symmetric, definite, bilinear function — and that the  $B$ -norm is actually a norm. The latter proof is essentially the same as for the 2-norm and proceeds via the  $B$ -Cauchy inequality

$$|(x, y)_B| \leq \|x\|_B \|y\|_B.$$

- There is a curious inversion of the order of  $x$  and  $y$  as one passes from the left- to the right-hand side of (4.2). This makes no difference when we are working over the field of real numbers. In the complex case, however, the order matters. Specifically, if we define

$$(x, y)_B = y^H B x,$$

then  $(\mu x, y)_B = \mu(x, y)_B$  and  $(x, \mu y)_B = \bar{\mu}(x, y)_B$ , which is how functional analysts expect their inner products to behave. The alternate definition —  $(x, y)_B = x^H B y$  — would give  $(\mu x, y)_B = \bar{\mu}(x, y)_B$ , which is not standard.

- It is easily verified that:

*The matrix  $C$  is  $B$  symmetric if and only if  $BC$  is symmetric.*

In particular, if  $A$  is symmetric, then the generalized shift-and-invert transform  $(A - \kappa B)^{-1} B$  is  $B$ -symmetric.

## Orthogonality

It is natural to say that two vectors  $u$  and  $v$  are  $B$ -orthogonal if  $(u, v)_B = 0$ . All of the usual properties and algorithms associated with orthogonality carry over to  $B$ -orthogonality.

We will say that  $U$  is  $B$ -orthonormal if  $U^TBU = I$ . If  $U$  is square it is  $B$ -orthogonal. The following easily proved result will be important later.

If  $U$  is  $B$ -orthonormal and  $Q$  is orthogonal, then  $UQ$  is  $B$ -orthonormal. (4.3)

Thus postmultiplication by an orthogonal matrix (but not a  $B$ -orthogonal matrix) preserves  $B$ -orthonormality.

## $B$ -Orthogonalization

In what follows we will need to  $B$ -orthogonalize a vector  $v$  against a set  $u_1, \dots, u_k$  of  $B$ -orthonormal vectors. This can be done by a variant of the Gram–Schmidt algorithm in which the  $B$  inner product replaces the ordinary inner product. However, notice that the notation  $(x, y)_B$  conceals a multiplication of a vector by  $B$ . Unless we are careful, we can perform unnecessary multiplications.

To see this, consider the following  $B$  variant of the modified Gram–Schmidt algorithm.

1. **for**  $j = 1$  **to**  $k$
2.      $h_j = (u_j, v)_B$
3.      $v = v - h_j * u_j$
4. **end for**  $j$

When we recast this algorithm in matrix terms we get the following.

1. **for**  $j = 1$  **to**  $k$
2.      $w_j = B * u_j$
3.      $h_j = w_j^T * v$
4.      $v = v - h_j * u_j$
5. **end for**  $j$

This program requires  $k$  multiplications by  $B$ .

On the other hand, consider the  $B$  variant of the classical Gram–Schmidt algorithm.

1. **for**  $j = 1$  **to**  $k$
2.      $h_j = (u_j, v)_B$
3. **end for**  $j$
4. **for**  $j = 1$  **to**  $k$
5.      $v = v - h_j * u_j$
6. **end for**  $j$

In matrix terms this algorithm becomes

1.  $w = B*v$
  2. **for**  $j = 1$  **to**  $k$
  3.      $h_j = u_j^T * w$
  4. **end for**  $j$
  5. **for**  $j = 1$  **to**  $k$
  6.      $v = v - h_j * u_j$
  7. **end for**  $j$
- (4.4)

This version requires only one multiplication by  $B$ .

In large problems multiplications by  $B$  may dominate the computation. Hence it is desirable to use the classical Gram–Schmidt algorithm whenever possible. Now it is well known that the modified Gram–Schmidt algorithm is numerically superior to the classical Gram–Schmidt algorithm. There are two ways around this difficulty.

1. Reorthogonalize. This puts the modified and classical Gram–Schmidt algorithms on an equal footing.
  2. If, as in Krylov sequence methods, we will be repeatedly orthogonalizing against the same vectors  $u_j$ , we can compute  $p_j = Bu_j$  as  $u_j$  is generated and use it in the modified Gram–Schmidt algorithm. This process trades off storage for the superior properties of the modified Gram–Schmidt algorithm.
- (4.5)

### The $B$ -Arnoldi method

The  $B$ -Arnoldi method generates an Arnoldi decomposition of the form

$$CU_k = U_k H_k + \beta_k u_{k+1} e_k^T, \quad (4.6)$$

where  $C$  is the generalized shift-and-invert transform and  $U_k$  is  $B$ -orthonormal. The procedure amounts to substituting  $B$ -orthogonalization for ordinary orthogonalization. Here are some details.

- **Orthogonalization.** Because reorthogonalization is always used in the Arnoldi process, we can use the classical Gram–Schmidt algorithm, as suggested in (4.5), to save multiplications by  $B$ .
- **Ritz pairs and convergence.** Ritz pairs may be computed from the Rayleigh quotient  $H_k = U_k^T B C U_k$ . Specifically, if  $Cz = \nu z$ , where  $z = U_k w$ , then from (4.6)

$$\nu U_k w = U_k H_k w + \beta_k \omega_k u_{k+1},$$

where  $\omega_k$  is the last component of  $w$ . It follows on premultiplication by  $U_k^T B$  that  $H_k w = \nu w$ , so that we can find the primitive Ritz vector  $v$  as an eigenvector of  $H_k$ .

On the other hand, if  $H w = \nu w$ , then with  $z = U w$ , we have from (4.6)

$$s \equiv Cz - \nu z = \beta_k \omega_k u_{k+1}.$$

Hence the residual 2-norm of the Ritz pair  $(\nu, U_k w)$  is

$$\|s\|_2 = |\beta_k| |\omega_k| \|u_{k+1}\|_2. \quad (4.7)$$

Thus, as usual, we can determine convergence of Ritz pairs directly from the decomposition and the primitive Ritz vector without having to compute the residual vector. Note that (4.7) contains  $\|u_{k+1}\|_2$ . In the ordinary Arnoldi method, this quantity is one. In the  $B$ -Arnoldi method, however,  $\|u_{k+1}\|_B$  is one, and  $\|u_{k+1}\|_2$  will generally be different from one.

- **Restarting.** Let  $Q$  be orthogonal. By (4.3), we can transform the  $B$ -Arnoldi decomposition  $CU = UH - \beta u e_k^T$  to

$$C(UQ) = (UQ)(Q^T H Q) + \beta u (e_k^T Q)$$

while preserving the  $B$ -orthonormality of  $UQ$ . (In our terminology the transformed decomposition is a  $B$ -orthonormal Krylov decomposition.) Thus we are free to perform orthogonal similarities on the Rayleigh quotient  $H_k$ . This means that both implicit and Krylov–Schur restarting can be applied to the  $B$ -Arnoldi method without change.

- **Deflation.** Deflation is somewhat more complicated than restarting. To see what is involved, suppose we have transformed our decomposition into the form

$$C(u_1 \ U_2) = (u_1 \ U_2) \begin{pmatrix} \nu & g_{12}^T \\ g_{21} & G_{22} \end{pmatrix} + u_{k+1} (\gamma_{k+1,1} \ g_{k+1,2}^T)^H.$$

Then we can deflate the pair  $(\nu, u_1)$  by setting  $g_{21}$  and  $\gamma_{k+1,1}$  to zero. The error introduced into the decomposition by this deflation is

$$\left\| (U_2 \ u_{k+1}) \begin{pmatrix} g_{21} \\ \gamma_{k+1,1} \end{pmatrix} \right\|_2 \leq \|(U_2 \ u_{k+1})\|_2 \left\| \begin{pmatrix} g_{21} \\ \gamma_{k+1,1} \end{pmatrix} \right\|_2.$$

The problem is that we must estimate  $\|(U_2 \ u_{k+1})\|_2$ , which is a rather large matrix. However, in the restarted Arnoldi method this matrix will be in main memory and we can use the bound

$$\|A\|_2^2 \leq \|A\|_1 \|A\|_\infty$$

to compute a cheap bound. It should be noted that when we are deflating Ritz values the vector  $g_{21}$  is zero, so that all we need to calculate is  $\|u_{k+1}\|_2$ .

—

To summarize, the  $B$ -Arnoldi method for the generalized eigenproblem with shift-and-invert enhancement is essentially a minor modification of the standard Arnoldi method, the main difference being the need to compute the matrix-vector products of the form  $Bv$ .

### The restarted $B$ -Lanczos method

We have already noted that if  $A$  is symmetric then  $C = (A - \kappa B)^{-1}B$  is  $B$ -symmetric; i.e.,  $BC$  is symmetric. It follows that the Rayleigh quotient  $H = U^TBCU$  is symmetric; and since  $H$  is Hessenberg, it must be tridiagonal. Thus the  $B$ -Arnoldi method becomes a  $B$ -Lanczos method. Just as the restarted  $B$ -Arnoldi method is a slightly modified version of the usual restarted Arnoldi method, so the restarted  $B$ -Lanczos method is a modification of the ordinary restarted Lanczos method (see §3.2).

### The $B$ -Lanczos method with periodic reorthogonalization

In §3.3 we described the Lanczos method with periodic reorthogonalization. Once again the  $B$  version of the method can be obtained by obvious modifications of the original method. The estimates of the deterioration of orthogonality in Algorithm 3.2 are unchanged, although they now estimate deviation from  $B$ -orthogonality.

When it is necessary to reorthogonalize, the columns of  $U_k$  must be brought in from backing store in blocks that fit in main memory. Thus we cannot implement the classical Gram–Schmidt algorithm in the  $B$ -inner product; the best we can hope for is a block modified Gram–Schmidt algorithm. If the block size is small, it makes sense to store the columns of  $BU_k$ , as suggested in (4.5), to avoid forming matrix–vector products involving  $B$ . On the other hand if the block size is large or multiplications by  $B$  are cheap, it may be more efficient to process each block by the classical Gram–Schmidt algorithm as in (4.4).

### Semidefinite $B$

Since the generalized shift-and-invert transform  $C = (A - \kappa B)^{-1}B$  does not involve the inverse of  $B$ , the algorithms described above can be formally applied when  $B$  is positive semidefinite. In this case the  $B$ -inner product is said to be a semi-inner product, and the  $B$ -norm a seminorm. The  $B$  semi-inner products is blind to vectors lying in the null space of  $B$ —that is,  $(x + z, y) = (x, y)$  whenever  $z \in \mathcal{N}(B)$ —and this fact presents complications for our algorithms.

To see what is going on, we need a little geometry. We first observe that the null space of  $B$  and  $C$  are the same, and  $\text{rank}(C) = n - \text{null}(B)$ . To see this, note that if  $x$  is a null vector of  $B$  then  $x$  is a null vector of  $C$ . On the other hand if  $x$  is a null vector of  $C$ , then  $(A - \kappa B)^{-1}Bx = 0$ . Multiplying by  $A - \kappa B$ , we get  $Bx = 0$ , so that  $x \in \mathcal{N}(B)$ . The result now follows from the fact that for any matrix  $C$  of order  $n$ ,  $\text{rank}(C) + \text{null}(C) = n$ .

We will now make the simplifying assumption that

$$\mathcal{R}(C) \cap \mathcal{N}(B) = \{0\}, \quad (4.8)$$

so that any vector  $z$  can be uniquely written in the form  $z = x + y$ , where  $x \in \mathcal{R}(C)$  and  $y \in \mathcal{N}(B)$ . (This decomposition of  $z$  into  $x + y$  is *not* an orthogonal decomposition.) The assumption is always true when  $C$  is nondefective. For if there is an  $x$  such that

$Cx$  is nonzero and lies in  $\mathcal{N}(B)$ , then  $C^2x = 0$ , something that can happen only if  $C$  has a nontrivial Jordan form.

Now the eigenvectors of  $C$ , other than the null vectors of  $B$ , lie in  $\mathcal{R}(C)$ . Consequently, if we are going to use a Krylov method to calculate eigenvectors of  $B$  we must insure that the Krylov vectors lie in  $\mathcal{R}(C)$ . Mathematically, this can be done as follows. Let  $v$  be, say, a random vector and let  $u_1 = Cv/\|Cv\|_B$ . Then  $u_1 \in \mathcal{R}(C)$ . Now suppose that  $u_1, \dots, u_k$  lie in  $\mathcal{R}(C)$ . Since  $u_{k+1}$  is a linear combination of  $u_1, \dots, u_k$  and  $Cu_k$ , it must also lie in  $\mathcal{R}(C)$ . Thus if we start with a vector in  $\mathcal{R}(C)$ , our Krylov subspaces lie in  $\mathcal{R}(C)$ .

In practice, at each stage rounding error will introduce small errors lying in  $\mathcal{N}(B)$ . Since the  $B$ -inner product cannot see vectors in  $\mathcal{N}(B)$ , it is possible for the Lanczos or Arnoldi process to magnify these small errors, and such magnifications are empirically observed to happen. There are two potential dangers here.

1. The null-space error in  $U_k$  may somehow effect the Rayleigh quotient  $H_k$ , so that the primitive Ritz vectors are calculated inaccurately.
2. Even if a primitive Ritz vector  $w$  is calculated accurately, the null-space error may contaminate the Ritz vector  $U_kw$ .

To analyze these possibilities, let us suppose that we have the  $B$ -Arnoldi decomposition

$$CU_k = U_k H_k + \beta_k u_{k+1} \mathbf{e}_k^T + F_k, \quad (4.9)$$

where  $F_k$  is an error matrix of order of the rounding unit that accounts for rounding errors made during the Arnoldi process. Write

$$U_k = \tilde{U}_k + V_k,$$

where the columns of  $\tilde{U}_k$  are in  $\mathcal{R}(C)$  and the columns of  $V_k$  are in  $\mathcal{N}(B)$ . Then it follows that

$$BU_k = B\tilde{U}_k + O(\epsilon_M), \quad (4.10)$$

and

$$\tilde{U}_k^T B \tilde{U}_k = I + O(\epsilon_M). \quad (4.11)$$

The contamination of the Rayleigh quotient is not a problem. Specifically, from (4.10) we have

$$H_k = U_k^T B (A - \kappa B)^{-1} BU_k + O(\epsilon_M) = \tilde{U}_k^T B (A - \kappa B)^{-1} B \tilde{U}_k + O(\epsilon_M).$$

It then follows from (4.11) that  $H_k$  is essentially the Rayleigh quotient with respect to a set of vectors uncontaminated by contributions from  $\mathcal{N}(B)$ .

It turns out that the errors  $V_k$  do not contaminate the Ritz vectors either — at least asymptotically. To see this let  $(\nu, w)$  be a primitive Ritz pair. From (4.9) it follows that

$$CU_kw = \nu U_kw + \beta_k \omega_k u_{k+1} + O(\epsilon_M), \quad (4.12)$$

where, as usual,  $\omega_k$  is the last component of  $w$ . Now the left-hand side of this relation has no components in  $\mathcal{N}(B)$ . Hence the null-space error in  $U_kw$  is small in proportion as the quantity

$$\frac{|\beta_k \omega_k|}{|\nu|} \|u_{k+1}\|_2$$

is small. But  $|\beta_k \omega_k| \|u\|_2$  is the residual norm of the Ritz pair  $(\nu, U_kw)$ . Hence, as the Ritz pair converges, the null-space error will go to zero. The following example illustrates this phenomenon.

**Example 4.3.** Let  $D_k = \text{diag}(1, .95, \dots, .95^{k-1})$ . For  $m > n/2$ , let

$$B = \text{diag}(I_m, 0_{n-m})$$

and

$$C = \begin{pmatrix} D_m + 0.1I & 0 \\ (I_{n-m} & 0) & 0 \end{pmatrix}.$$

In this example we apply the  $B$ -Arnoldi method to  $C$  with  $n = 500$  and  $m = 400$ . The starting vector  $u_1$  is of the form  $Cr/\|Cr\|_B$ , where  $r$  is a random vector. Each multiplication by  $C$  was followed by the addition of a random perturbation on the order of the rounding unit. The following table summarizes the results. In it  $P$  denotes the projection onto  $\mathcal{N}(B)$  corresponding to the decomposition  $\mathbb{R}^n = \mathcal{R}(C) + \mathcal{N}(B)$  [see (4.8) and the following discussion].

| $k$ | $PU_{k+1}$ | $P(U_kw)$ | $ \beta_k \omega_k $ | $P(U_{k+1}Q)$ | $\ R^{-1}\ _2$ |  |
|-----|------------|-----------|----------------------|---------------|----------------|--|
| 5   | 6.4e-12    | 8.9e-13   | 2.8e-02              | 6.1e-13       | 9.1e+00        |  |
| 10  | 1.9e-10    | 2.0e-11   | 8.5e-02              | 6.5e-13       | 9.9e+00        |  |
| 15  | 5.4e-09    | 2.6e-10   | 4.8e-02              | 6.5e-13       | 1.0e+01        |  |
| 20  | 1.5e-07    | 9.2e-09   | 6.5e-02              | 6.6e-13       | 1.0e+01        |  |
| 25  | 6.2e-06    | 2.8e-09   | 4.7e-04              | 6.7e-13       | 1.0e+01        |  |
| 30  | 3.3e-04    | 6.3e-10   | 2.0e-06              | 6.8e-13       | 1.0e+01        |  |
| 35  | 2.2e-02    | 9.2e-11   | 4.1e-09              | 6.8e-13       | 1.0e+01        |  |
| 40  | 2.1e+00    | 2.9e-12   | 1.4e-12              | 7.8e-13       | 1.0e+01        |  |
| 45  | 3.8e+02    | 7.4e-14   | 7.8e-17              | 8.3e-13       | 1.0e+01        |  |
| 50  | 1.1e+05    | 6.8e-14   | 1.3e-21              | 4.0e-12       | 1.0e+01        |  |
| 55  | 4.9e+07    | 6.8e-14   | 6.4e-27              | 1.8e-09       | 1.0e+01        |  |
| 60  | 4.4e+10    | 6.8e-14   | 9.0e-33              | 1.4e-06       | 1.0e+01        |  |

The second column contains the norm of the component of  $U_{k+1}$  in  $\mathcal{N}(B)$ , which is seen to grow steadily as  $k$  increases. The third column contains the norm of the component in  $\mathcal{N}(B)$  of the Ritz vector corresponding to the fifth eigenvalue of  $C$ . The fourth column contains the product  $|\beta_k \omega_k|$ , which along with  $\|u_{k+1}\|_2$  estimates the residual norm. Initially the component in the null space increases; but as the residual norm decreases so does the component. We will return to the remaining columns later.

So far all is according to theory. However, note that the null-space error (column 2) ultimately increases far above one. Other examples of rapid growth of null-space error may be found in the literature. This puts a limit on how far we can extend an Arnoldi or Lanczos sequence, since in finite precision arithmetic the null space error will ultimately overwhelm any useful information in the Krylov vectors. The following suggests a way of purging the unwanted components from the entire sequence. It is based on the following characterization of the effects of a step of implicit restarting with zero shift.

**Theorem 4.4.** Let  $AU_k = U_{k+1}\hat{H}_k$  be an Arnoldi decomposition, where  $\hat{H}_k$  is unreduced. Let  $A\tilde{U}_{k-1} = \tilde{U}_k\tilde{H}_k$  be the decomposition resulting from one step of zero-shift implicit restarting. If  $\hat{H}_k = QR$  is the QR factorization of  $\hat{H}_k$ , then (up to column scaling)  $\tilde{U}_k = U_k Q$ .

**Proof.** Because  $\tilde{H}_k$  is unreduced Hessenberg,  $Q_k$  is unreduced Hessenberg and  $R_k$  is nonsingular. Define  $\tilde{U}_k = U_{k+1}Q$ . Then

$$CU_k = U_{k+1}\hat{H}_k = U_{k+1}QR = \tilde{U}_kR. \quad (4.14)$$

By definition  $\tilde{U}_{k-1} = U_k Q[1:k, 1:k-1]$ . Hence if we multiply equation (4.14) by  $Q[1:k, 1:k-1]$ , we get

$$C\tilde{U}_{k-1} = \tilde{U}_k RQ[1:k, 1:k-1] \equiv \tilde{U}_k \tilde{H}_{k-1}. \quad (4.15)$$

Since  $R$  is nonsingular and  $Q[1:k, 1:k-1]$  is an unreduced Hessenberg matrix,  $\tilde{H}_{k-1} = RQ[1:k, 1:k-1]$  is also unreduced Hessenberg. Thus (4.15) is an unreduced Arnoldi decomposition. Moreover, from (4.14) the first column of  $\tilde{U}_k$  satisfies

$$Cu_1 = \tilde{u}_1 r_{11}.$$

Since  $r_{11} \neq 0$ , the first column of  $\tilde{U}_k$  is a multiple of  $Cu_1$ . Hence by the uniqueness of the unreduced Arnoldi decomposition, (4.15) is the decomposition that results from one step of zero-shifted implicit restarting. ■

Although we have proved this theorem for the ordinary Arnoldi process, it also holds for the  $B$ -Arnoldi process, even when  $B$  is semidefinite. The chief difference is that  $U_{k+1}$  is no longer unique. Instead, we can add to it any matrix  $V$  in  $\mathcal{N}(B)$  such that  $V\hat{H}_k = 0$ .

The application of this theorem is simple. From (4.9) we have

$$CH_k R^{-1} + F_k R^{-1} = U_{k+1} Q = \tilde{U}_k.$$

Since the left-hand side of this equation is in  $\mathcal{R}(C)$ , so is the right-hand side — i.e., one step of zero-shifted implicit restarting purges the  $B$ -Arnoldi vectors of null-space error. For this to happen it is essential that  $R^{-1}$  not be large, since in the presence of rounding error it will magnify the residual error  $F_k$  in the decomposition.

The last two columns in (4.13) illustrate the effectiveness of this method of purging null-space error. The fifth column is the null-space error in  $U_{k+1} Q$ . For the first 50 iterations it remains satisfactorily small. However, thereafter it grows. The reason is that the null-space error in  $U_k$  (column 2) has grown so large that it is destroying information about the component of  $U_k$  in  $\mathcal{R}(C)$ .

—

The  $B$ -Arnoldi and Lanczos methods with semidefinite  $B$  are imperfectly understood at this time. The chief problem is that the null-space error in the Arnoldi vectors can increase until it overwhelms the column-space information. Since in practice the Arnoldi method must be used with restarts, including a zero shift among the restart shifts may suffice to keep the errors under control, although some experimentation may be required to determine the longest safe restarting interval. It helps that we can tell when the null-space errors are out of hand by the sudden growth of the Arnoldi vectors.

With the Lanczos algorithms, the problem of purification is analogous to that of reorthogonalization — it requires access to the entire set of Lanczos vectors. The cure is to perform periodic purging in the same style as periodic reorthogonalization. Unfortunately, at present we lack a reliable recursion for the null-space error, which is needed to know when to purge.

#### 4.3. NOTES AND REFERENCES

##### **The generalized shift-and-invert transformation and the Lanczos algorithm**

The idea of using a shift-and-invert strategy for large generalized eigenvalue problems arose in connection with the Lanczos process. Ericsson and Ruhe [72] proposed applying the Lanczos algorithm to the matrix  $R(A - \kappa B)^{-1} R^T$ , where  $B = R^T R$  is the Cholesky factorization of  $B$ . This approach has the difficulty that it requires two factorizations, one of  $B$  and one of  $A - \kappa B$  (however, the authors assume that one will use several shifts, so that the initial factorization of  $B$  can be prorated against the several factorizations of  $A - \kappa B$ ). They called their transformation “the spectral transformation.” Here we have preferred the more informative “shift-and-invert transformation.”

The transform  $(A - \kappa B)^{-1} B$  appears implicitly in a book by Parlett [206, p. 319], who rejects it. He also gives a version of the Lanczos algorithm for  $B^{-1} A$ , in which the Lanczos vectors are  $B$ -orthogonal, which, however, requires a factorization of  $B$ .

Scott [236] applied Parlett’s algorithm to the pencil  $[B(A - \kappa B)^{-1} B, B]$ , showing that it required no factorization of  $B$ . (He also says that the algorithm is not new, citing

two technical reports [183, 282].) When reduced to a standard eigenvalue problem by multiplying by  $B^{-1}$ , this pencil becomes  $(C, I)$ , in which  $C$  is what we have been calling the generalized shift-and-invert transform. He credits Parlett with recognizing that the algorithm can be regarded as the Lanczos algorithm in the  $B$ -inner product. He also considers the case of semidefinite  $B$ .

### Residuals and backward error

Theorem 4.1 follows from generalizations of a theorem of Rigal and Gaches [216] for the backward error of linear systems and is due to Frayssé and Toumazou [77] and Higham and Higham [112].

### Semidefinite $B$

The basic paper for the semidefinite  $B$ -Lanczos algorithm is by Nour-Omid, Parlett, Ericsson, and Jensen [184]. They point out that the null-space error in the Lanczos vectors can grow and give an example in which it increases 19 orders of magnitude in 40 iterations. They also point out that the formula (4.12) can be used to purge a Ritz vector of null-space error. (The same formula was used earlier by Ericsson and Ruhe [72], but for a different purpose.) The fact that the purging becomes unnecessary as a Ritz vector converges appears to have been overlooked.

The hypothesis (4.8) that  $\mathcal{R}(C)$  and  $\mathcal{N}(B)$  have only a trivial intersection is violated in practical problems. Ericsson [71] considers these problems in detail for the Lanczos algorithm. His solution is a double purging of the Ritz vectors based on the formula (4.12). Meerbergen and Spence [176] treat the nonsymmetric case — i.e., the semidefinite  $B$ -Arnoldi algorithm. Their example of the need for purification increases the null-space error by 13 orders of magnitude in 11 iterations. They introduce the elegant method of using zero-shift implicit restarting to purge the null-space error, which they combine with the purification of Nour-Omid et al. to handle the problem of overlapping null and column spaces. Finally, the authors give a rounding-error analysis of their procedures.

# 6

---

## ALTERNATIVES

Although the most widely used algorithms for large eigenproblems are based on Krylov sequences, there are other approaches. In this chapter we will consider two alternatives: subspace iteration and methods based on Newton's method — particularly the Jacobi–Davidson method.

Subspace iteration is a block generalization of the power method. It is an older method and is potentially slow. But it is very reliable; its implementation is comparatively simple; and it is easy to use. Moreover, combined with shift-and-invert enhancement or Chebyshev acceleration, it sometimes wins the race.

We next turn to methods based on Newton's method for solving systems of nonlinear equations. Since Newton's method requires the solution of a linear system, these methods appear to be mostly suitable for dense problems. But with some care they can be adapted to large sparse eigenproblems. When we combine Newton's method with Rayleigh–Ritz refinement, we get the Jacobi–Davidson algorithm, with which this chapter and volume conclude.

### 1. SUBSPACE ITERATION

In motivating the power method with starting vector  $u_0$  (§1.1, Chapter 2) we assumed that  $A$  was diagonalizable and expanded

$$A^k u_0 = \gamma_1 \lambda_1^k x_1 + \gamma_2 \lambda_2^k x_2 + \gamma_3 \lambda_3^k x_3 + \dots \quad (1.1)$$

in terms of the eigenvectors  $x_j$  of  $A$ . If

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$$

and  $\gamma_1 \neq 0$ , then the first term in the expansion dominates, and  $A^k u_0$ , suitably scaled, converges to  $x_1$ .

When  $|\lambda_2|$  is near  $|\lambda_1|$ , the convergence of the power method will be slow. However, if we perform the power method on a second vector  $u'_0$  to get the expansion

$$A^k u'_0 = \gamma'_1 \lambda_1^k x_1 + \gamma'_2 \lambda_2^k x_2 + \gamma'_3 \lambda_3^k x_3 + \dots ,$$

we can eliminate the effect of  $\lambda_2$  by forming the linear combination

$$\gamma'_2 A^k u_0 - \gamma_2 A^k u'_0 = (\gamma_1 \gamma'_2 - \gamma'_1 \gamma_2) \lambda_1^k x_1 + (\gamma_3 \gamma'_2 - \gamma'_3 \gamma_2) \lambda_3^k x_3 + \dots,$$

whose convergence depends of the ratio  $|\lambda_3/\lambda_1|$ . If this ratio is less than  $|\lambda_2/\lambda_1|$ , the effect is to speed up the convergence.

Subspace iteration is a generalization of this procedure. Beginning with an  $n \times m$  matrix  $U_0$ , one considers the powers  $A^k U_0$ . It turns out that power subspace  $\mathcal{R}(A^k U_0)$  generally contains a more accurate approximation to  $x_1$  than could be obtained by the power method. Moreover, the power subspace contains approximations to other dominant eigenspaces.

However, there are two problems that must be addressed before we can derive an algorithm based on the above observations. First, the columns of  $A^k U_0$  will tend toward linear dependence, since they all tend toward the dominant eigenspaces of  $A$ . But in fact we are only interested in the space spanned by  $A^k U_0$ . Hence we can cure the problem of dependency by maintaining an orthonormal basis for  $\mathcal{R}(A^k U_0)$ . This cure leads to an interesting and useful connection between subspace iteration and the QR algorithm. The second problem is how to extract approximations to eigenvectors from the space. We could, of course, use a Rayleigh–Ritz procedure. However, since we are working with orthonormal vectors, it is more natural to use a variant of the Rayleigh–Ritz method that computes the dominant Schur vectors of  $A$ .

In §1.1 we will develop the basic theory of the algorithm. In §1.2 we will show how to implement subspace iteration. Finally, in §1.3 we will treat the symmetric case and in particular show how to use Chebyshev polynomials to accelerate convergence.

## 1.1. THEORY

As suggested in the introduction to this section, the theory of subspace iteration divides into three parts: convergences of the subspaces, the relation to the QR algorithm, and the extraction of Schur vectors. We will treat each of these in turn.

### Convergence

Let  $U_0$  be an  $n \times m$  matrix. We wish to show that as  $k$  increases the space  $\mathcal{R}(A^k U_0)$  contains increasingly accurate approximation to the dominant eigenspaces of  $A$  and also to determine the rates of convergence. We will suppose that  $A$  has the spectral representation [see (1.7), Chapter 4]

$$A = X_1 L_1 Y_1^H + X_2 L_2 Y_2^H + X_3 L_3 Y_3^H,$$

where  $L_1$  is of order  $\ell$ ,  $L_2$  is of order  $m-\ell$ , and  $L_3$  is of order  $n-m$ . The eigenvalues of these matrices, whose union is  $\Lambda(A)$ , are assumed to be ordered so that

$$\underbrace{|\lambda_1| \geq \dots \geq |\lambda_\ell|}_{\Lambda(L_1)} > \underbrace{|\lambda_{\ell+1}| \geq \dots \geq |\lambda_m|}_{\Lambda(L_2)} > \underbrace{|\lambda_{m+1}| \geq \dots \geq |\lambda_n|}_{\Lambda(L_3)}. \quad (1.2)$$

Since  $(X_1 \ X_2 \ X_3)^{-1} = (Y_1 \ Y_2 \ Y_3)^H$ , we can write

$$U_0 = (X_1 \ X_2) \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} + X_3 C_3, \quad (1.3)$$

where  $C_i = Y_i^H U_0$ . Multiplying (1.3) by  $A^k$  and recalling that  $A X_i = X_i L_i$ , we get

$$\begin{aligned} A^k U_0 &= (X_1 \ X_2) \begin{pmatrix} L_1^k & 0 \\ 0 & L_2^k \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} + X_3 L_3^k C_3 \\ &= X_1 L_1^k C_1 + X_2 L_2^k C_2 + X_3 L_3^k C_3. \end{aligned}$$

The bottommost expression in this equation is analogous to (1.1). Moreover, (1.2) suggests that the first term in the expression will dominate, so that the column space of  $A^k U_0$  will approach the eigenspace  $\mathcal{R}(X_1)$ .

To make this more precise, assume that  $C_{12} = (C_1^H \ C_2^H)^H$  is nonsingular and let

$$D = (D_1 \ D_2) = C_3 C_{12}^{-1},$$

where  $D_1$  has  $\ell$  columns. Then

$$V_k \equiv A^k U_0 C_{12}^{-1} \text{diag}(L_1^{-k}, L_2^{-k}) = (X_1 \ X_2) + X_3 (L_3^k D_1 L_1^{-k} \ L_3^k D_1 L_2^{-k}).$$

If we partition  $V_k = (V_1^{(k)} \ V_2^{(k)})$ , then

$$V_1^{(k)} = X_1 + L_3^k D_1 L_1^{-k}.$$

By construction  $\mathcal{R}(V_1^{(k)}) \subset \mathcal{R}(A^k U_0)$ . Moreover, by Theorem 2.9, Chapter 1, we have that for any  $\epsilon > 0$

$$\|L_3^k D_1 L_1^{-k}\| = O \left[ \left( \frac{|\lambda_{m+1}|}{|\lambda_\ell|} + \epsilon \right)^k \right].$$

Thus we see that the column space of  $A^k U_0$  contains an approximation to the eigenspace  $\mathcal{R}(X_1)$  that converges essentially as  $|\lambda_{m+1}/\lambda_\ell|^k$ .

We summarize these results in the following theorem, in which we recapitulate a bit.

**Theorem 1.1.** *Let the matrix  $A$  of order  $n$  have the spectral representation*

$$A = X_1 L_1 Y_1^H + X_2 L_2 Y_2^H + X_3 L_3 Y_3^H,$$

*where  $L_1$  is of order  $\ell$ ,  $L_2$  is of order  $m-\ell$ ,  $L_3$  is of order  $n-m$ , and their eigenvalues are ordered so that*

$$\underbrace{|\lambda_1| \geq \cdots \geq |\lambda_\ell|}_{\Lambda(L_1)} > \underbrace{|\lambda_{\ell+1}| \geq \cdots \geq |\lambda_m|}_{\Lambda(L_2)} > \underbrace{|\lambda_{m+1}| \geq \cdots \geq |\lambda_n|}_{\Lambda(L_3)}.$$

Let  $U_0 \in \mathbb{C}^{n \times m}$  be given and write

$$U_0 = (X_1 \ X_2) \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} + X_3 C_3,$$

where  $C_i = Y_i^H U_0$ . Let  $\theta_k$  be the largest canonical angle between  $\mathcal{R}(A^k U_0)$  and  $\mathcal{R}(X_1)$ . If the matrix  $(C_1^H \ C_2^H)^H$  is nonsingular, then for any  $\epsilon > 0$ ,

$$\theta_k = O \left[ \left( \frac{|\lambda_{m+1}|}{|\lambda_\ell|} + \epsilon \right)^k \right]. \quad (1.4)$$

There are four comments to be made about this theorem.

- The theorem shows that subspace iteration is potentially much faster than the power method. For if  $\ell = 1$ , then the convergence depends essentially on the ratio  $|\lambda_{m+1}/\lambda_1|$ , which is in general smaller than  $|\lambda_2/\lambda_1|$ .
- The theorem contains no hypothesis that  $A$  is nondefective. This suggests that subspace iteration can deal with defectiveness, which Krylov sequence methods do not handle well [see the discussion following (3.14), Chapter 4]. If, however, the eigenvalues  $\lambda_\ell$  and  $\lambda_{m+1}$  are nondefective, then Theorem 2.9, Chapter 1, implies that we can take  $\epsilon = 0$  in (1.4).
- We are free to position  $\ell$  at any point in the spectrum of  $A$  where there is a gap in the magnitudes of the eigenvalues. If, for example, the magnitudes of the first  $m$  eigenvalues of  $A$  are distinct, the columns space of  $A^k U_0$  will contain approximations to each of the eigenspaces corresponding to the first  $\ell$  eigenvalues of  $A$ . However, the convergence of these approximations deteriorates as  $\ell$  increases, since the convergence ratio is essentially  $|\lambda_{m+1}/\lambda_\ell|$ .
- The theorem would appear to depend on choosing  $m$  so that  $|\lambda_m|$  is strictly greater than  $|\lambda_{m+1}|$ . But if that condition fails, we can reduce the size of  $U_0$  to the point where the hypothesis is satisfied (at the very worst we might end up with  $m = \ell$ ). Theorem 1.1 then applies. Moreover, adding back the missing columns of  $U_0$  can only improve the accuracy of the approximation to  $\mathcal{R}(X_1)$ , so that the theorem is valid for the original  $U_0$ . This observation is important in applications, since we will not know the eigenvalues of  $A$  when we choose  $m$ .

### The QR connection

One difficulty with the naive version of subspace iteration is that the columns of  $A^k U_0$  tend to become increasingly dependent. For example, if  $|\lambda_1| > |\lambda_2|$ , the columns will all tend to lie along the dominant eigenvector of  $A$ . Fortunately, we are not interested in the matrix  $A^k U_0$  — only the subspace spanned by it. Consequently, when the columns of  $A^k U_0$  begin to deteriorate, we can replace it with  $A^k U_0 S$ , where  $S$  is chosen to restore independence. A natural choice is to make  $A^k U_0 R$  orthogonal. If

we orthogonalize by computing a QR factorization after each multiplication by  $A$ , we get the following algorithm.

1. **for**  $k = 0, \dots$
  2.      $V = AU_k$
  3.     Compute the QR factorization  $V = U_{k+1}R_{k+1}$
  4. **end for**  $k$
- (1.5)

An easy induction shows that if  $\check{R}_k = R_1 \cdots R_k$

$$U_k \check{R}_k = A^k U_0;$$

that is,  $Q_k R_k$  is the QR factorization of  $A^k U_0$ . Now recall (Theorem 2.1, Chapter 2) that if  $A^k = \check{Q}_k \check{R}_k$  is the QR decomposition of  $A^k$  then  $\check{Q}_k$  and  $\check{R}_k$  are the accumulated Q- and R-factors from the unshifted QR algorithm. This suggests that subspace iteration is in some way connected with the QR algorithm. In fact, the proof of Theorem 2.2, Chapter 2, can be adapted to establish the following result.

**Theorem 1.2.** *Let  $U_0 \in \mathbb{C}^{n \times m}$  be orthonormal and let*

$$X^{-1}AX = \Lambda \equiv \text{diag}(\lambda_1, \dots, \lambda_n),$$

where

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0. \quad (1.6)$$

Suppose that  $X^{-1}U_0$  has an LU decomposition  $LT$ , where  $U$  is unit lower trapezoidal and  $T$  is upper triangular. Partition  $X = (X_1 \ X_2)$ , and let  $X_1 = QR$  be the QR factorization of  $X_1$ . If  $A^k U_0$  has the QR factorization  $U_k R_k$ , then there are diagonal matrices  $D_k$  with  $|D_k| = I$  such that  $U_k D_k \rightarrow Q$ .

The importance of this theorem is that the matrices  $U_k$  generated by (1.5) converge to the Schur vectors spanning the dominant eigenspaces

$$\mathcal{X}_j = \text{span}(x_1, \dots, x_j), \quad j = 1, \dots, m.$$

[see the comments at (1.9), Chapter 1]. The diagonal elements of  $R$  converge to the dominant eigenvalues of  $A$ . When the hypothesis (1.6) is not satisfied, so that some eigenvalues have equal magnitudes, the corresponding columns of  $Q_k$  will not settle down. Nonetheless, where there is a gap in the magnitudes of the eigenvalues, the space spanned by the corresponding columns of  $Q_k$  converge to the corresponding dominant eigenspace.

As in Theorem 2.2, Chapter 2, the convergence of the  $j$ th column of  $Q_k$  is governed by the largest of the ratios  $|\lambda_j/\lambda_{j-1}|$  and  $|\lambda_{j+1}/\lambda_j|$  (but only the latter when  $j = 1$ ). But we know that  $\mathcal{R}(Q_k)$  must contain an approximation to  $\mathcal{X}_j$  that is converging as  $|\lambda_j/\lambda_{m+1}|^k$ . We now turn to the problem of how to extract an approximation that converges at this faster rate.

### Schur–Rayleigh–Ritz refinement

Given the sequence  $Q_k$ , we can use (1.5) to generate approximations to eigenvectors of  $A$  by the standard Rayleigh–Ritz method. However, one of the strengths of subspace iteration is that it gives meaningful results when the matrix  $A$  is defective or has nearly degenerate eigenvectors. What we really want to do is to rearrange the orthogonal matrix  $Q_k$  so that its columns reflect the fast convergence of the dominant subspaces. The following procedure, which is called the *Schur–Rayleigh–Ritz (SRR) method*, accomplishes this.

1.  $V = AQ_k$
2.  $B = Q_k^H V$
3. Compute the Schur decomposition  $B = WTW^H$ , where  
the eigenvalues of  $B$  are arranged in descending  
order of magnitude
4.  $V = VW$
5. Compute the QR factorization  $V = Q_{k+1}R_{k+1}$

The motivation for this procedure is the following. The matrix  $B$  is the Rayleigh quotient  $B = Q_k^H AQ_k$ . Now if we wish to use the Rayleigh–Ritz method to compute an orthonormal approximation to the dominant eigenspace  $\mathcal{X}_j$ , we would compute an orthonormal basis  $W_j$  for the corresponding invariant subspace of  $B$  and for  $Q_k W_j$ . But the columns of  $W_j$  are just the first  $j$  Schur vectors of  $B$ . Consequently, the matrix  $Q_k W$  contains Ritz bases for all the dominant eigenspaces  $\mathcal{X}_j$ . Since  $AQ_k W = VW$ , we can obtain the iterate following the SRR refinement by orthogonalizing  $VW$ .

The full analysis of this method is complicated and we will not give it here. The basic result is that if  $|\lambda_j| > |\lambda_{j+1}|$ , so that the dominant eigenspace  $\mathcal{X}_j$  is well defined, the approximations  $\mathcal{R}[Q_k(:, 1:j)]$  converge to  $\mathcal{X}_j$  essentially as  $|\lambda_m/\lambda_j|^k$ . Thus the SRR procedure gives properly converging approximations to all the dominant eigenspaces at once.

## 1.2. IMPLEMENTATION

In this subsection we will sketch at an intermediate level what is needed to produce an implementation of subspace iteration. The reader should keep in mind that actual working code requires many ad hoc decisions, which we do not treat here. For further details, see the notes and references.

### A general algorithm

The theory of the last section shows that there are three nested operations that we must perform. The innermost operation is a power step in which we calculate  $AU$  (for notational simplicity we drop the iteration subscript). The next level is the orthogonalization of  $U$ . Finally, at the outer level is the performance of an SRR step.

The power step must be performed at each iteration. But since we are only concerned with the subspace spanned by  $U$ , we do not need to orthogonalize until there

This algorithm is an outline of a general subspace iteration for the first  $nsv$  dominant Schur vectors of a general matrix  $A$ . The input to the program is an  $n \times m$  orthonormal starting matrix  $U$ , where  $m \geq nsv$ .

1. **while** (the number of converged vectors is less than  $nsv$ )
2.     **while** (convergence is not expected)
3.         **while** (the columns of  $U$  are sufficiently independent)
4.              $U = A*U$
5.         **end while**
6.         Orthogonalize the columns of  $U$
7.         **end while**
8.     Perform an SRR step
9.     Test for convergence and deflate
10. **end while**

### Algorithm 1.1: Subspace iteration

---

is a danger of a nontrivial loss of orthogonality. Thus we may be able to perform several power steps before we need to orthogonalize  $U$ . Similarly, the SRR step is needed only when we expect some of the columns of  $U$  to have converged.

These observations are summarized in Algorithm 1.1. Because it is so general, the algorithm leaves several important questions unanswered:

1. How do we test for convergence of Schur vectors?
2. How do we deflate converged Schur vectors?
3. How can we tell when to perform an SRR step?
4. How can we tell when to orthogonalize?

We will treat each of these problems in turn.

#### Convergence

Convergence is tested immediately after an SRR step. As usual we will test a residual. To motivate the test, suppose that  $U$  is composed of the dominant Schur vectors of  $A$ , so that

$$AU = UT,$$

where  $T$  is upper triangular with its eigenvalues arranged in descending order of magnitude. In particular, the  $j$ th column of  $U$  satisfies

$$Au_j - Ut_j = 0,$$

where  $t_j$  is the  $j$ th column of  $T$ . Thus we declare that the  $j$ th column of  $U$  has converged if

$$\|r_j\|_2 \equiv \|Au_j - Ut_j\|_2 \leq \gamma \cdot \text{tol},$$

where  $\text{tol}$  is a convergence criterion (e.g.,  $\epsilon_M$ ) and  $\gamma$  is a scaling factor. A natural choice of  $\gamma$  is an estimate of some norm of  $A$ . However, if  $A$  has been obtained by shift-and-invert enhancement, then such an estimate might be too large, and a better choice might be  $\gamma = |\tau_{jj}|$ —i.e., the magnitude of the  $j$ th eigenvalue in the Schur decomposition [cf. (2.21), Chapter 5].

Our converge theory says that the Schur vectors corresponding to the larger eigenvalues will converge more swiftly than those corresponding to smaller eigenvalues. For this reason, we should test convergence of the Schur vectors in the order  $j = 1, 2, \dots$  and stop with the first one to fail the test.

This criterion is backward stable. Suppose that  $\ell$  vectors have converged. Let

$$AU - UT_\ell = R,$$

where  $T_\ell$  consists of the first  $\ell$  columns of  $T$ . Then by Theorem 2.8, Chapter 4, there is a matrix  $E$  with  $\|E\|_2 = \|R\|_2$  such that  $(A + E)U = UT_\ell$ . In other words, the first  $\ell$  columns of  $U$  are exact Schur vectors of  $A + E$ .

## Deflation

Because the columns of  $U$  converge in order, deflation simply amounts to freezing the converged columns of  $U$ . This freezing results in significant savings in the power, orthogonalization, and SRR steps.

Specifically, suppose the first  $\ell$  columns of  $U$  have converged, and partition  $U = (U_1 \ U_2)$ , where  $U_1$  has  $\ell$  columns. Then when we multiply by  $A$  we form the matrix  $(U_1, AU_2)$ —i.e., we do not multiply  $U_1$  by  $A$ . At each orthogonalization step we orthogonalize the current  $U_2$  first against  $U_1$  and then against itself. Finally, to perform an SRR step, we calculate the Rayleigh quotient

$$B = U^T AU = \begin{pmatrix} T_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}.$$

Here  $T_{11}$  is the triangular part of the Schur decomposition that corresponds to  $U_1$ . Because of the triangularity of  $T_{11}$ , the matrix  $W$  of Schur vectors of  $B$  has the form

$$W = \begin{pmatrix} I_\ell & 0 \\ 0 & W_{22} \end{pmatrix},$$

and the new  $U$  has the form

$$(U_1 \ U_2 W_{22}).$$

### When to perform an SRR step

The theory of the last section suggests that the residuals for the approximate Schur vectors will converge linearly to zero. We can use this fact to make a crude prediction of when we expect a Schur vector to converge.

Let us focus on a particular Schur vector, and suppose at step  $j$  it has a residual norm  $\rho_j$  and at step  $k > j$  it has a residual norm  $\rho_k$ . If the residual norms are converging linearly with constant  $\sigma$ , then  $\rho_k = \sigma^{k-j} \rho_j$ , so that we can estimate  $\sigma$  by

$$\sigma \cong \left( \frac{\rho_k}{\rho_j} \right)^{\frac{1}{k-j}}. \quad (1.7)$$

We can now use  $\sigma$  to predict when the residual norm will converge. Specifically, we wish to determine  $l$  so that

$$\gamma \cdot \text{tol} \geq \rho_l \cong \sigma^{l-k} \rho_k.$$

Hence

$$l - k \cong \frac{\log(\gamma \cdot \text{tol} / \rho_k)}{\log \sigma}.$$

It follows from (1.7) that

$$l \cong k + (k - j) \frac{\log(\gamma \cdot \text{tol} / \rho_k)}{\log(\rho_k / \rho_j)}. \quad (1.8)$$

The formula (1.8) is a crude extrapolation and must be used with some care. Precautions must be taken against it wildly overpredicting the value of  $l$ , especially in the early stages when linear convergence has not set in. One must also take care that the formula does not slightly underpredict  $l$ , so that an unnecessary SRR step is performed just before convergence. Finally, if  $A$  has nearly equimodular eigenvalues, it will pay to extrapolate the average of their residual norms, so that one SRR step detects the convergence of the entire cluster. For more see the notes and references.

### When to orthogonalize

As we have mentioned, the columns of the matrices  $U, AU, A^2U, \dots$  become increasingly dependent. The cure is to orthogonalize the columns of  $A^l U$  for some  $l$ . If we take  $l$  too small, then we perform unnecessary work. On the other hand, if we take  $l$  too large we risk losing information about the subspace spanned by  $A^l U$ .

To see what that risk is, let  $X = A^l U$ , and consider the effects on  $\mathcal{R}(X)$  of the perturbation  $X + E$ , where  $\|E\|_2 \leq \|X\| \epsilon_M$  — i.e., where  $E$  is comparable to the error in rounding  $X$  on a machine whose rounding unit is  $\epsilon_M$ . By Theorem 2.2, Chapter 4, if  $\tilde{Q}$  is an orthonormal basis for  $\mathcal{R}(X + E)$  and  $Q_\perp$  is an orthonormal basis for  $\mathcal{R}(X)^\perp$ , then  $\|Q_\perp^H \tilde{Q}\|_2$  is the sine of the largest canonical angle  $\theta$  between  $\mathcal{R}(X)$  and  $\mathcal{R}(X + E)$ .

To find an orthonormal basis for  $\mathcal{R}(X + E)$ , let

$$X = QR \quad (1.9)$$

be the QR factorization of  $X$ . Then

$$X + E = (Q + ER^{-1})R,$$

so that  $\mathcal{R}(X + E) = \mathcal{R}(Q + ER^{-1})$ . Now the matrix

$$\begin{aligned} \tilde{Q} &= (Q + ER^{-1})[(Q + ER^{-1})^H(Q + ER^{-1})]^{-\frac{1}{2}} \\ &= (Q + ER^{-1})(I + O(\|E\|_2)) \end{aligned}$$

has orthonormal columns. Hence if  $Q_\perp$  is any orthonormal basis for  $\mathcal{R}(X)$ , then

$$Q_\perp^H \tilde{Q} = Q_\perp^H ER^{-1} + O(\|E\|_2^2).$$

Thus  $\sin \theta \leq \|E\|_2 \|R^{-1}\|_2 + O(\|E\|_2^2)$ . Now  $\|E\|_2 = \|X\|_2 \epsilon_M$ , and by (1.9),  $\|X\|_2 = \|R\|_2$ . Hence

$$\sin \theta \lesssim \|R\|_2 \|R^{-1}\|_2 \epsilon_M \equiv \kappa(R) \epsilon_M,$$

and the degradation in the subspace of  $X$  is proportional to  $\kappa(R)$ .

To apply this result to subspace iteration, note that once the process starts converging, we will have

$$AU \cong UT,$$

where  $T$  is the Schur form from the last SRR step. Hence,

$$A^l U \cong UT^l.$$

Since  $U$  is orthonormal, the sensitivity of  $A^l U$  is proportional to  $\kappa(T^l) \leq \kappa(T)^l$ . Thus if we decide that no deterioration greater than  $10^l$  is to be tolerated, we should orthogonalize after at most  $l$  steps, where  $l$  satisfies

$$l = \left\lfloor \frac{t}{\log_{10}[\kappa(T)]} \right\rfloor. \quad (1.10)$$

An implication of this analysis is that if  $|\lambda_1| \gg |\lambda_2|$  — as it might be when we use shift-and-invert enhancement with a very accurate shift — then we will have to reorthogonalize at each step. The reason is that for any matrix  $T$  we have  $\kappa(T) \leq |\lambda_{\max}(T)/\lambda_{\min}(T)|$ , where  $\lambda_{\max}(T)$  and  $\lambda_{\min}(T)$  denote the largest and smallest eigenvalues of  $T$  in magnitude. Hence as convergence sets in, we will almost certainly have  $\kappa(T) \geq |\lambda_1/\lambda_2|$ , which by hypothesis is large. Thus  $l$  in (1.10) will be zero. The common sense of this is that the larger the value of  $|\lambda_1/\lambda_2|$ , the more all the columns

of  $AU$  will be contaminated by components of  $x_1$ , which we must eliminate by orthogonalization.

It should be stressed that after deflation, we must continue to work with the full  $T$ , not the trailing principal submatrix of  $T$  associated with the remaining subspaces. The reason is that in the powering step,  $\lambda_1$  still makes its contribution to nonorthogonality, whether or not its eigenvector has been deflated.

Once again we must treat the crude extrapolation (1.10) with care. For example, once it settles down it is likely to be pessimistic, since we have used several norm inequalities in deriving it. However, in practice the technique has shown itself to be effective.

### Real matrices

If  $A$  is real, we will want to work with the real Schur form of the Rayleigh quotient (Theorem 3.1, Chapter 2). Most of the above discussion goes through *mutatis mutandis*. The only problem is the ordering of the blocks of the real Schur form, which has been treated in §2.2, Chapter 5.

### 1.3. SYMMETRIC MATRICES

When the matrix  $A$  is symmetric, we can obtain some obvious economies. For example, the Schur vectors are now eigenvectors, so that there is no need of a postprocessing step to compute the eigenvectors. Moreover, the Rayleigh quotient is symmetric, so that we can use the more efficient symmetric eigenvalue routines to perform the SRR step. There are two economies, however, that are less obvious. First we can make do with a single array  $U$  instead of the two arrays required by the nonsymmetric algorithm. More important, we can speed up convergence by replacing the power step with a Chebyshev iteration.

#### Economization of storage

In economizing storage it is important to appreciate the tradeoffs. In statement 4 of Algorithm 1.1 we wrote  $U = AU$ , indicating that the product  $AU$  is to overwrite the contents of  $U$ . This can be done in two ways. First, we can use an auxiliary matrix  $V$  and proceed as follows.

1.  $V = A*U$
  2.  $U = V$
- (1.11)

Alternatively, we can introduce an auxiliary vector  $v$  of order  $n$  and write

1. **for**  $j = 1$  **to**  $m$
  2.      $v = A*U[:, j]$
  3.      $U[:, j] = v$
  4. **end for**  $j$
- (1.12)

Algorithm (1.12) obviously uses less storage than (1.11) — significantly less if  $n$  is large. The tradeoff is that we must perform  $m$  separate multiplications by the matrix

A. If the matrix  $A$  is sparse, it will generally be represented by some data structure, and a price must be paid for accessing that structure. In the algorithm (1.12) that price must be paid  $m$  times. In the algorithm (1.11), on the other hand, we need only access  $A$  once to compute the entire product  $AU$  — but, of course, at the cost of twice the storage, which brings the tradeoffs full circle.

So far as computing the product  $AU$  is concerned, both algorithms have the option of economizing storage. But in calculating the Rayleigh quotient  $B = U^H AU$ , we have no choice when  $A$  is non-Hermitian: we must use an extra array  $V$ . But if  $A$  is Hermitian, then so is  $B$ , and we need only compute its upper half. This can be done without an auxiliary array as follows.

1. **for**  $j = 1$  to  $m$
2.      $v = A * U[:, j]$
3.      $B[j, j:m] = v^H * U[:, j:m]$
4.      $U[:, j] = v$
5. **end for**  $j$

Thus there is a true choice in the non-Hermitian case. How that choice will be made must depend on the representation of  $A$  and the properties of the machine in question.

### Chebyshev acceleration

We have noted that if the eigenvalues of  $A$  are well separated in magnitude, then convergence will be fast, but we will perform many orthogonalization steps. Conversely, if the eigenvalues of  $A$  are clustered in magnitude, then convergence will be slow, but we will need few orthogonalization steps. In this case, however, we can use the power steps to accelerate convergence.

Specifically, instead of computing  $A^k U$  we will compute  $p_k(A)U$ , where  $p_k$  is a polynomial of degree  $k$ , chosen to enhance the gap between  $|\lambda_1|$  and  $|\lambda_{m+1}|$ . A natural choice is the scaled Chebyshev polynomial

$$p_k(t) = c_k(t/s),$$

where  $|\lambda_{m+1}| \leq s \leq |\lambda_m|$ . To see why, note that when we evaluate  $c_k(A/s)$ , we have effectively scaled  $A$  by  $s$ . For the purposes of argument, assume that we have chosen  $s = |\lambda_{m+1}|$ , so that we may assume that  $|\lambda_{m+1}| = 1$  and hence  $|c_k(\lambda_{m+1})| = 1$ . From (3.8), Chapter 4, we have

$$p_k(|\lambda_1|) = \cosh(k \cosh^{-1} |\lambda_1|) \geq \frac{e^{k \cosh^{-1} |\lambda_1|}}{2}.$$

Thus  $P_k(A)U$  will contain approximations to the dominant eigenvector that converge at least as fast as  $e^{-k \cosh^{-1} |\lambda_1|}$  converges to zero. The analogous result is true for the other eigenvalues.

When  $|\lambda_1|$  is near one, the convergence ratio

$$e^{-\cosh^{-1} |\lambda_1|}$$

can be much smaller than the usual ratio  $|\lambda_1|^{-k}$  obtained by powering. For example, suppose that  $\lambda_1 = 1.01$ , so that ordinarily we could expect a convergence ratio of  $1/1.01 \cong 0.99$ . On the other hand, if we use Chebyshev acceleration, we get a convergence ratio of 0.87 — a great improvement.

In implementing Chebyshev acceleration there are three points to consider.

1. What value should we use for the scaling factor  $s$ ?
2. How do we compute  $c_k(A/s)U$ ?
3. How does the acceleration affect our procedure for predicting when to perform orthogonalization steps?

Let  $\nu_i$  ( $i = 1, \dots, m$ ) be the Ritz values computed in the SRR step arranged in descending order of magnitude. Since by the interlacing theorem (Theorem 3.6, Chapter 1) we know that  $|\lambda_{m+1}| \leq |\nu_m| \leq |\lambda_m|$ , it is natural to take  $s = |\nu_m|$  for the scaling factor. The choice has the drawback that as the iteration converges,  $\nu_m \rightarrow \lambda_m$ , so that we do not get the best convergence ratio. But in subspace iteration we generally choose  $m$  somewhat greater than the number of eigenpairs we desire, so that this drawback is of minimal importance.

To evaluate  $c_k(A/s)U$ , we can use the three-term recurrence (see item 1 in Theorem 3.7, Chapter 4)

$$\begin{aligned} c_0(t/s) &= 1, \\ c_1(t/s) &= t/s, \\ c_{k+1}(t/s) &= 2(t/s)c_k(t/s) - c_{k-1}(t/s), \quad k = 1, 2, \dots. \end{aligned}$$

The following algorithm implements this recurrence.

1.  $V = U$
2.  $U = s^{-1} * A * V$
3. **for**  $k = 1, 2, \dots$
4.      $W = 2 * s^{-1} * A * U - V$
5.      $V = U$
6.      $U = W$
7. **end for**  $k$

Note that we have used two extra  $n \times m$  arrays —  $U$  and  $V$  — to implement the recurrence. If we generate the final  $U$  column by column, we can make do with two extra  $n$ -vectors. As above, the tradeoff is that we must call the routine to multiply  $A$  into an array more often.

Turning now to orthogonalization, we must modify the formula

$$l = \left\lfloor \frac{t}{\log_{10}[\kappa(T)]} \right\rfloor \quad (1.13)$$

for the interval between orthogonalizations in two ways. First, since  $A$  is symmetric, the Schur  $T$  of the Rayleigh quotient is diagonal. Thus we can avoid the invocation of a condition estimator and compute  $\kappa(T)$  as the ratio of  $\nu_1/\nu_m$  of the Ritz values. Second, if  $l$  is greater than one, we will be performing Chebyshev iteration, so that

$$c_l(A/s)U \cong U c_l(T/s).$$

Since  $s = |\nu_m|$ , we have

$$\kappa[c_l(T/s)] = c_k(|\lambda_1|/|\lambda_m|) \cong \rho^l,$$

where by (3.9), Chapter 4,

$$\rho = 1 + \sqrt{|\lambda_1/\lambda_m|^2 - 1}.$$

Hence we must replace the formula (1.13) by

$$l = \left\lfloor \frac{t}{\log_{10}(\rho)} \right\rfloor.$$

## 1.4. NOTES AND REFERENCES

### Subspace iteration

Subspace iteration originated in Bauer's *Treppeniteration*, [16, 1957], which bears the same relation to Rutishauser's LR algorithm [224, 1955] as the orthogonal version presented here does to the QR algorithm. In *The Algebraic Eigenvalue Problem* Wilkinson [300, 1965] introduced the orthogonal variant. For symmetric matrices Jennings [130, 1967] introduced an approximate Rayleigh–Ritz step using first-order approximations to the primitive Ritz vectors. The full Rayleigh–Ritz acceleration was introduced independently by Rutishauser [227, 1969], Stewart [249, 1969], and Clint and Jennings [41, 1970]. For nonsymmetric matrices, Clint and Jennings [42, 1971] compute Ritz vectors to get good approximations to eigenvectors from the  $U_k$ . Stewart [257, 1976] introduced and analyzed the Schur–Rayleigh–Ritz method for approximating Schur vectors.

Although the hypothesis  $|\lambda_m| > |\lambda_{m+1}|$  is not necessary for the spaces  $\mathcal{R}(U_k)$  to contain good approximations to the Schur vectors, it is necessary to prove that the SRR procedure retrieves them. A counterexample is given in [257], where it is argued that the phenomenon is unlikely to be of importance in practice.

The idea of predicting when to orthogonalize is due to Rutishauser [228] for the symmetric case and was generalized to the nonsymmetric case by Stewart [257]. The latter paper also introduces the technique of extrapolating residual norms to predict convergence.

### Chebyshev acceleration

The use of Chebyshev polynomials to accelerate subspace iteration for symmetric matrices is due to Rutishauser [228]. When  $A$  is positive definite it is more natural to base the Chebyshev polynomials on the interval  $[0, s]$  rather than on  $[-s, s]$ , as is done in the text. Chebyshev acceleration can also be performed on nonsymmetric matrices and can be effective when eigenpairs with the largest real parts are desired. For details and further references see [68].

### Software

Rutishauser [228] published a program for the symmetric eigenproblem. The program LOPSI of Stewart and Jennings [270, 271] is designed for real nonsymmetric matrices and uses Rayleigh–Ritz refinement to compute eigenvectors. The program SRRIT of Bai and Stewart [12] uses SRR steps to compute Schur vectors. Duff and Scott [68] describe code in the Harwell Library that includes Chebyshev acceleration for the symmetric and nonsymmetric cases. All these references contain detailed discussions of implementation issues and repay close study.

## 2. NEWTON-BASED METHODS

One of the most widely used methods for improving an approximation to a root of the equation  $f(x) = 0$  is the Newton–Raphson formula

$$\hat{x} = x - \frac{f(x)}{f'(x)}.$$

It is a useful fact that specific instances of this formula can be derived without differentiation by using first-order perturbation expansions. For example, suppose we wish to compute  $\sqrt{a}$  for some positive  $a$ . This amounts to solving the equation  $f(x) = x^2 - a = 0$ . The Newton correction is

$$\hat{d} = -\frac{f(x)}{f'(x)} = \frac{a - x^2}{2x}.$$

On the other hand if we write

$$(x + d)^2 = a,$$

where  $d$  is presumed small, expand the square, and drop terms in  $d^2$ , we get the approximate linear equation

$$x^2 + 2xd \cong a,$$

from which we find that

$$d \cong \frac{a - x^2}{2x}. \quad (2.1)$$

Thus the Newton correction  $\hat{d}$  and the approximation (2.1) to the true correction  $d$  are identical.

In this section we are going to explore the consequences of applying this approach to the eigenvalue problem. It turns out that it yields some old friends like the Rayleigh-quotient method and the QR algorithm. However, it also yields some new methods and suggests ways of implementing them for large sparse eigenproblems. When combined with the Rayleigh–Ritz method, it becomes the Jacobi–Davidson method.

In the next section we will introduce approximate Newton methods and a class of variants that we will call *orthogonal correction methods*. In §2.2 we will treat the Jacobi–Davidson algorithm.

## 2.1. APPROXIMATE NEWTON METHODS

This section concerns the derivation of Newton-based methods and the analysis of their orthogonal variants. We will begin with the general method and then specialize.

### A general approximate Newton method

Suppose we are given an approximate eigenpair  $(\mu, z)$  of  $A$  and wish to determine  $\eta$  and  $v$  so that  $(\mu + \eta, z + v)$  is an improved approximation. For reasons that will become clear later, we will assume that we have an approximation

$$\tilde{A} = A + E \tag{2.2}$$

to  $A$ , where  $E$  is small. For purposes of derivation, let  $(\mu + \hat{\eta}, z + \hat{v})$  be an exact eigenpair of  $A$ . Then

$$[(\tilde{A} - E) - (\mu + \hat{\eta})](z + \hat{v}) = 0.$$

If we discard products of small terms, we obtain the following approximate equation in  $\hat{\eta}$  and  $\hat{v}$ :

$$(\tilde{A} - \mu I)\hat{v}' - \hat{\eta}z \cong -(A - \mu I)z.$$

We therefore *define* approximate corrections to be the solutions of the equation

$$(\tilde{A} - \mu I)v - \eta z = -(A - \mu I)z \equiv -r.$$

Unfortunately, this equation is underdetermined: there are only  $n$  relations to determine the  $n$  components of  $v$  and the scalar  $\eta$ . To determine the system completely, we add the condition

$$w^H v = 0,$$

where  $w$  is a suitably chosen vector. For example, we could choose  $w = e_k$ , in which case the  $k$ th component of  $z + v$  remains unchanged. Later we will consider the consequences of choosing  $w = z$ .

Now in matrix form our equations become

$$\begin{pmatrix} \tilde{A} - \mu I & -z \\ w^H & 0 \end{pmatrix} \begin{pmatrix} v \\ \eta \end{pmatrix} = \begin{pmatrix} -r \\ 0 \end{pmatrix}. \quad (2.3)$$

By multiplying the first row of this relation by  $w^H(\tilde{A} - \mu I)^{-1}$  and subtracting it from the second row, we obtain the triangular system

$$\begin{pmatrix} \tilde{A} - \mu I & -z \\ 0 & w^H(\tilde{A} - \mu I)^{-1}z \end{pmatrix} \begin{pmatrix} v \\ \eta \end{pmatrix} = \begin{pmatrix} -r \\ w^H(\tilde{A} - \mu I)^{-1}r \end{pmatrix}.$$

Hence

$$v = -(\tilde{A} - \mu I)^{-1}(r - \eta z), \quad \text{where } \eta = \frac{w^H(\tilde{A} - \mu I)^{-1}r}{w^H(\tilde{A} - \mu I)^{-1}z}. \quad (2.4)$$

We will call (2.4) the *approximate Newton correction formula*. If  $\tilde{A} = A$ , we will simply call it the *Newton correction formula*.

### The correction equation

There is an alternative to the correction formula (2.4), involving only  $v$ , that is of theoretical and practical use. To derive it, write the first equation in (2.4) in the form

$$(\tilde{A} - \mu I)v = -r - \eta z.$$

The first order of business is to get rid of  $\eta$ . Let

$$P_z^\perp = I - \frac{zr_\perp^H}{r_\perp^H z},$$

where  $r_\perp$  is any vector that is orthogonal to  $r$  for which  $r_\perp^H z \neq 0$ . In the language of projectors,  $P_z^\perp$  is an oblique projector onto the orthogonal complement of  $r_\perp$  along  $z$ . In particular,  $P_z^\perp z = 0$  and  $P_z^\perp r = r$ . It follows that

$$P_z^\perp(\tilde{A} - \mu I)v = -r.$$

Now let

$$P_w^\perp = I - \frac{uw^H}{w^H u}$$

be the projector onto the orthogonal complement of  $w$  along  $u$ . Then  $P_w^\perp v = v$ . It follows that

$$P_z^\perp(\tilde{A} - \mu I)P_w^\perp v = -r, \quad v \perp w. \quad (2.5)$$

We will call (2.5) the *approximate Newton correction equation*. It actually characterizes the correction vector  $v$ , as the following theorem shows.

**Theorem 2.1.** *In the above notation, let  $\tilde{A} - \mu I$  be nonsingular. Then  $v$  satisfies (2.4) if and only if  $v$  satisfies (2.5).*

**Proof.** We have already shown that if  $v$  satisfies (2.4) then  $v$  satisfies (2.5). Suppose now that  $v$  satisfies (2.5). Then

$$(\tilde{A} - \mu I)P_w^\perp v = -r + \frac{zr_\perp^H}{r_\perp^H z}(\tilde{A} - \mu I)P_w^\perp v \equiv -r - \alpha z.$$

Since  $P_w^\perp v = v$ , we have

$$v = -(\tilde{A} - \mu I)^{-1}r - \alpha(\tilde{A} - \mu I)^{-1}z.$$

Since  $w^H v = 0$ , we have

$$\alpha = -\frac{w^H(\tilde{A} - \mu I)^{-1}r}{w^H(\tilde{A} - \mu I)^{-1}z} = \eta.$$

Thus  $v$  satisfies (2.5). ■

The correction equation says that the correction  $v$  to  $z$  satisfies a simple equation involving the projected operator

$$A_\perp = P_z^\perp(\tilde{A} - \mu I)P_w^\perp.$$

Although the equation is pretty, it does not seem to offer much computational advantage over the correction formula, which exhibits the correction explicitly. However, the correction formula is cast in terms of inverses, and their implementation requires a factorization of the matrix  $\tilde{A} - \mu I$ , which may be too expensive to compute for large sparse problems. On the other hand, if we use an iterative method based on Krylov sequences to solve the correction equation, we need only compute matrix-vector products of the form  $A_\perp y$ , where  $y$  is an arbitrary vector, something that is easily done. We will return to this point later.

### Orthogonal corrections

The presence of oblique projectors in the correction equation is disturbing. Oblique projectors can be arbitrarily large, and large projections can cause numerical cancellation in expressions that contain them. However, if we restrict  $\mu$  appropriately, we can replace the projections  $P_z^\perp$  and  $P_w^\perp$  by a single orthogonal projection.

We first observe that it is natural to take  $w = z$  — that is, to require the correction  $v$  to be orthogonal to the approximate eigenvector  $z$ . For if  $v \perp z$  and  $v$  is small compared with  $z$ , then a simple picture shows that the minimum distance between  $z$  and  $\text{span}(z + v)$  is approximately  $\|v\|_2$ . In other words,  $v$  is close to the smallest possible correction.

If we take  $w = z$ , then we may take

$$P_w^\perp = I - \frac{zz^T}{\|z\|_2} \equiv P_\perp,$$

which is an orthogonal projection. We should also like  $P_z^\perp$  to be the same. However, in this case the correction equation becomes

$$P_\perp(\tilde{A} - \mu I)P_\perp v = -r, \quad v \perp z,$$

and this equation is consistent only if  $r \perp z$ . Now by Theorem 1.4, Chapter 2, if we choose  $\mu$  to be the Rayleigh quotient

$$\mu = \frac{z^H A z}{z^H z},$$

then  $r$  is indeed orthogonal to  $z$ . Thus we have established the following result.

**Theorem 2.2.** *Let  $(\mu, z)$  be an approximate normalized eigenpair of  $A$  where*

$$\mu = z^H A z$$

*and let*

$$r = Az - \mu z.$$

*Let*

$$\tilde{A} = A + E$$

*and assume that  $\tilde{A}$  is nonsingular. Then the Newton-based correction  $v$  that is orthogonal to  $z$  is given by the correction formula*

$$v = -(\tilde{A} - \mu I)^{-1}(r + \eta z), \quad \text{where } \eta = -\frac{z^H(\tilde{A} - \mu I)^{-1}r}{z^H(\tilde{A} - \mu I)^{-1}z}. \quad (2.6)$$

*Equivalently,  $v$  is the unique solution of the correction equation*

$$(I - zz^H)(\tilde{A} - \mu I)(I - zz^H)v = -r, \quad v \perp z. \quad (2.7)$$

Three comments on this theorem.

- We call the vector  $v$  of Theorem 2.2 the *orthogonal (approximate Newton) correction*, and (2.6) and (2.7) the *orthogonal correction formula* and the *orthogonal correction equation*. But since we will be concerned exclusively with orthogonal corrections in the sequel, we will usually drop the qualifier “orthogonal” and speak of the approximate Newton correction, correction formula, and correction equation, dropping the approximate when  $A = \tilde{A}$ .

- We have used the expression “approximate Newton method” rather than simply “Newton method” because both the correction formula and the correction equation contain the approximation  $\tilde{A}$ . However, even when  $A = \tilde{A}$ , the iterated method is not strictly speaking Newton’s method. The reason is that we use the normalized vector  $(z + v)/\|z + v\|_2$  and its Rayleigh quotient in place of  $z + v$  and  $\mu + \eta$ . The distinction, however, is in the higher-order terms.
- We have used the term “orthogonal” it more than one sense. The obvious sense is that we require  $v$  to be orthogonal to  $z$ . But equally important is that the operator  $A - \mu I$  is projected onto the orthogonal complement of  $z$  in the correction equation.

### Analysis

The fact that the operator in the projection equation (2.7) is a projection of  $A$  suggests that for the purposes of analysis we should rotate the problem into a coordinate system in which the projector projects along the coordinate axes. Specifically, let  $Z = (z \ Z_\perp)$  be a unitary matrix and partition

$$Z^H A Z \equiv \begin{pmatrix} z^H \\ Z_\perp^H \end{pmatrix} A \begin{pmatrix} z & Z_\perp \end{pmatrix} = \begin{pmatrix} \mu & h^H \\ g & M \end{pmatrix}.$$

We will also write

$$Z^H A Z = \begin{pmatrix} \mu & h^H \\ g & \tilde{M} - F \end{pmatrix},$$

where

$$\tilde{M} = Z_\perp^H \tilde{A} Z_\perp \quad \text{and} \quad F = Z_\perp^H E Z_\perp$$

[see (2.2)]. In the  $Z$  coordinate system, we have the following correspondences:

$$\begin{aligned} z &\longleftrightarrow \mathbf{e}_1, \\ zz^H &\longleftrightarrow \text{diag}(1, 0), \\ I - zz^H &\longleftrightarrow \text{diag}(0, I_{n-1}), \\ r &\longleftrightarrow \begin{pmatrix} 0 \\ g \end{pmatrix}, \\ (I - zz^H)A(I - zz^H) &\longleftrightarrow \begin{pmatrix} 0 & 0 \\ 0 & M \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & \tilde{M} - F \end{pmatrix}, \\ v &\longleftrightarrow \begin{pmatrix} 0 \\ p \end{pmatrix}, \end{aligned}$$

in which  $p$  is to be determined.

The correction equation can now be written in the form

$$\begin{pmatrix} 0 & 0 \\ 0 & \tilde{M} - \mu I \end{pmatrix} \begin{pmatrix} 0 \\ p \end{pmatrix} = \begin{pmatrix} 0 \\ -g \end{pmatrix}.$$

Hence we have

$$p = -(\tilde{M} - \mu I)^{-1}g. \quad (2.8)$$

This expression corresponds to the correction formula. Thus in the  $Z$  coordinate system the correction equation and the correction formula are so simply related that there is no need to prove a theorem to establish their equivalence.

We now turn to the iterative application of the approximate Newton method. A natural quantity to measure the quality of the current approximation  $z$  is the residual norm  $\|r\|_2$ . Thus our problem is to develop a bound for the residual norm corresponding to the corrected vector  $\hat{z} = z + v$  in terms of the original residual norm  $\|r\|_2$ .

There are three ways to simplify the analysis. First, although we should, in principle, work with the normalized vector  $\hat{z}/\|\hat{z}\|_2$ , since  $v \perp z$ , we have  $\|\hat{z}\|_2^2 = \|z\|_2^2 + \|v\|_2^2 = 1 + \|v\|_2^2$ . Consequently, if  $v$  is small—which we may assume it to be in a convergence analysis— $\hat{z}$  is essentially normalized.

Second, it is not necessary to compute  $\hat{r}$ . By definition it uses the Rayleigh quotient  $\hat{\mu}$  corresponding to  $\hat{z}$ , and hence by Theorem 1.4, Chapter 2,  $\hat{r}$  has the smallest possible residual. Thus any residual norm will serve to bound  $\|\hat{r}\|_2$ , and our only problem is to find a reasonably small residual.

Third, the problem is trivial in the  $Z$  coordinate system. For in this case the residual for the corrected vector is

$$\begin{pmatrix} \mu & h^H \\ g & \tilde{M} - F \end{pmatrix} \begin{pmatrix} 1 \\ p \end{pmatrix} - \nu \begin{pmatrix} 1 \\ p \end{pmatrix},$$

where, as pointed out in the last paragraph, we may choose  $\nu$  at our convenience. The easiest choice is  $\nu = \mu + h^H p$ , which makes the first component of the residual equal to zero. The remaining component is

$$g + (\tilde{M} - \mu I)p + Fp - (h^H p)p.$$

By the correction formula (2.8) the first two terms in this expression vanish. Hence

$$\|\hat{r}\|_2 \leq (\|F\|_2 + \|h\|_2 \|p\|_2) \|p\|_2.$$

To complete the iteration cycle, we must relate  $\|p\|_2$  to  $\|r\|_2 = \|g\|_2$ . If we set

$$\tilde{\sigma} = \|(\tilde{M} - \mu I)^{-1}\|_2,$$

then from (2.8) we have  $\|p\|_2 \leq \sigma \|g\|$ . Hence

$$\|\hat{r}\|_2 \leq \tilde{\sigma} (\|F\|_2 + \tilde{\sigma} \|h\|_2 \|r\|_2) \|r\|_2.$$

Inserting iteration subscripts, we have the following theorem.

**Theorem 2.3.** Let  $(\mu_k, z_k)$  ( $k = 0, 1, \dots$ ) be a sequence of normalized approximate eigenpairs produced by the orthogonal approximate Newton method applied to the matrix  $A = \tilde{A} + E$ . Let  $r_k = Az_k - \mu_k z_k$ . Further let  $Z = (z_k \ Z_{\perp}^{(k)})$  be unitary, and define

$$\eta_k = \|z_k^H A Z_{\perp}^{(k)}\|_2,$$

$$\tilde{\sigma}_k = \|(Z_{\perp}^{(k)H} A Z_{\perp}^{(k)} - \mu_k I)^{-1}\|_2,$$

and

$$\epsilon_k = \|Z_{\perp}^{(k)H} E Z_{\perp}^{(k)}\|_2.$$

Then

$$\|r_{k+1}\|_2 \leq \tilde{\sigma}_k \epsilon_k \|r_k\|_2 + \tilde{\sigma}_k^2 \eta_k \|r_k\|_2^2. \quad (2.9)$$

### Local convergence

The recurrence (2.9) can be made the basis of a local convergence proof for the orthogonal Newton-based method. We will not present the proof here. However, the idea is to obtain uniform upper bounds  $\sigma$  on  $\sigma_k$  and  $\epsilon$  on  $\epsilon_k$  (the quantities  $\eta_k$  are uniformly bounded by  $\|A\|_2$ ). If

$$\sigma\epsilon < 1,$$

then for all sufficiently small  $\|r_k\|$ , the quantity  $\sigma_k(\epsilon_k + \tilde{\sigma}_k \eta_k \|r_k\|_2)$  is uniformly bounded below one. In particular, if  $\|r_0\|_2$  is sufficiently small, then the residual norms  $\|r_k\|_2$  converge monotonically to zero. It can then be shown that  $\mu_k$  and  $M_k$  (suitably defined) approach fixed limits  $\lambda, M$ , so that by Theorem 2.10, Chapter 4, the eigenpairs  $(\mu_k, z_k)$  converge to an eigenpair  $(\lambda, z)$  of  $A$ .

If  $\epsilon > 0$ , then the convergence of the method is linear with convergence ratios not greater than  $\sigma\epsilon$ . Since  $\mu_k$  and  $M_k$  are converging to  $\lambda$  and  $M$ , we have by Theorem 2.11, Chapter 4, that

$$\sigma_k \lesssim \frac{1}{\text{sep}(\lambda, M) - \epsilon}.$$

Hence for  $\epsilon$  not too large, the bound  $\sigma\epsilon < 1$  is essentially

$$\frac{\epsilon}{\text{sep}(\lambda, M) - \epsilon} < 1.$$

What this says is that if we are going to use an approximation  $\tilde{A}$  instead of  $A$  in our iteration, the error in the approximation must be reasonably less than the separation of  $\lambda$  from its neighbors in the sense of the function  $\text{sep}$ . In other words, convergence to a poorly separated eigenvalue requires a correspondingly accurate approximation.

When  $\epsilon = 0$ , the convergence is quadratic. When  $A$  is Hermitian,  $h_k = g_k$ , so that  $\eta_k = \|r_k\|_2$ . Hence the convergence is cubic.

### Three approximations

In the approximate Newton method, the rate of convergence depends on the accuracy of the approximation  $\tilde{A} = A + E$ . Ideally, we would take  $\tilde{A} = A$ , so that the method converges quadratically. But this is not always possible. Here we give three situations in which it may be desirable to take  $\tilde{A} \neq A$ .

- **Natural approximations.** Such approximations arise when  $A$  can be split into a matrix  $\tilde{A}$  that is easy to invert (in the sense of being able to solve systems involving  $\tilde{A} - \mu_k I$ ) and a small remainder  $E$ . For example, if  $A$  is diagonally dominant with small off-diagonal elements, we might split  $A = D - E$ , where  $D$  is the diagonal of  $A$  and  $E = D - A$ . In this case the approximate Newton method can be implemented solving only diagonal systems. Moreover, if  $E$  is small enough, the method will converge rapidly. This diagonal approximation is associated with the name of Jacobi. See the notes and references.
- **Constant shift approximations.** In the pure Newton method, we solve equations involving the matrix  $A - \mu_k I$ . Since  $\mu_k$  changes with each iteration, this may require a matrix factorization at each step. In many instances it is sufficient to replace  $A - \mu_k I$  by  $(A - \tau I)$ , where  $\tau$  is the center of a focal region containing the eigenvalues of interest. (Note that  $\tau$  is sometimes illogically called a target in the literature.) If we write  $\tilde{A} = A + E_k$ , where  $E_k = (\mu_k - \tau)I$ , then  $\tilde{A} - \mu_k I = A - \tau I$ . This is an example of an approximation that changes from iteration to iteration.

In this case, our iteration bound (2.9) simplifies to

$$\|r_{k+1}\|_2 \leq \sigma_k |\tau - \mu_k| \|r_k\|_2^2 + \sigma_k^2 \|h_k\|_s \|r_k\|_2^k,$$

where in the above notation  $\sigma_k = \|(M_k - \tau I)^{-1}\|_2$ . This bound is isomorphic to the bound (2.12), Chapter 2, for the local convergence of the QR algorithm. This is no coincidence. This variant of the approximate Newton method is closely related to the RQ variant of the QR algorithm. See the notes and references.

- **Inexact solutions.** In some applications, it is impossible to solve the equation  $A - \mu_k I$  directly. If we lack a natural approximation, then we must use an iterative method to solve the correction equation (2.7). In this case the approximate Newton's method consists of *inner iterations* to solve the correction equations and *outer iterations* consisting of Newton steps.

As with eigenvalue problems, we generally terminate an inner iteration when the residual

$$s = -r - (I - zz^H)(A - \mu I)(I - zz^H)v \tag{2.10}$$

is sufficiently small. It is natural to ask what effect a residual-based convergence criterion has on the convergence of the approximate Newton method. It turns out the analysis given above covers this case.

From Theorem 1.3, Chapter 2, there is a matrix

$$E = -\frac{sv^H}{\|v\|_2^2}$$

such that

$$[(I - zz^H)(A - \mu I)(I - zz^H) + E]v = -r. \quad (2.11)$$

Now since  $(I - zz^H)r = r$ , it follows from (2.10) that  $(I - zz^H)s = s$ . Moreover, it is reasonable to assume that the iterative method in question produces an approximate solution  $v$  that satisfies  $(I - zz^H)v = v$ . This implies that we may move  $E$  inside the brackets in (2.11), so that

$$(I - zz^H)(A + E - \mu I)(I - zz^H)v = -r.$$

Thus the effect of terminating the inner iteration with a residual  $s$  is the same as performing the outer iteration with an approximation in which  $\|E\|_2 = \|s\|_2$ . This fact along with the iteration bound (2.9) can guide our choice of a convergence tolerance for the inner iteration.

## 2.2. THE JACOBI–DAVIDSON METHOD

The Jacobi–Davidson method is a combination of the approximate Newton iteration and Rayleigh–Ritz approximation. It has the advantage that the Rayleigh–Ritz procedure can focus the Newton iteration on a region containing the eigenvalues we are interested in. Moreover, it sometimes happens that while the method is converging to one eigenpair, it produces information that later speeds convergence to other eigenpairs in the focal area. The method is new and less than well understood, and its presence in a work whose concern is mostly the tried and true is something of an anomaly. Nonetheless, it is promising, it has been used successfully, and it is technically interesting—and that is reason enough to include it.

We will begin with a general formulation of the Jacobi–Davidson algorithm and then introduce the refinements necessary to implement it.

### The basic Jacobi–Davidson method

The approximate Newton method is suitable for improving a known approximation to an eigenpair. Moreover, as we shall see, it can be combined with an effective deflation technique that prevents it from reconverging to pairs that have already been found. However, it has two drawbacks.

1. If we do not start the method close enough to an eigenpair, there is no real control over what eigenvalue the method will converge to. In particular, it may converge to eigenpairs whose eigenvalues lie entirely outside the region we are interested in.

Given a normalized starting vector  $z$ , this algorithm attempts to compute an eigenpair whose eigenvalue is near a focal point  $\tau$ . The matrix  $\tilde{A}$  is a suitable approximation of  $A$ .

1.  $V = v$
2. **for**  $k = 1, \dots, k_{max}$
3.     Using  $V$  compute a normalized Ritz pair  $(\mu, z)$  such that  
        $\mu$  is near  $\tau$
4.      $r = Az - \mu z$
5.     **if** ( $r$  is sufficiently small) **return**  $(\mu, z)$ ; **fi**
6.     Choose a shift  $\kappa$  near  $\tau$  or  $\mu$
7.     Solve the system  
        $(I - zz^H)(\tilde{A} - \kappa I)(I - zz^H)v = -r, \quad v \perp z$
8.     Orthonormalize  $v$  with respect to  $V$
9.      $V = (V \ v)$
10. **end for**  $k$
11. Signal nonconvergence

**Algorithm 2.1:** The basic Jacobi–Davidson algorithm

---

2. It is greedy.

By greedy, we mean that the method attempts to do as well as possible at each step without any attempt at global optimization. Specifically, the correction vector  $v_k$  is applied once to the current approximate  $z_k$  and discarded — much as past iterates in the power method are thrown away. In particular (up to normalization),

$$z_{k+1} = z_0 + v_0 + v_1 + \cdots + v_k. \quad (2.12)$$

Yet this is only one vector in the *correction space* spanned by  $z_0, v_0, \dots, v_k$ , and it is possible that another vector in this space will provide a better approximate eigenvector. This suggests that we save the vectors  $z_0, v_0, \dots, v_k$  and use a Rayleigh–Ritz procedure to generate  $z_{k+1}$ . Although the Rayleigh–Ritz method is not infallible, the new approximation will generally be better than (2.12). Moreover, because we can choose from among  $k+1$  Ritz pairs, we are free to choose one near a focal point, so that we can mitigate the effects of wild jumps in the Newton iterates.

Algorithm 2.1 implements the basic Jacobi–Davidson algorithm. It consists of a basic Rayleigh–Ritz step to generate a starting vector for an approximate Newton step. The correction is then folded into the correction subspace and the process repeated.

The algorithm contains three shift parameters:  $\tau$ ,  $\mu$ , and  $\kappa$ . As stated in the prologue,  $\tau$  is the center of a *focal region* in which we wish to find eigenpairs. The quantity  $\mu$  is the Rayleigh quotient corresponding to  $z$ . If it is expensive to vary  $\kappa$  in solving

| $\ r\ $ | Case 1  | $\ r\ $ | Case 2  | $\tan \theta_1$ | $\tan \theta_2$ |
|---------|---------|---------|---------|-----------------|-----------------|
| 3.0e-01 | 3.0e-01 | 4.4e-01 | 1.3e+01 |                 |                 |
| 1.3e-02 | 1.3e-02 | 8.5e-02 | 2.4e+00 |                 |                 |
| 3.1e-03 | 3.7e-03 | 1.2e-02 | 2.6e-01 |                 |                 |
| 8.0e-04 | 2.2e-03 | 1.4e-03 | 5.2e-02 |                 |                 |
| 1.0e-04 | 1.5e-03 | 1.7e-04 | 1.5e-02 |                 |                 |
| 1.1e-05 | 9.7e-04 | 3.1e-05 | 5.9e-03 |                 |                 |
| 2.0e-06 | 6.5e-04 | 5.5e-06 | 2.0e-03 |                 |                 |
| 3.8e-07 | 4.3e-04 | 6.5e-07 | 4.9e-04 |                 |                 |
| 4.3e-08 | 2.9e-04 | 8.2e-08 | 1.1e-04 |                 |                 |
| 6.5e-09 | 1.9e-04 | 7.9e-09 | 1.9e-05 |                 |                 |
| 6.0e-10 | 1.3e-04 | 5.3e-10 | 2.1e-06 |                 |                 |
| 4.4e-11 | 8.5e-05 | 2.9e-11 | 2.0e-07 |                 |                 |
| 2.5e-12 | 5.7e-05 | 1.3e-12 | 1.5e-08 |                 |                 |
| 1.1e-13 | 3.8e-05 | 5.5e-14 | 1.0e-09 |                 |                 |
| 4.7e-15 | 2.5e-05 | 2.2e-15 | 6.4e-11 |                 |                 |
| 2.1e-16 | 1.7e-05 | 7.6e-16 | 6.1e-12 |                 |                 |

Figure 2.1: Convergence of the Jacobi–Davidson method

---

the correction equation, it will generally be chosen equal to  $\tau$ . However, if it is inexpensive to vary  $\kappa$  and  $\tilde{A} = A$ , then by the analysis of the last section the choice  $\kappa = \mu$  will give quadratic convergence — provided something does not go wrong with the Rayleigh–Ritz procedure.

The following example shows that the Rayleigh–Ritz refinement can improve the convergence of the approximate Newton method.

**Example 2.4.** As in Example 3.1, Chapter 4, we begin with a diagonal matrix  $A$  with eigenvalues

$$1, .95, .95^2, .95^3, \dots, .95^{99}.$$

The Jacobi–Davidson algorithm was used to approximate the eigenvector  $e_1$ . A starting vector  $u$  was obtained by normalizing a vector of the form  $e_1 + 0.05e$ , where  $e$  is a vector of random normal deviates. Two cases were considered: Case 1 in which the Rayleigh–Ritz step was included and Case 2 in which it was not. To slow convergence so that the process has time to evolve,  $\kappa$  was set to  $\mu + 0.1$ .

The table in Figure 2.1 exhibits the results. The first column shows the convergence of the residual norm in Case 1; the second, in Case 2. It is seen that the Jacobi–Davidson method is converging faster than the ordinary approximate Newton method. The convergence ratios for the first case decrease from about 0.24 to 0.041, while for the second case they increase from a value of 0.28 to 0.67. (All this excludes the first step, which is atypical.)

The third column shows the tangents of the angles between the correction space and the eigenvector  $e_1$ . They track the residual norms rather closely and show that it is the Rayleigh–Ritz approximation that is driving the convergence.

The example actually shows more. The last column exhibits the tangents of the angles between the correction subspace and the subdominant eigenvector  $e_2$ . If we decide at the last iteration to declare the iteration has produced a satisfactory approximation to  $e_1$  and start looking for approximations to  $e_2$ , we will begin with a very accurate starting vector.

### Extending Schur decompositions

We now turn to implementation issues. The first issue is the matter of deflation. Algorithm 2.1 is designed to find a single eigenpair. When more than one eigenpair is to be computed, we must take care that the iteration does not reconverge to a vector that has already been found; i.e., we must restrict the iteration to proceed outside the space spanned by the previous eigenvectors. Unfortunately, we cannot say just how the iteration should be restricted without knowing the unknown eigenvectors. A cure for this problem is to work with Schur vectors. If we have found a set  $u_1, \dots, u_p$  of Schur vectors, we know that the other Schur vectors lie in the orthogonal complement of  $\text{span}(u_1, \dots, u_p)$ . Thus, the restriction can be cast in terms of orthogonal projections involving known vectors.

Let us begin by assuming that we have an orthonormal matrix  $U$  of Schur vectors of  $A$ ; that is,  $U$  satisfies the relation

$$AU = UT \quad (2.13)$$

for some upper triangular matrix  $T = U^H AU$ . We wish to extend this *partial Schur decomposition* to a partial Schur decomposition of order one greater. An algorithm to do this would allow us to build up ever larger partial Schur decompositions a step at a time.

We begin by observing that we can extend (2.13) by mimicking the reduction to Schur form. Specifically, let  $(U \ U_\perp)$  be unitary, and write

$$\begin{pmatrix} U^H \\ U_\perp^H \end{pmatrix} A(U \ U_\perp) = \begin{pmatrix} T & H \\ 0 & B \end{pmatrix}.$$

Now suppose  $(\mu, z)$  is a normalized eigenpair of  $B$ , and let  $(z \ Z_\perp)$  be unitary. Then

$$\begin{pmatrix} z^H \\ Z_\perp^H \end{pmatrix} B(z \ Z_\perp) = \begin{pmatrix} \mu & g^H \\ 0 & C \end{pmatrix}$$

is block upper triangular. It follows that

$$\begin{pmatrix} U^H \\ z^H U_\perp^H \\ Z_\perp^H U_\perp^H \end{pmatrix} A(U \ U_\perp z \ U_\perp Z_\perp) = \begin{pmatrix} T & HU_\perp z & HU_\perp Z_\perp \\ 0 & \mu & g^H \\ 0 & 0 & C \end{pmatrix}.$$

Hence,

$$A(U \ U_{\perp} z) = (U \ U_{\perp} z) \begin{pmatrix} T & HU_{\perp} z \\ 0 & \mu \end{pmatrix} \quad (2.14)$$

is a partial Schur decomposition of  $A$  of order one higher than (2.13).

This typically dense matrix process is not much use in the large sparse problems that are our present concern. However, the following observation allows us to recast the algorithm in terms of the original matrix  $A$ . Let

$$P_{\perp} = U_{\perp} U_{\perp}^H = I - UU^H \equiv I - P$$

be the orthogonal projector onto the column space of  $U_{\perp}$ . Then it is easily verified that

$$A_{\perp} = P_{\perp} A P_{\perp} = U_{\perp} B U_{\perp}^H.$$

Now suppose that  $(\mu, z)$  is an eigenpair of  $B$ , and let  $y = U_{\perp} z$ . Then  $(\mu, y)$  satisfies

1.  $A_{\perp} y = \mu y$ ,
  2.  $y \perp U$ .
- (2.15)

Conversely, suppose  $y$  satisfies (2.15). Then  $y = U_{\perp} z$ , where  $z = U_{\perp}^H y$ . Since  $\mu y = A_{\perp} y = U_{\perp} B z$ , it follows on multiplying by  $U_{\perp}^H$  that  $B z = \mu z$ . In other words there is a one-one correspondence to eigenvectors of  $A_{\perp}$  lying in  $\mathcal{R}(U_{\perp})$  and eigenvectors of  $B$ .

Note that a knowledge of the eigenvector  $y$  is all that is necessary to extend the partial Schur decomposition. For it follows from (2.14) that the extended partial Schur decomposition is

$$A(U \ y) = (U \ y) \begin{pmatrix} T & t \\ 0 & \mu \end{pmatrix}, \quad (2.16)$$

where

$$t = U^H A y. \quad (2.17)$$

To compute a vector  $y$  satisfying (2.15), we will use the Jacobi–Davidson procedure. The ingredients for a step of the procedure are the matrix  $U$  of Schur vectors, an orthonormal matrix  $V$  of orthogonalized corrections, and a new correction vector  $v$ . We will assume that  $V$  and  $v$  are orthogonal to  $\mathcal{R}(U)$ .

We begin by orthogonalizing  $v$  with respect to  $V$  and incorporating it into  $V$ . We next compute the Rayleigh quotient  $B = V^T A_{\perp} V$  and a Ritz pair  $(\mu, z)$ . Note that because  $V$  is orthogonal to  $\mathcal{R}(U)$ , the Rayleigh quotient can also be computed from the original matrix  $A$  in the form  $V^T A V$ .

We now compute the residual

$$r = A_{\perp} z - \mu z$$

and test to see if it is small enough that  $(\mu, z)$  can be accepted as our eigenpair. There is a subtle point here. What we really want is to get a satisfactorily small residual when we substitute  $z$  for  $y$  in (2.16) and (2.17)—that is, we want

$$s = Az - Ut - \mu z$$

to be sufficiently small. To assess the size of  $s$ , we compute the size of its components in  $U$  and  $U_{\perp}$ . First, multiplying  $s$  by  $U^H$ , we get

$$U^H s = U^H A z - t - \mu U^H z = t - t = 0.$$

Hence  $P_s = 0$ . On the other hand, if we multiply  $s$  by  $P_{\perp} = U_{\perp} U_{\perp}^H$ , we get

$$\begin{aligned} P_{\perp} s &= P_{\perp} A z - P_{\perp} U t - \mu P_{\perp} z \\ &= P_{\perp} A P_{\perp} z - \mu z \\ &= A_{\perp} z - \mu z = r. \end{aligned}$$

It follows that  $\|s\|_2 = \|r\|_2$ , so that we may use  $r$  in a convergence test.

Finally, we solve the correction equation

$$(I - zz^H)A_{\perp}(I - zz^T)v = r,$$

where  $v \perp U, z$ , to get another correction to add to  $V$ .

Algorithm 2.2 implements this scheme. There are several comments to be made.

- The routine *EPSD* (Extend Partial Schur Decomposition) is designed to be used by a higher-level program that decides such things as the number of Schur vectors desired, the focal point  $\tau$  (which may vary), and how to restart when the arrays  $V$  and  $W$  are about to overflow storage.
- Excluding operations with  $A$  (e.g., in statements 4 and 15), the storage requirements are up to  $2nm$  floating-point numbers.
- Again excluding operations with  $A$ , the main floating-point operations come from matrix-vector products such as  $Vp$ . Thus if  $U$  is not too large and the algorithm returns at  $k = \hat{m} \leq m$ , we can expect the operation count to be a multiple of  $n(\hat{m}-\ell)^2$  flam.
- As  $V$  grows, it is not necessary to recompute the entire Rayleigh quotient  $C$ —only its southeast boundary. This is done in statement 5.
- In principle, we do not have to orthogonalize  $v$  against  $U$  in statement 3, since by definition  $U^H v = 0$ . However, if there is cancellation when  $v$  is orthogonalized against  $V$ , the results will no longer be orthogonal to  $U$  to working accuracy. Thus  $v$  should be orthogonalized against both  $U$  and  $V$ —with reorthogonalization if necessary.
- We do not compute the residual  $r$  directly. Rather we compute  $Az - \mu z$  and orthogonalize it against  $U$ .

Given a partial Schur decomposition  $AU = UT$ , EPSD extends it to a partial Schur decomposition of the form

$$A(U \ u) = (U \ u) \begin{pmatrix} T & t \\ 0 & \mu \end{pmatrix}.$$

The inputs, in addition to  $U$ , are  $V$ , an  $n \times (\ell - 1)$  matrix of orthonormalized correction vectors; an additional correction vector  $v$ ;  $W = AV$ ,  $C = W^H V$ ; a focal point  $\tau$ ; and an approximation  $\tilde{A} = A + E$ . In the following code,  $P_\perp = I - UU^H$ , and  $\tilde{A}_\perp = P_\perp \tilde{A} P_\perp$ . It is assumed that the arrays  $V$  and  $W$  cannot be expanded to have more than  $m$  columns. If convergence does not occur before that happens, EPSD returns the next correction vector  $v$ .

1.  $\text{EPSD}(U, V, v, W, C, \tau, \ell, m)$
2. **for**  $k = \ell$  **to**  $m$
3.     Orthonormalize  $v$  with respect to  $U$  and  $V$
4.      $w = Av$
5.      $C = \begin{pmatrix} C & V^H * w \\ v^H * w & v^H * W \end{pmatrix}$
6.      $V = (V \ v)$ ;  $W = (W \ w)$
7.     Find an eigenpair  $(\mu, p)$  of  $C$  such that  $\mu$  is nearest  $\tau$
8.      $z = Vp$
9.      $y = Wp$
10.     $r = y - \mu z$
11.    Orthogonalize  $r$  with respect to  $U$
12.    **if** ( $\|r\|$  is sufficiently small)
13.      **return**  $u = z$ ,  $t = U^H y$ , and  $\mu$
14.    **end if**
15.    Choose a shift  $\kappa$  and solve the correction equation  

$$(I - zz^T)(\tilde{A}_\perp - \kappa I)(I - zz^T)v = r, \quad v \perp U, z$$
16.   **end for**  $k$
17.   **return**  $v$
18. **end EPSD**

**Algorithm 2.2:** Extending a partial Schur decomposition



- As mentioned above, the natural choice of  $\kappa$  is either  $\tau$  or  $\mu$ . The later choice will give asymptotically swifter convergence; but since  $\mu$  varies with  $k$ , it may make the correction equation harder to solve — requiring, for example, a matrix factorization at each step.

### Restarting

Like Krylov sequence methods, the Jacobi–Davidson algorithm builds up a basis  $V$  from which eigenvectors can be extracted by the Rayleigh–Ritz method. Ultimately,  $V$  will become so large that it exceeds the storage capacity of the machine in question. At that point we must restart with a reduced basis. Unlike Krylov sequence methods, the Jacobi–Davidson method does not have to preserve any special relation in the new basis — we are free to restart in any way we desire.

It is possible, however, to abuse that freedom. The basis  $V$  contains hard-won information about the vectors near the focal point, information that should not be discarded in the process of restarting. Fortunately we can extract that information by using a Schur–Rayleigh–Ritz process. Specifically, given  $V$ , a focal point  $\tau$ , and a radius  $\rho$ , we may proceed as follows. Here we use the notation of Algorithm 2.2.

1. Compute the Schur decomposition  $C = QTQ^H$ , where the diagonals of  $T$  are ordered so that  $|t_{ii} - \tau| \leq |t_{i+1,i+1} - \tau|$
  2. Determine the largest integer  $\ell$  such that  $|t_{\ell\ell} - \tau| \leq \rho$
  3.  $V = V * Q[:, 1:\ell]$
  4.  $W = W * Q[:, 1:\ell]$
  5.  $C = Q[:, 1:\ell]^H * C * Q[:, 1:\ell]$
- (2.18)

Otherwise put, we replace  $V$  with a basis for the primitive Ritz subspace corresponding to the Ritz values within  $\rho$  of  $\tau$ .

### Harmonic Ritz vectors

In §4.4, Chapter 4, we noted that an interior eigenvalue could be approximated by a Rayleigh quotient of a vector that has nothing to do with the eigenvalue in question. If such a vector is accepted as a Ritz vector, the Rayleigh–Ritz method may fail. We can, of course, detect these imposters by the fact that they have large residuals. But it is better to have a good vector in hand than to know that a vector is bad.

An alternative is to use the harmonic Rayleigh–Ritz procedure, which discourages imposters. Recall (Definition 4.11, Chapter 4) that  $(\tau + \delta, Vp)$  is a harmonic Ritz pair with orthonormal basis  $V$  and shift  $\tau$  if

$$V^H(A - \tau I)^H(A - \tau I)Vp = \delta V^H(A - \tau I)Vp.$$

Moreover, if  $WR$  is the QR factorization of  $(A - \tau I)U$ , then

$$(W^H U)p = \delta^{-1} R p$$

so that we can calculate  $\delta$  and  $p$  from this simpler and numerically more stable generalized eigenproblem. Note that the matrix  $B = W^H U$  can be updated as we bring in new correction vectors  $v$ .

Having computed a primitive harmonic Ritz pair  $(p, \tau + \delta)$ , we cannot use the harmonic eigenvalue in computing the residual, since that value will not make the residual orthogonal to  $V$ . Instead we must use the ordinary Rayleigh quotient  $p^H V A V p$ . In §4.4, Chapter 4, we showed that

$$p^H V A V p = z^H W R p + \tau,$$

where  $z = V p$  is the harmonic Ritz vector. A little algebra then shows that

$$A z - (z^H A z) z = W R p - (z^H W R p) z.$$

Algorithm 2.3 implements the harmonic version of partial Schur extension. Although it is more complicated than Algorithm 2.2, it has comparable storage requirements and operation counts.

The algorithm for restarting the harmonic extension algorithm is more complicated than Algorithm 2.2 owing to the need to keep the columns of  $W$  orthogonal. Specifically, we have the following algorithm.

1. Let  $(Q S Z^H, Q T Z^H)$  be the generalized Schur decomposition of the pencil  $(B, R)$ , where the eigenvalues  $\mu_i = s_{ii}/t_{ii}$  are ordered so that  $|\mu_i - \tau| \leq |\mu_{i+1} - \tau|$
2. Determine the largest integer  $\ell$  such that  $|\mu_\ell - \tau| \leq \rho$
3.  $V = V * Q[:, 1:\ell]$
4.  $G = R * Q[:, 1:\ell]$
5. Let  $H = K R$  be the QR factorization of  $G$
6.  $W = W * K$
7.  $B = K^H * B * Q[:, 1:\ell]$

Note the indirect way of updating the QR factorization of the restarted matrix

$$A V Q[:, 1:\ell] = W R Q[:, 1:\ell].$$

It is faster than simply forming  $W R Q[:, 1:\ell]$  and computing its QR factorization.

### Hermitian matrices

When  $A$  is Hermitian some obvious simplifications occur in Algorithm 2.2 (*EPSD*). For example, the Rayleigh quotient  $C$  is Hermitian, so that only the upper half of  $C$  need be retained. The Schur form is diagonal, so that *EPSD* need only return the converged Ritz pair (statement 13). Similarly economies can be achieved in the restarting procedure (2.18).

The principle change in the harmonic extension (Algorithm 2.3), other than the natural economies listed in the last paragraph, is in how the harmonic Ritz vectors are

Given a partial Schur decomposition  $AU = UT$ , *HEPSD* extends it to a partial Schur decomposition of the form

$$A(U \ u) = (U \ u) \begin{pmatrix} T & t \\ 0 & \mu \end{pmatrix},$$

using harmonic Ritz vectors. The inputs, in addition to  $U$ , are  $V$ , an  $n \times (\ell - 1)$  matrix of orthonormalized correction vectors; an additional correction vector  $v$ ; the QR factorization  $WR$  of  $(A - \tau I)V$ ; the current harmonic pair  $(B, R)$ ; a focal point  $\tau$ ; and an approximation  $\tilde{A} = A + E$ . In the following code  $P_\perp = I - UU^H$ , and  $A_\perp = P_\perp AP_\perp$ . It is assumed that the arrays  $V$  and  $W$  cannot be expanded to have more than  $m$  columns. If convergence does not occur before that happens, *HEPSD* returns the next correction vector  $v$ .

1. *HEPSD*( $U, V, v, W, R, B, \tau, \ell, m$ )
2.   **for**  $k = \ell$  **to**  $m$
3.     Orthonormalize  $v$  with respect to  $U$  and  $V$
4.      $y = A*v - \tau v$
5.     *gsreorthog*( $W, y, w, r, \rho$ )
6.      $R = \begin{pmatrix} R & r \\ 0 & \rho \end{pmatrix}$
7.      $B = \begin{pmatrix} B & W^H * v \\ w^H * V & w^H * v \end{pmatrix}$
8.      $V = (V \ v); W = (W \ w)$
9.     Find an eigenpair  $(\delta, p)$  of  $Bp = \delta^{-1}Rp$  such that  $\delta + \tau$  is nearest  $\tau$
10.     $z = Vp$
11.     $y = W*(R*p); \mu = z^H y$
12.     $r = y - \mu z$
13.    Orthogonalize  $r$  with respect to  $U$
14.    **if** ( $\|r\|$  is sufficiently small)
15.      **return**  $u = z, t = U^H y$ , and  $\mu$
16.    **end if**
17.    Choose a shift  $\kappa$  and solve the correction equation  

$$(I - zz^T)(A_\perp - \kappa I)(I - zz^T)v = r,$$
where  $v \perp U, z$
18.   **end for**  $k$
19.   **return**  $v$
20. **end HEPsd**

**Algorithm 2.3:** Harmonic extension of a partial Schur decomposition

computed. As noted in §4.4, Chapter 4, when  $A$  is Hermitian, the nonsymmetric eigenproblems (4.22) and (4.23), Chapter 4, can give complex harmonic Ritz pairs. We should therefore use the Hermitian eigenproblem

$$(W^H V R^{-1})(Rp) = \delta^{-1}(Rp) \quad (2.19)$$

to compute harmonic Ritz pairs, both for residuals for the correction equation and for restarting.

### Solving projected systems

We now turn to the problem of solving the correction equation

$$(I - zz^H)(I - UU^H)(A - \kappa I)(I - UU^H)(I - zz^H)v = r, \quad v, r \perp z, U.$$

It will pay to consider this problem in greater generality than just the correction equation. Therefore, we will consider the general system

$$(I - QQ^H)M(I - QQ^T)x = y, \quad x, y \perp Q, \quad (2.20)$$

in which we will assume  $M$  is nonsingular.

To derive the solution of this system, we will assume that a solution  $x$  exists. Since  $x \perp Q$ , we have

$$(I - QQ^H)Mx = y,$$

or

$$Mx = y - QQ^H Mx \equiv y - Qa.$$

Hence,

$$x = M^{-1}y - M^{-1}Qa \equiv \hat{x} - \hat{Q}a. \quad (2.21)$$

Since  $Q^H x = 0$ , we have

$$(Q^H \hat{Q})a = Q^H \hat{x}, \quad (2.22)$$

a linear system from which we can compute  $a$ . It is easy to verify that if  $x$  is defined by (2.21), where  $a$  is a solution of (2.22), then  $x$  satisfies (2.20).

Algorithm 2.4 implements this scheme. Here are three comments.

- Statement 1 anticipates the use of this algorithm to solve several systems. If it is expensive to solve systems involving the matrix  $M$ , then the bulk of the work in the algorithm will be concentrated in the formation of  $\hat{Q} = M^{-1}Q$ . If  $C$  has been precomputed, it can be used whenever another solution involving  $M$  and  $Q$  is required.

This algorithm solves the projected equation

$$(I - QQ^H)M(I - QQ^T)x = y, \quad x, y \perp Q,$$

where  $Q$  is orthonormal, and  $M$  is nonsingular.

1. Precompute or update  $\hat{Q} = M^{-1}Q$  and  $C = Q^H\hat{Q}$
2. Solve the system  $M\hat{x} = y$
3.  $b = Q^H\hat{x}$
4. Solve the system  $Ca = b$
5.  $x = \hat{x} - Qa$

#### Algorithm 2.4: Solution of projected equations

---

Likewise, if  $Q$  is augmented by a single vector  $z$ , then one can solve the system  $M\hat{z} = z$  and update  $C$  in the form

$$\begin{pmatrix} C & Q^H\hat{z} \\ z^H\hat{Q} & z^H\hat{z} \end{pmatrix}.$$

In both cases there is a great savings in work. However, we must allocate additional storage to contain  $\hat{Q}$  — at least when  $M$  is nonsymmetric. For more see the discussion on economization of storage on page 391.

- We could go further and precompute or update a factorization of  $C$  by which we could calculate the solution of the system in statement 4. However, if  $n$  is very large, the savings will be insignificant. But if a factorization is to be updated, it should be one that can be stably updated without pivoting — e.g., a QR decomposition.
- If  $M$  is ill conditioned but  $(I - QQ^H)M(I - QQ^T)$  is not, then the vector  $x$  will not be large, while the vector  $\hat{x}$  can be very large. In this case, we can expect cancellation in statement 5. Even when there is no cancellation,  $\hat{x}$  is likely to have inaccuracies that will propagate to the solution  $x$ .

#### The correction equation: Exact solution

We now return to the solution of the correction equation

$$(I - zz^H)(I - UU^H)(A - \kappa I)(I - UU^H)(I - zz^H)v = r, \quad v, r \perp z, U.$$

Since  $z \perp U$ , we can set  $Q = (U \ z)$  and use Algorithm 2.4 to solve the correction equation.

The mechanics of the algorithm argue strongly against changing the shift  $\kappa$  frequently. In the first place, the algorithm requires that we solve systems of the form

$(A - \kappa I)\hat{x} = y$ . This requires a factorization of  $A$ , which has to be recomputed whenever  $\kappa$  changes. Second, if  $\kappa$  does not change, we can precompute the matrix  $C$ , and update it by adding the vector  $z$  as described above. All this suggests that we use the focal point  $\tau$  in Algorithms 2.2 and 2.3 as the shift  $\kappa$ .

Our comments on the accuracy of Algorithm 2.4 suggest that we should avoid ill conditioning in  $A - \kappa I$ . This means we should avoid a focal point  $\tau$  that accurately approximates an eigenvalue  $\lambda$  of  $A$ . Even though such a focal point will speed up convergence to  $\lambda$ , it could retard convergence to other eigenpairs.

### The correction equation: Iterative solution

An alternative to the exact solution of the correction equation is to use an iterative method to provide an approximate solution. The subject of iterative methods is a large one—far beyond the scope of this volume. Fortunately, to describe how iterative solutions are used in the Jacobi–Davidson method, we need only know the operations we are required to perform with the matrix of our system.

Let us consider a general linear system  $Ku = b$ . The most successful of the various iterative methods for solving this system are based on Krylov sequences. This means that we only need to be able to form the matrix–vector product  $Kw$  for an arbitrary vector  $w$ . In the Jacobi–Davidson method, the matrix is  $(I - QQ^H)(A - \kappa I)(I - QQ^H)$ , where  $Q = (U \ z)$ . Since the Jacobi–Davidson algorithm already requires that we be able to form matrix–vector products in  $A$ , it is easy to use Krylov sequence iterations with it.

But that is not the whole story. The convergence of most Krylov sequence methods can be improved, often dramatically, by a process called *preconditioning*. The idea is to determine a matrix  $M$ —usually based on special characteristics of the problem in question—such that  $M^{-1}K$  is in some sense near the identity matrix. If it is cheap to solve systems in  $M$ , we can form the matrix–vector product  $z = M^{-1}Kw$  by the following algorithm.

1.  $y = Kw$
  2. Solve the system  $Mz = y$
- (2.23)

In application to the Jacobi–Davidson algorithm, the matrix  $K$  is, as usual,  $(I - QQ^H)(\tilde{A} - \kappa I)(I - QQ^H)$ . The preconditioner  $M$  is determined from the matrix  $A - \kappa I$ ; however, since the preconditioner must approximate the projected system, it itself must be projected. Thus our equivalent of (2.23) is the following algorithm.

1.  $y = (I - QQ^H)(A - \kappa I)(I - QQ^H)w$
2. Solve the system  $(I - QQ^H)M(I - QQ^H)z = y$ , where  $z, y \perp Q$

The system in statement 2 can be solved by Algorithm 2.4.

The use of a preconditioned solver gives us more freedom to choose a shift  $\kappa$ . First, the system to be solved involves the preconditioner, not the shifted matrix. Thus, if we can find an effective preconditioner that does not depend on  $\kappa$ , we can vary  $\kappa$ —at least within the limits for which the preconditioner is effective. Second, we have more

| $\ r\ $ | $\tan \theta_1$ | $\tan \theta_2$ |
|---------|-----------------|-----------------|
| 3.6e-01 | 5.8e-01         | 7.3e+01         |
| 8.0e-02 | 5.4e-01         | 6.2e+01         |
| 6.0e-02 | 8.0e-02         | 2.4e+01         |
| 9.1e-03 | 1.0e-02         | 2.0e+01         |
| 1.1e-03 | 2.3e-03         | 9.3e+00         |
| 1.9e-04 | 9.0e-04         | 2.7e+00         |
| 5.4e-05 | 3.5e-04         | 7.7e-01         |
| 2.3e-05 | 8.0e-05         | 2.4e-01         |
| 8.0e-06 | 1.3e-05         | 8.1e-02         |
| 1.5e-06 | 1.7e-06         | 4.6e-02         |
| 2.0e-07 | 2.2e-07         | 4.3e-02         |
| 2.6e-08 | 3.2e-08         | 4.2e-02         |
| 3.6e-09 | 5.2e-09         | 4.1e-02         |
| 5.1e-10 | 1.0e-09         | 4.1e-02         |
| 9.9e-11 | 1.9e-10         | 4.1e-02         |
| 2.1e-11 | 3.1e-11         | 4.1e-02         |

Figure 2.2: Convergence of the Jacobi–Davidson method with errors in the correction equations

---

freedom to place  $\kappa$  near an eigenvalue, since we work only with the projected operator  $(I - QQ^H)(A - \kappa I)(I - QQ^H)$  and never have to explicitly solve systems involving  $A - \kappa I$ .

Since we are free to stop an iterative method at any time, we can choose to solve the correction equation more or less accurately. Unfortunately, there is very little hard analysis to guide us in this matter. There is a tension between the local convergence driven by the approximate Newton iteration and the broader convergence driven by the correction subspace. From a local point of view, only a certain number of figures of  $v$  actually contribute to the improvement of  $z$ , and there is no reason to solve the correction equation more accurately than to that number of figures.

However, this lack of accuracy can exact a price when we come to compute other eigenpairs. The following example illustrates this phenomenon.

**Example 2.5.** The data is the same as in Example 2.4 (which the reader should review). The only difference is that at each stage a normwise relative error of 0.01 is introduced in the correction equation. The table in Figure 2.2 shows the progress of the iteration (for a different random starting vector). The convergence is satisfactory, though not as fast as in Example 2.4. However, the approximations to  $e_2$  in the correction space stagnate after an initial decline.

If we decrease the error in the solution to the correction equation in the above ex-

ample to  $10^{-3}$  and  $10^{-4}$ , the convergence to  $e_1$  is marginally faster, but the convergence of approximations to  $e_2$  stagnates at  $5 \cdot 10^{-3}$  and  $5 \cdot 10^{-4}$ , suggesting that the approximations to vectors other than the current vector will stagnate at about the level of the error in the solution of the correction equations. More generally, when the approximate Newton method is used, any convergence of the correction subspace should be toward eigenvectors of  $\tilde{A} = A + E$ , so that approximations to eigenvectors of  $A$ , other than the one currently being found, should stagnate at the level of  $\|E\|$ .

### 2.3. NOTES AND REFERENCES

#### Newton's method and the eigenvalue problem

According to Kollerstrom [150] and Ypma [307] neither Newton nor Raphson used the calculus in the derivation of their methods. Newton, working with a polynomial equation  $p(x) = 0$ , expanded  $p(x + d) = q(d)$  and ignored higher terms in  $d$  to get a linear equation for  $d$ . He then continued by expanding  $q(d + e) = r(e)$ , linearizing, and solving for  $e$ . The scheme involved explicitly computing the polynomials  $p, q, r$ , etc.—obviously a computationally expensive procedure. (However, Newton was actually interested in computing series expansions for  $x$ , in which case the extra work in computing the polynomials is justified.) Simpson showed how to effect the same calculations by working only with  $p$ . The formulation in terms of derivatives is due to Simpson. It is ironic that although Newton cannot be credited with the formula that bears his (and Raphson's) name, he seems to have been the originator of the idea expanding and linearizing to compute corrections to zeros of functions—just the approach we use here.

Many people have applied Newton's method to the eigenvalue problem. For some references, see [4]. The formula (2.3), with  $w = z$  is due to Peters and Wilkinson [211]. They justify the choice  $w = z$  by observing that if we require  $z$  and  $z + w$  to be normalized, then up to second-order terms in  $w$  we must have  $w^T z = 0$ .

Peters and Wilkinson [211] explore in detail the relation of Newton's method to the inverse power method. The fact that asymptotically Newton's method and the Rayleigh quotient method produce essentially the same iterates can be established by a theorem of Denis and Schnabel [64] to the effect that any two superlinearly converging methods launched from the same point must up to higher-order terms produce the Newton direction.

The relation of Newton's method to the RQ variant of the QR algorithm is best seen by considering the RQ factorization of  $A - \kappa I$  in the  $Z$  coordinate system. Specifically, partitioning the R- and Q-factors, we must have

$$\begin{pmatrix} \mu - \kappa & h^H \\ g & B - \kappa I \end{pmatrix} \begin{pmatrix} 1 & q^H \\ p & Q \end{pmatrix} = \begin{pmatrix} R & r \\ 0 & \rho \end{pmatrix}$$

(here we are ignoring second-order terms in the normalization of the first row and column of the Q-factor). It then follows that

$$g + (B - \kappa I)p = 0,$$

This algorithm takes a symmetric diagonally dominant matrix  $A = D + E$ , where  $D$  is the diagonal of  $A$  and an orthonormal matrix  $V$  of correction vectors and returns a Ritz pair  $(\mu, z)$  approximating the  $k$ th smallest eigenvalue of  $A$  and an augmented correction matrix  $\hat{B}$ . Here  $P_\perp$  is the projection onto  $\mathcal{R}(V)^\perp$ .

1. Using  $V$ , compute the Ritz pair  $(\mu, z)$  where  $\mu$  is the  $k$ th smallest eigenvalue of the Rayleigh quotient
2.  $r = Az - \mu z$
3. Solve the system  $(D - \mu)v = r$
4.  $v = P_\perp v / \|P_\perp v\|_2$
5.  $\hat{V} = (V \ v)$

### Algorithm 2.5: Davidson's method

---

so that up to second-order terms the correction from the Q-factor and the Newton correction are the same.

### Inexact Newton methods

In 1982, Dembo, Eisenstat, and Steihaug [55] introduced and analyzed inexact Newton methods, in which the Newton equation is solved inexactly. In some sense these methods are related to the approximate Newton methods introduced here, since any residual in the approximate solution of the Newton equation can be thrown back onto the Jacobian and conversely any perturbation of the Jacobian in the Newton equation corresponds to an approximate solution with a nonzero residual. However, it differs from the approximate Newton method given here in that we are working with a special constrained problem and the approximation is not to the Jacobian itself. To honor these distinctions we use the term “approximate” rather than “inexact” Newton method.

### The Jacobi–Davidson method

As it has been derived here, the Jacobi–Davidson method might better be called the Newton–Rayleigh–Ritz method, and it is instructive to see how the current name came about. The story starts with Davidson's algorithm [53, 1975], which is sketched in Algorithm 2.5. It is very close to Algorithm 2.1 — it builds up a correction subspace, from which it computes a Ritz vector and a new correction that is folded back into the subspace. The chief difference is in the correction equation, which is essentially an approximate inverse power method. For diagonally dominant matrices the approximation is quite good, so that the asymptotic relation between the inverse power method and Newton's method insures that Davidson's method will work well. But in other, less favorable problems it tends to stagnate.

In 1990, Olsen [186] and his coauthors observed that the lack of orthogonality of  $v$  to  $z$  was a drawback. Using first-order perturbation expansions, they rediscovered

the matrix formulation of Peters and Wilkinson, but with an approximation as in (2.3), and solved it to get the correction formula (2.6). Thus Olsen's method is Davidson's method with an approximate Newton correction. It is worth noting that the author's application was so large that they could not save the corrections, and hence what they actually used was the approximate Newton method.

In 1996, Sleijpen and van der Vorst [240] derived essentially the same algorithm by considering an algorithm of Jacobi [128, 1846] for approximating eigenvectors of diagonally dominant matrices. Specifically, Jacobi partitioned a dominant symmetric matrix  $A$  in the form

$$A = \begin{pmatrix} \mu & g^T \\ g & D + E \end{pmatrix}.$$

Assuming an approximate eigenvector in the form  $e_1$ , he computes a refined eigenvector of the form

$$\begin{pmatrix} 1 \\ -(D + E)^{-1}g \end{pmatrix}.$$

This is easily seen to be one step of the approximate Newton method with a diagonal (Jacobi) approximation. Sleijpen and van der Vorst extended this example to general matrices, thus deriving the orthogonal Newton step. From the point of view taken here, their major contribution was the introduction of the correction equation, which expresses the correction in terms of a projection of the original matrix. It is the key to making the method effective for large sparse matrices. In particular, they went on to show how to deflate the problem and to incorporate iterative solutions with preconditioning. In later papers [74, 239, 245] Sleijpen, van der Vorst, and their colleagues also adapted the algorithm to harmonic Ritz vectors and the generalized eigenvalue problem. Templates may be found in [241, 242, 243, 244].

### **Hermitian matrices**

The implementation for harmonic Ritz vectors suggested here differs somewhat from the implementation in [243]. There the authors accumulate  $R^{-1}$  in the basis  $V$ . We have preferred to leave  $V$  orthogonal and work with the harmonic Rayleigh–Ritz equation (2.19).

# APPENDIX: BACKGROUND

The reader of this volume is assumed to know the basics of linear algebra and analysis as well as the behavior of matrix algorithms on digital computers. This background has been presented in full detail in Volume I of this work. Here we confine ourselves to a review of some of the more important notation and facts.

## 1. SCALARS, VECTORS, AND MATRICES

1. Scalars are real or complex numbers. The set of real numbers will be denoted by  $\mathbb{R}$ ; the set of complex numbers by  $\mathbb{C}$ . All scalars are denoted by lowercase Latin or Greek letters.
2. An  $n$ -vector is a collection of  $n$  scalars arranged in a column. The set of real  $n$ -vectors is denoted by  $\mathbb{R}^n$ ; the set of complex  $n$ -vectors by  $\mathbb{C}^n$ . Vectors are denoted by lowercase Latin letters.
3. The scalars that form a vector are called the components of the vector. If a vector is denoted by a letter, its components will be denoted by a corresponding lowercase Latin or Greek letter. Thus the vector  $a$  may be written

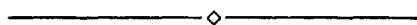
$$a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}.$$

Figure A.1 gives the Greek alphabet and its corresponding Latin letters (some of the correspondences are quite artificial).

4. An  $m \times n$  matrix is a rectangular array of  $mn$  scalars having  $m$  rows and  $n$  columns. The set of real  $m \times n$  matrices is denoted by  $\mathbb{R}^{m \times n}$ ; the set of complex  $m \times n$  matrices by  $\mathbb{C}^{m \times n}$ . Matrices are denoted by uppercase Latin and Greek letters.
5. The scalars in a matrix are called the elements of the matrix. If a matrix is denoted by a letter, its elements will be denoted by a corresponding lowercase Latin or Greek

|         |            |        |         |           |           |         |             |     |
|---------|------------|--------|---------|-----------|-----------|---------|-------------|-----|
| alpha   | $\alpha$   | $a$    | kappa   | $\kappa$  | $k$       | sigma   | $\varsigma$ | $q$ |
| beta    | $\beta$    | $b$    | lambda  | $\lambda$ | $l, \ell$ | tau     | $\tau$      | $t$ |
| gamma   | $\gamma$   | $c, g$ | mu      | $\mu$     | $m$       | upsilon | $v$         | $u$ |
| delta   | $\delta$   | $d$    | nu      | $\nu$     | $n, v$    | phi     | $\phi$      | $f$ |
| epsilon | $\epsilon$ | $e$    | xi      | $\xi$     | $x$       | chi     | $\chi$      |     |
| zeta    | $\zeta$    | $z$    | omicron | $\circ$   | $o$       | psi     | $\psi$      |     |
| eta     | $\eta$     | $y, h$ | pi      | $\pi$     | $p$       | omega   | $\omega$    | $w$ |
| theta   | $\theta$   |        | rho     | $\rho$    | $r$       |         |             |     |
| iota    | $\iota$    | $i$    | sigma   | $\sigma$  | $s$       |         |             |     |

Figure A.1: The Greek alphabet and Latin equivalents



letter. Thus the matrix  $A$  may be written

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \vdots & \vdots & & \vdots \\ \alpha_{m1} & \alpha_{m2} & \cdots & \alpha_{mn} \end{pmatrix}.$$

6. An  $n \times n$  matrix is said to be square of order  $n$ .
  7. The matrix sum and matrix product are written as usual as  $A + B$  and  $AB$ .
  8. The conjugate of a matrix  $A$  is written  $\bar{A}$ . The transpose of  $A$  is written  $A^T$ , and the conjugate transpose as  $A^H$ .
  9. If  $A$  is nonsingular, its inverse is written  $A^{-1}$ . We also write
- $$A^{-T} = (A^{-1})^T = (A^T)^{-1} \quad \text{and} \quad A^{-H} = (A^{-1})^H = (A^H)^{-1}.$$
10. The trace of a square matrix is the sum of its diagonal elements. It is written  $\text{trace}(A)$ .
  11. The determinant of  $A$  is written  $\det(A)$ . It has the following useful properties.
    1.  $\det(A^H) = \overline{\det(A)}$ .
    2. If  $A$  is of order  $n$ , then  $\det(\mu A) = \mu^n \det(A)$ .
    3.  $\det(AB) = \det(A)\det(B)$ .
    4.  $\det(A^{-1}) = \det(A)^{-1}$ .
    5. If  $A$  is block triangular with diagonal blocks  $A_{11}, A_{22}, \dots, A_{kk}$ , then

$$\det(A) = \det(A_{11})\det(A_{22})\cdots\det(A_{kk}).$$

6.  $\det(A)$  is the product of the eigenvalues of  $A$ .
  7.  $|\det(A)|$  is the product of the singular values of  $A$ .
  
  12. The absolute value of a matrix  $A$  is the matrix whose elements are the absolute value of the elements of  $A$ . It is written  $|A|$ .
  13. If  $A$  and  $B$  are matrices of the same dimension, we write  $A \leq B$  to mean that  $a_{ij} \leq b_{ij}$ . Similarly for the relations  $A < B$ ,  $A \geq B$ , and  $A > B$ .
  14. Any vector whose components are all zero is called a zero vector and is written  $0$ , or  $0_n$  when it is necessary to specify the dimension.
  15. Any matrix whose components are all zero is called a zero matrix and is written  $0$ , or  $0_{m \times n}$  when it is necessary to specify the dimension.
  16. The vector whose  $i$ th component is one and whose other components are zero is called the  $i$ th unit vector and is written  $e_i$ . The vector whose components are all one is written  $e$ .
  17. The matrix  $I_n$  whose diagonal elements are one and whose off-diagonal elements are zero is called the identity matrix of order  $n$ . When the context makes the order clear, an identity matrix is written simply  $I$ .
  18. If  $i_1, \dots, i_n$  is a permutation of the integers  $1, \dots, n$ , the matrix
- $$P = (e_{i_1} \ \cdots \ e_{i_n})$$
- is called a permutation matrix. Premultiplying a matrix by  $P$  rearranges its rows in the order  $i_1, \dots, i_n$ . Postmultiplying a matrix by  $P$  rearranges its columns in the order  $i_1, \dots, i_n$ .
19. The matrix
- $$F = (e_n \ e_{n-1} \ \cdots \ e_1)$$
- is called the cross operator. Premultiplying a matrix by the cross operator reverses the order of its rows. Postmultiplying a matrix by the cross operator reverses the order of its columns.
20. A matrix  $A$  is symmetric if  $A^T = A$ . It is Hermitian if  $A^H = A$ .
  21. A real matrix  $U$  is orthogonal if  $U^T U = I$ . A matrix  $U$  is unitary if  $U^H U = I$ .
  22. A matrix with orthonormal rows or columns is said to be orthonormal.
  23. A square matrix whose off-diagonal elements are zero is said to be diagonal. A diagonal matrix whose diagonal elements are  $\delta_1, \dots, \delta_n$  is written  $\text{diag}(\delta_1, \dots, \delta_n)$ .

24. A Wilkinson diagram is a device for illustrating the structure of a matrix. The following is a Wilkinson diagram of a diagonal matrix of order four:

$$\begin{pmatrix} X & 0 & 0 & 0 \\ 0 & X & 0 & 0 \\ 0 & 0 & X & 0 \\ 0 & 0 & 0 & X \end{pmatrix}.$$

Here the symbol “0” stands for a zero element. The symbol “X” stands for a quantity that may or may not be zero (but usually is not). Letters other than X are often used to denote such elements, but a zero is always 0 or 0.

25. Lower and upper triangular matrices have Wilkinson diagrams of the form

$$\begin{pmatrix} X & 0 & 0 & 0 \\ X & X & 0 & 0 \\ X & X & X & 0 \\ X & X & X & X \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} X & X & X & X \\ 0 & X & X & X \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \end{pmatrix}.$$

26. A rectangular matrix whose elements below the main diagonal are zero is upper trapezoidal. A rectangular matrix whose elements above the main diagonal are zero is lower trapezoidal.

27. Triangular or trapezoidal matrices whose main diagonal elements are one are unit triangular or trapezoidal.

## 2. LINEAR ALGEBRA

1. With the exception of the sets  $\mathbb{R}$ ,  $\mathbb{C}$ ,  $\mathbb{R}^n$ , etc., sets of scalars, vectors, or matrices are denoted by calligraphic letters—e.g.,  $\mathcal{X}$ .

2. The dimension of a subspace  $\mathcal{X}$  is written  $\dim(\mathcal{X})$ . The subspace spanned by a set of vectors  $\mathcal{X}$  is written  $\text{span}(\mathcal{X})$ . If  $\mathcal{X}$  and  $\mathcal{Y}$  are disjoint subspaces, then

$$\mathcal{X} \oplus \mathcal{Y} = \text{span}(\mathcal{X} + \mathcal{Y}) = \text{span}\{x + y : x \in \mathcal{X}, y \in \mathcal{Y}\}.$$

3. Let  $\mathbb{F}$  be  $\mathbb{R}$  or  $\mathbb{C}$  depending on the underlying scalars. The column space of a  $m \times n$  matrix  $A$  is the subspace  $\mathcal{R}(A) = \{Ax : x \in \mathbb{F}^n\}$ .

4. The null space of  $A$  is  $\mathcal{N}(A) = \{x : Ax = 0\}$ .

5. The rank  $A$  is

$$\text{rank}(A) = \dim(\mathcal{R}(A)) = \dim(\mathcal{R}(A^H)).$$

6. The nullity of  $A$  is

$$\text{null}(A) = \dim(\mathcal{N}(A)).$$

7. An  $n \times p$  matrix  $X$  is of full column rank if  $\text{rank}(X) = p$ . It is of full row rank if  $\text{rank}(X) = n$ . We say that  $X$  is of full rank when the context tells whether we are referring to column or row rank (e.g., when  $n \geq p$ ).

### 3. POSITIVE DEFINITE MATRICES

1. Let  $A$  be of order  $n$ . Then  $A$  is positive definite if

1.  $A$  is Hermitian
2.  $x \neq 0 \implies x^H A x > 0$ .

If  $x \neq 0 \implies x^H A x \geq 0$ , then  $A$  is positive semidefinite.

2. In the real case, some people drop the symmetry condition and call any matrix  $A$  satisfying the condition  $x \neq 0 \implies x^H A x > 0$  a positive definite matrix. Be warned.

3. Let  $A$  be positive definite and let  $X \in \mathbb{C}^{n \times p}$ . Then  $X^H A X$  is positive semidefinite. It is positive definite if and only if  $X$  is of full column rank.

4. Any principal submatrix of a positive definite matrix is positive definite.
5. The diagonal elements of a positive definite matrix are positive.
6.  $A$  is positive (semi)definite if and only if its eigenvalues are positive (nonnegative).
7. If  $A$  is positive semidefinite, its determinant is nonnegative. If  $A$  is positive definite, its determinant is positive.
8. If  $A$  is positive definite, then  $A$  is nonsingular and  $A^{-1}$  is positive definite.
9. If  $A$  is positive (semi)definite, then there is a positive (semi)definite matrix  $A^{\frac{1}{2}}$  such that

$$(A^{\frac{1}{2}})^2 = A.$$

The matrix  $A^{\frac{1}{2}}$  is unique.

### 4. THE LU DECOMPOSITION

1. Let  $A$  be of order  $n$  and assume that the first  $n-1$  leading principal submatrices of  $A$  are nonsingular. Then  $A$  can be factored uniquely in the form

$$A = LU,$$

where  $L$  is unit lower triangular and  $U$  is upper triangular. We call the factorization the LU decomposition of  $A$  and call the matrices  $L$  and  $U$  the L-factor and the U-factor.

2. Gaussian elimination (Algorithm I:3.1.1) computes the LU decomposition. Pivoting may be necessary for stability (see Algorithm I:3.1.3).

## 5. THE CHOLESKY DECOMPOSITION

1. Any positive definite matrix can be uniquely factored in the form

$$A = R^H R,$$

where  $R$  is an upper triangular matrix with positive diagonal elements. This factorization is called the Cholesky decomposition of  $A$ . The matrix  $R$  is called the Cholesky factor of  $A$ . It is unique.

2. The Cholesky decomposition can be computed by variants of Gaussian elimination. See Algorithm I:3.2.1.
3. There is a pivoted version of the decomposition. Specifically, there is a permutation matrix  $P$  such that

$$P^T A P = R^T R,$$

where the elements of  $R$  satisfy

$$|r_{kk}|^2 \geq \sum_{i=k}^i |r_{ij}|^2, \quad j = 1, \dots, n.$$

Algorithm I.5.2.2 computes a pivoted Cholesky decomposition.

## 6. THE QR DECOMPOSITION

1. Let  $X$  be an  $n \times p$  matrix with  $n \geq p$ . Then there is a unitary matrix  $Q$  of order  $n$  and an upper triangular matrix  $R$  of order  $p$  with nonnegative diagonal elements such that

$$X = Q \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

This factorization is called the QR decomposition of  $A$ . The matrix  $Q$  is the Q-factor of  $A$ , and the matrix  $R$  is the R-factor.

2. If we partition  $Q = (Q_X \ Q_\perp)$ , where  $Q_X$  has  $p$  columns, then we can write

$$X = Q_X R,$$

which is called the QR factorization of  $R$ .

3. If  $X$  is of full rank, the QR factorization is unique.
4. If  $X$  is of full rank, then  $Q_X$  is an orthonormal basis for the column space of  $X$  and  $Q_\perp$  is an orthonormal basis for the orthogonal complement of the column space of  $X$ .

5. If  $X$  is of full rank,  $A = X^H X$  is positive definite, and  $R$  is the Cholesky factor of  $A$ .
6. The QR decomposition can be computed by orthogonal triangularization (Algorithm I.4.1.2). The QR factorization can be computed by the Gram–Schmidt algorithm (Algorithms I.4.1.11–13).
7. There is a pivoted version of the QR decomposition. Specifically, there is a permutation matrix  $P$  such that  $X P$  has the QR decomposition

$$X P = Q \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where the elements of  $R$  satisfy

$$|r_{kk}|^2 \geq \sum_{i=k}^i |r_{ij}|^2, \quad j = 1, \dots, n.$$

Algorithm I.5.2.1 computes a pivoted QR decomposition.

## 7. PROJECTORS

1. If  $P^2 = P$ , then  $P$  is a projector (or projection).
2. Any projector  $P$  can be written in the form

$$P = XY^H, \quad Y^H X = I.$$

The spaces  $\mathcal{X} = \mathcal{R}(X)$  and  $\mathcal{Y} = \mathcal{R}(Y)$  uniquely define  $P$ , which is said to project onto  $\mathcal{X}$  along the orthogonal complement  $\mathcal{Y}_\perp$  of  $\mathcal{Y}$ .

3. If  $P$  is a projector onto  $\mathcal{X}$  along  $\mathcal{Y}_\perp$ ,  $I - P$  is its complementary projector. It projects onto  $\mathcal{Y}_\perp$  along  $\mathcal{X}$ . Any vector  $z$  can be represented uniquely as the sum of a vector  $Px \in \mathcal{X}$  and a vector  $(I - P)x \in \mathcal{Y}_\perp$ .
4. If  $P$  is a Hermitian projector, then  $P = XX^H$ , where  $X$  is any orthonormal basis for  $\mathcal{R}(P)$ . Such a projector is called an orthogonal projector. Its complementary projector is written  $P_\perp$ .

*This page intentionally left blank*

## REFERENCES

- [1] A. C. Aitken. On Bernoulli's numerical solution of algebraic equations. *Proceedings of the Royal Society of Edinburgh*, 46:289–305, 1926.
- [2] A. C. Aitken. Studies in practical mathematics, II: The evaluation of the latent roots and latent vectors of a matrix. *Proceedings of the Royal Society of Edinburgh*, 57:269–304, 1937.
- [3] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, second edition, 1995.
- [4] P. M. Anselone and L. B. Rall. The solution of characteristic value-vector problems by Newton's method. *Numerische Mathematik*, 11:38–45, 1968.
- [5] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29, 1951.
- [6] J. Baglama, D. Calvetti, and L. Reichel. Iterative methods for the computation of a few eigenvalues of a large symmetric matrix. *BIT*, 37:400–440, 1996.
- [7] J. Baglama, D. Calvetti, and L. Reichel. Computation of a few small eigenvalues of a large matrix with application to liquid crystal modeling. *Journal of Computational Physics*, 146:203–226, 1998.
- [8] Z. Bai. Note on the quadratic convergence of Kogbetliantz's algorithm for computing the singular value decomposition. *Linear Algebra and Its Applications*, 104:131–140, 1988.
- [9] Z. Bai and J. Demmel. On a block implementation of Hessenberg QR iteration. *International Journal of High Speed Computing*, 1:97–112, 1989. Cited in [29]. Also available online as LAPACK Working Note 8 from <http://www.netlib.org/lapack/lawns/lawn08.ps>.
- [10] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, 2000.
- [11] Z. Bai and J. W. Demmel. On swapping diagonal blocks in real Schur form. *Linear Algebra and Its Applications*, 186:73–95, 1993.

- [12] Z. Bai and G. W. Stewart. Algorithm 776: SRRIT — A Fortran subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix. *ACM Transactions on Mathematical Software*, 23:494–513, 1997.
- [13] J. L. Barlow. Error analysis of update methods for the symmetric eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 14:598–816, 1993.
- [14] R. H. Bartels and G. W. Stewart. Algorithm 432: The solution of the matrix equation  $AX - BX = C$ . *Communications of the ACM*, 8:820–826, 1972.
- [15] W. Barth, R. S. Martin, and J. H. Wilkinson. Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection. *Numerische Mathematik*, 9:386–393, 1967. Also in [305, pages 249–256].
- [16] F. L. Bauer. Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischer Eigenwertprobleme. *Zeitschrift für angewandte Mathematik und Physik*, 8:214–235, 1957.
- [17] C. Bavey and G. W. Stewart. An algorithm for computing reducing subspaces by block diagonalization. *SIAM Journal on Numerical Analysis*, 16:359–367, 1979.
- [18] R. Bellman. *Introduction to Matrix Analysis*. McGraw–Hill, New York, second edition, 1970.
- [19] E. Beltrami. Sulle funzioni bilineari. *Giornale di Matematiche ad Uso degli Studenti Delle Università*, 11:98–106, 1873. An English translation by D. Boley is available in Technical Report 90–37, Department of Computer Science, University of Minnesota, Minneapolis, 1990.
- [20] A. Berman and A. Ben-Israel. A note on pencils of Hermitian or symmetric matrices. *SIAM Journal on Applied Mathematics*, 21:51–54, 1971.
- [21] M. W. Berry. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6:13–49, 1992.
- [22] R. Bhatia. *Perturbation Bounds for Matrix Eigenvalues*. Pitman Research Notes in Mathematics. Longmann Scientific & Technical, Harlow, Essex, 1987. Published in the USA by John Wiley.
- [23] R. Bhatia. *Matrix Analysis*. Springer, New York, 1997.
- [24] R. Bhatia, L. Elsner, and G. Krause. Bounds for the variation of the roots of a polynomial and the eigenvalues of a matrix. *Linear Algebra and Its Applications*, 142:195–209, 1990.
- [25] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

- [26] Å. Björck and G. H. Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27:579–594, 1973.
- [27] Å. Björck, R. J. Plemmons, and H. Schneider, editors. *Large-Scale Matrix Problems*. North-Holland, New York, 1981. A reprint of Volumn 34, 1980, of *Linear Algebra and Its Applications*.
- [28] C. W. Borchart. Bemerkung über die beiden vorstehenden Aufsätze. *Journal für die reine und angewandte Mathematik*, 53:281–283, 1857.
- [29] K. Braman, R. Byers, and R. Mathias. The multi-shift QR-algorithm: Aggressive deflation, maintaining well focused shifts, and level 3 performance. Technical report, Department of Mathematics, University of Kansas, Lawrence, 1999.
- [30] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen. Rank one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31:31–48, 1978.
- [31] A. L. Cauchy. Sur l'équation à l'aide de laquelle on détermine les inégalités séculaires des mouvements des planètes. In *Oeuvres Complètes (II<sup>e</sup> Série)*, volume 9. 1829.
- [32] A. Cayley. A memoir on the theory of matrices. *Philosophical Transactions of the Royal Society of London*, 148:17–37, 1858. Cited and reprinted in [33, v. 2, pp. 475–496].
- [33] A. Cayley. *The Collected Mathematical Papers of Arthur Cayley*. Cambridge University Press, Cambridge, 1889–1898. Arthur Cayley and A. R. Forsythe, editors. Thirteen volumes plus index. Reprinted 1963 by Johnson Reprint Corporation, New York.
- [34] T. F. Chan. Algorithm 581: An improved algorithm for computing the singular value decomposition. *ACM Transactions on Mathematical Software*, 8:84–88, 1982.
- [35] T. F. Chan. An improved algorithm for computing the singular value decomposition. *ACM Transactions on Mathematical Software*, 8:72–83, 1982.
- [36] S. Chandrasekaran. An efficient and stable algorithm for the symmetric-definite generalized eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 21:1202–1228, 2000.
- [37] J. P. Charlier, M. Vangegin, and P. Van Dooren. On efficient implementations of Kogbetliantz's algorithm for computing the singular value decomposition. *Numerische Mathematik*, 52:270–300, 1988.
- [38] F. Chatelin. *Valeurs Propres de Matrices*. Masson, Paris, 1988.
- [39] F. Chatelin. *Eigenvalues of Matrices*. Wiley, New York, 1993. Expanded translation of [38].

- [40] S. H. Cheng and N. J. Higham. The nearest definite pair for the Hermitian generalized eigenvalue problem. *Linear Algebra and Its Applications*, 302–303:63–76, 1999.
- [41] M. Clint and A. Jennings. The evaluation of eigenvalues and eigenvectors of real symmetric matrices by simultaneous iteration. *Computer Journal*, 13:76–80, 1970.
- [42] M. Clint and A. Jennings. A simultaneous iteration method for the unsymmetric eigenvalue problem. *Journal of the Institute for Mathematics and Applications*, 8:111–121, 1971.
- [43] C. R. Crawford. Algorithm 646 PDFIND: A routine to find a positive definite linear combination of two real symmetric matrices. *ACM Transactions on Mathematical Software*, 12:278–282, 1986.
- [44] C. R. Crawford and Y. S. Moon. Finding a positive definite linear combination of two Hermitian matrices. *Linear Algebra and Its Applications*, 51:37–48, 1983.
- [45] J. Cullum and W. E. Donath. A block Lanczos algorithm for computing the Q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse real symmetric matrices. In *Proceedings of the 1974 IEEE Conference on Decision and Control, Phoenix, AZ*, pages 505–509, 1974. Cited in [94].
- [46] J. Cullum and R. A. Willoughby. Computing eigenvalues of very large symmetric matrices—An implementation of a Lanczos algorithm with no reorthogonalization. *Journal of Computational Physics*, 44:329–358, 1984.
- [47] J. Cullum, R. A. Willoughby, and M. Lake. A Lanczos algorithm for computing singular values and vectors of large matrices. *SIAM Journal on Scientific and Statistical Computing*, 4:197–215, 1983.
- [48] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations Volume I: Theory. II: Programs*. Birkhäuser, Stuttgart, 1985. Cited in [25].
- [49] J. J. M. Cuppen. A divide and conquer method for the symmetric eigenproblem. *Numerische Mathematik*, 36:177–195, 1981.
- [50] A. R. Curtis and J. K. Reid. On the automatic scaling of matrices for Gaussian elimination. *Journal of the Institute for Mathematics and Applications*, 10:118–124, 1972.
- [51] J. W. Daniel. *The Approximation Minimization of Functionals*. Prentice–Hall, Englewood Cliffs, NJ, 1971.
- [52] B. N. Datta. *Numerical Linear Algebra and Applications*. Brooks/Cole, Pacific Grove, CA, 1995.

- [53] E. R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *Journal of Computational Physics*, 17:87–94, 1975.
- [54] C. Davis and W. Kahan. The rotation of eigenvectors by a perturbation. III. *SIAM Journal on Numerical Analysis*, 7:1–46, 1970.
- [55] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19:400–408, 1982.
- [56] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač. Computing the singular value decomposition with high relative accuracy. *Linear Algebra and Its Applications*, 299:21–80, 1997.
- [57] J. Demmel and B. Kågström. Computing stable eigendecompositions of matrix pencils. *Linear Algebra and Its Applications*, 88/89:139–186, 1987.
- [58] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific and Statistical Computing*, 11:873–912, 1990.
- [59] J. Demmel and K. Veselić. Jacobi’s method is more accurate than QR. *SIAM Journal on Matrix Analysis and Applications*, 13:1204–1245, 1992.
- [60] J. W. Demmel. The condition number of equivalence transformations that block diagonalize matrix pencils. *SIAM Journal on Numerical Analysis*, 20:599–610, 1983.
- [61] J. W. Demmel. Computing stable eigendecompositions of matrices. *Linear Algebra and Its Applications*, 79:163–193, 1986.
- [62] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.
- [63] J. E. Dennis and J. J. Moré. Quasi-Newton methods, motivations and theory. *SIAM Review*, 19:46–89, 1977.
- [64] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.
- [65] I. S. Dhillon. A new  $O(n^2)$  algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem. Technical Report CSD-97-971, Computer Science Division, University of California at Berkeley, 1998.
- [66] J. J. Dongarra and D. C. Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM Journal on Scientific and Statistical Computing*, 8:s139–s154, 1987.
- [67] Z. Drmač. Implementation of Jacobi rotations for accurate singular value computation in floating point arithmetic. *SIAM Journal on Scientific Computing*, 18:1200–1222, 1997.

- [68] I. S. Duff and J. A. Scott. Computing selected eigenvalues of large sparse unsymmetric matrices using subspace iteration. *ACM Transactions on Mathematical Software*, 19:137–159, 1993.
- [69] L. Elsner. On the variation of the spectra of matrices. *Linear Algebra and Its Applications*, 47:127–138, 1982.
- [70] L. Elsner. An optimal bound for the spectral variation of two matrices. *Linear Algebra and Its Applications*, 71:77–80, 1985.
- [71] T. Ericsson. A generalized eigenvalue problem and the Lanczos algorithm. In J. Cullum and R. Willoughby, editors, *Large Scale Eigenvalue Problems*, pages 95–120. North-Holland, Amsterdam, 1986.
- [72] T. Ericsson and A. Ruhe. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Mathematics of Computation*, 35:1251–1268, 1980.
- [73] K. V. Fernando and B. N. Parlett. Accurate singular values and differential qd algorithms. *Numerische Mathematik*, 67:191–229, 1994.
- [74] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst. Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM Journal on Scientific Computing*, 20:94–125, 1998.
- [75] G. E. Forsythe and P. Henrici. The cyclic Jacobi method for computing the principle values of a complex matrix. *Transaction of the American Mathematical Society*, 94:1–23, 1960.
- [76] J. G. F. Francis. The QR transformation, parts I and II. *Computer Journal*, 4:265–271, 332–345, 1961, 1962.
- [77] V. Frayssé and V. Toumazou. A note on the normwise perturbation theory for the regular generalized eigenproblem  $Ax = \lambda Bx$ . *Numerical Linear Algebra with Applications*, 5:1–10, 1998.
- [78] R. W. Freund. Quasi-kernel polynomials and their use in non-Hermitian matrix iterations. *Journal of Computational and Applied Mathematics*, 43:135–148, 1992.
- [79] Roland W. Freund, M. H. Gutknecht, and N. M. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. Research Report 91–06, Interdisciplinary Project Center for Supercomputing, Eidgenössische Technische Hochschule Zürich, 1991.
- [80] F. R. Gantmacher. *The Theory of Matrices, Vols. I, II*. Chelsea Publishing Company, New York, 1959.

- [81] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler. Matrix eigensystem routines—EISPACK guide extension. In *Lecture Notes in Computer Science*. Springer-Verlag, New York, 1977.
- [82] S. A. Gershgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izvestia Akademii Nauk SSSR, Mathematics and Natural Sciences*, 6:749–754, 1931.
- [83] W. Givens. Numerical computation of the characteristic values of a real matrix. Technical Report 1574, Oak Ridge National Laboratory, Oak Ridge, TN, 1954.
- [84] H. H. Goldstine and L. P. Horowitz. A procedure for the diagonalization of normal matrices. *Journal of the ACM*, 6:176–195, 1959.
- [85] H. H. Goldstine, F. J. Murray, and J. von Neumann. The Jacobi method for real symmetric matrices. *Journal of the ACM*, 6:59–96, 1959.
- [86] G. H. Golub. Least squares, singular values, and matrix approximations. *Aplikace Mathematicky*, 13:44–51, 1968.
- [87] G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15:318–344, 1973.
- [88] G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis*, 2:205–224, 1965.
- [89] G. H. Golub, F. T. Luk, and M. L. Overton. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software*, 7:149–169, 1981.
- [90] G. H. Golub, S. Nash, and C. Van Loan. Hessenberg–Schur method for the problem  $AX + XB = C$ . *IEEE Transactions on Automatic Control*, AC-24:909–913, 1979.
- [91] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solution. *Numerische Mathematik*, 14:403–420, 1970. Also in [305, pages 134–151].
- [92] G. H. Golub and R. Underwood. The block Lanczos method for computing eigenvalues. In J. Rice, editor, *Mathematical Software III*, pages 364–377. Academic Press, New York, 1977.
- [93] G. H. Golub, R. Underwood, and J. H. Wilkinson. The Lanczos algorithm for the symmetric  $Ax = \lambda Bx$  problem. Technical Report STAN-CS-72-270, Computer Science, Stanford University, Stanford, CA, 1972. Cited in [94].
- [94] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, second edition, 1989.
- [95] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

- [96] G. H. Golub and J. H. Wilkinson. Ill-conditioned eigensystems and the computation of the Jordan canonical form. *SIAM Review*, 18:578–619, 1976.
- [97] J. Grcar. *Analyses of the Lanczos Algorithm and of the Approximation Problem in Richardson’s Method*. PhD thesis, University of Illinois at Urbana-Champaign, 1981.
- [98] R. G. Grimes, J. G. Lewis, and H. D. Simon. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 15:228–272, 1994.
- [99] M. Gu. Finding well-conditioned similarities to block-diagonalized nonsymmetric matrices is NP-hard. Technical Report LBL-35100, Physics Division, Lawrence Berkeley Laboratory, Berkeley, CA, 1994.
- [100] M. Gu and S. C. Eisenstat. Stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 15:1266–1276, 1994.
- [101] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM Journal on Matrix Analysis and Applications*, 16:79–92, 1995.
- [102] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 16:172–191, 1995.
- [103] M. H. Gutknecht. A completed theory of the unsymmetric Lanczos process and related algorithms: Part I. *SIAM Journal on Matrix Analysis and Applications*, 13:594–639, 1992.
- [104] M. H. Gutknecht. A completed theory of the unsymmetric Lanczos process and related algorithms: Part II. *SIAM Journal on Matrix Analysis and Applications*, 15:15–58, 1994.
- [105] M. H. Gutknecht and K. J. Ressel. Look-ahead procedures for Lanczos-type product methods based on three-term Lanczos recurrences. *SIAM Journal on Matrix Analysis and Applications*, 21:1051–1078, 2000.
- [106] M. J. Hadamard. Essai sur l’étude des fonctions données par leur développement de taylor. *Journ. de Math. (4<sup>e</sup> série)*, 8:101–186, 1892.
- [107] V. Hari. On sharp quadratic convergence bounds for the serial Jacobi methods. *Numerische Mathematik*, 60:375–407, 1991.
- [108] V. Hari and K. Veselić. On Jacobi methods for singular value decompositions. *SIAM Journal on Scientific and Statistical Computing*, 8:741–754, 1987.
- [109] M. T. Heath, A. J. Laub, C. C. Paige, and R. C. Ward. Computing the SVD of a product of two matrices. *SIAM Journal on Scientific and Statistical Computing*, 7:1147–1159, 1986.

- [110] P. Henrici. On the speed of convergence of cyclic and quasicyclic Jacobi methods for computing eigenvalues of Hermitian matrices. *Journal of the Society for Industrial and Applied Mathematics*, 6:144–162, 1958.
- [111] P. Henrici. Bounds for iterates, inverses, spectral variation and fields of values of nonnormal matrices. *Numerische Mathematik*, 4:24–39, 1962.
- [112] D. J. Higham and N. J. Higham. Structured backward error and condition of generalized eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 20:493–512, 1998.
- [113] N. J. Higham. The accuracy of floating point summation. *SIAM Journal on Scientific Computing*, 14:783–799, 1993.
- [114] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 1996.
- [115] N. J. Higham. QR factorization with complete pivoting and accurate computation of the SVD. *Linear Algebra and Its Applications*, 309:153–174, 2000.
- [116] D. Hilbert. *Grundzüge einer allgemeinen Theorie der linearen Integralgleichungen*. B. G. Teubner, Leipzig and Berlin, 1912. A collection of papers that appeared in *Gött. Nachr.* 1904–1910.
- [117] W. Hoffman. Iterative algorithms for Gram–Schmidt orthogonalization. *Computing*, 41:335–348, 1989.
- [118] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.
- [119] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991.
- [120] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441 and 498–520, 1933.
- [121] A. S. Householder. *The Theory of Matrices in Numerical Analysis*. Dover, New York, 1964. Originally published by Blaisdell, New York.
- [122] A. S. Householder. *The Numerical Treatment of a Single Nonlinear Equation*. McGraw–Hill, New York, 1970.
- [123] A. S. Householder and F. L. Bauer. On certain methods for expanding the characteristic polynomial. *Numerische Mathematik*, 1:29–37, 1958.
- [124] I. C. F. Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 39:254–291, 1997.
- [125] I. C. F. Ipsen. Relative perturbation bounds for matrix eigenvalues and singular values. In *Acta Numerica*, pages 151–201. Cambridge University Press, Cambridge, 1998.

- [126] I. C. F. Ipsen. Absolute and relative perturbation bounds for invariant subspaces of matrices. *Linear Algebra and Its Applications*, 309:45–56, 2000.
- [127] C. G. J. Jacobi. Ueber eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden lineären Gleichungen. *Astronomische Nachrichten*, 22:297–306, 1845.
- [128] C. G. J. Jacobi. Über ein leichtes Verfahren die in der Theorie der Säculärstörungen vorkommenden Gleichungen numerisch aufzulösen. *Journal für die reine und angewandte Mathematik*, 30:51–94, 1846.
- [129] C. G. J. Jacobi. Über eine elementare Transformation eines in Buzug jedes von zwei Variablen-Systemen linearen und homogenen Ausdrucks. *Journal für die reine und angewandte Mathematik*, 53:265–270, 1857, posthumous.
- [130] A. Jennings. A direct iteration method of obtaining latent roots and vectors of a symmetric matrix. *Proceedings of the Cambridge Philosophical Society*, 63:755–765, 1967.
- [131] Z. Jia. Refined iterative algorithm based on Arnoldi’s process for large unsymmetric eigenproblems. *Linear Algebra and Its Applications*, 259:1–23, 1997.
- [132] Z. Jia. A refined subspace iteration algorithm for large sparse eigenproblems. *Applied Numerical Mathematics*, 32:35–52, 2000.
- [133] Z. Jia and G. W. Stewart. On the convergence of Ritz values, Ritz vectors, and refined Ritz vectors. Technical Report TR-3986, Department of Computer Science, University of Maryland, College Park, 1999.
- [134] Z. Jia and G. W. Stewart. An analysis of the Rayleigh–Ritz method for approximating eigenspaces. *Mathematics of Computation*, 70:637–647, 2001.
- [135] E. Jiang and Z. Zhang. A new shift of the QL algorithm for irreducible symmetric tridiagonal matrices. *Linear Algebra and Its Applications*, 65:261–272, 1985.
- [136] C. Jordan. *Traité des Substitutions et des Équations Algébriques*. Paris, 1870.  
Cited in [167].
- [137] C. Jordan. Mémoire sur les formes bilinéaires. *Journal de Mathématiques Pures et Appliquées, Deuxième Série*, 19:35–54, 1874.
- [138] C. Jordan. Essai sur la géométrie à  $n$  dimensions. *Bulletin de la Société Mathématique*, 3:103–174, 1875.
- [139] B. Kågström and A. Ruhe. Algorithm 560: JNF, an algorithm for numerical computation of the Jordan normal form of a complex matrix. *Transactions on Mathematical Software*, 6:437–443, 1980.

- [140] B. Kågström and A. Ruhe. An algorithm for numerical computation of the Jordan normal form of a complex matrix. *Transactions on Mathematical Software*, 6:398–419, 1980.
- [141] B. Kågström and P Wiberg. Extracting partial canonical structure for large scale eigenvalue problems. *Numerical Algorithms*, 24:195–237, 1999.
- [142] W. Kahan. Inclusion theorems for clusters of eigenvalues of Hermitian matrices. Technical report, Computer Science Department, University of Toronto, 1967.
- [143] W. Kahan and B. N. Parlett. How far should you go with the Lanczos process? In J. Bunch and D. Rose, editors, *Sparse Matrix Computations*, pages 131–144. Academic Press, New York, 1976.
- [144] W. Kahan, B. N. Parlett, and E. Jiang. Residual bounds on approximate eigen-systems of nonnormal matrices. *SIAM Journal on Numerical Analysis*, 19:470–484, 1982.
- [145] S. Kaniel. Estimates for some computational techniques in linear algebra. *Mathematics of Computation*, 20:369–378, 1966.
- [146] T. Kato. *Perturbation Theory for Linear Operators*. Springer-Verlag, New York, 1966.
- [147] M. Kline. *Mathematical Thought from Ancient to Modern Times*. Oxford University Press, New York, 1972.
- [148] A. V. Knyazev. New estimates for Ritz vectors. *Mathematics of Computation*, 66:985–995, 1997.
- [149] E. G. Kogbetliantz. Solution of linear systems by diagonalization of coefficients matrix. *Quarterly of Applied Mathematics*, 13:123–132, 1955.
- [150] N. Kollerstrom. Thomas Simpson and “Newtons method of approximation”: An enduring myth. *British Journal for the History of Science*, 25:347–354, 1992.
- [151] J. König. Über eine Eigenschaft der Potenzreihen. *Mathematische Annalen*, 23:447–449, 1884.
- [152] L. Kronecker. Algebraische Reduction der Schaaren bilinearer Formen. *Sitzungberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, pages 1225–1237, 1890.
- [153] A. N. Krylov. O Čislennom rešenii uravnenija, kotorym v techničeskikh voprasah opredeljajutsja častoy malyh kolebanií material'nyh. *Izv. Adad. Nauk SSSR otd. Mat. Estest.*, pages 491–539, 1931. Cited in [121].

- [154] V. N. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *Zhurnal Vychislitelnoi Matematiki i Matematicheskoi Fiziki*, 1:555–570, 1961. In Russian. Translation in *USSR Computational Mathematics and Mathematical Physics*, 1:637–657, 1962.
- [155] J.-L. Lagrange. Solution de différents problèmes de calcul intégral. *Miscellanea Taurinensis*, 3, 1762–1765. Cited and reprinted in [156, v. 1, pages 472–668].
- [156] J.-L. Lagrange. *Oeuvres de Lagrange*. Gauthier–Villars, Paris, 1867–1892.
- [157] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45:255–282, 1950.
- [158] R. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications*, 17:789–821, 1996.
- [159] R. B. Lehoucq. The computation of elementary unitary matrices. *ACM Transactions on Mathematical Software*, 22:393–400, 1996.
- [160] R. B. Lehoucq. On the convergence of an implicitly restarted Arnoldi method. Technical Report SAND99-1756J, Sandia National Laboratories, Albuquerque, NM, 1999.
- [161] R. B. Lehoucq and J. A. Scott. An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices. Preprint MCS-P547-1195, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1996.
- [162] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, 1998.
- [163] C.-K. Li and R. Mathias. The definite generalized eigenvalue problem: A re-worked perturbation theory. Department of Mathematics, College of William & Mary, 2000.
- [164] R.-C. Li. Solving secular equations stably and efficiently. Technical Report UCB//CSD-94-851, Computer Science Division, University of California, Berkeley, 1993.
- [165] R.-C. Li. Relative perturbation theory: I. Eigenvalue and singular value variations. *SIAM Journal on Matrix Analysis and Applications*, 19:956–982, 1998.
- [166] R.-C. Li. Relative perturbation theory: II. Eigenspace and singular subspace variations. *SIAM Journal on Matrix Analysis and Applications*, 20:471–492, 1998.
- [167] C. C. Mac Duffee. *The Theory of Matrices*. Chelsea, New York, 1946.

- [168] M. Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities*. Allyn and Bacon, Boston, 1964.
- [169] R. S. Martin, C. Reinsch, and J. H. Wilkinson. Householder tridiagonalization of a real symmetric matrix. *Numerische Mathematik*, 11:181–195, 1968. Also in [305, pages 212–226].
- [170] R. S. Martin and J. H. Wilkinson. Reduction of the symmetric eigenproblem  $Ax = \lambda Bx$  and related problems to standard form. *Numerische Mathematik*, 11:99–110, 1968. Also in [305, pages 303–314].
- [171] W. F. Mascarenhas. On the convergence of the Jacobi method for arbitrary orderings. *SIAM Journal on Matrix Analysis and Applications*, 16:1197–1209, 1995.
- [172] R. Mathias. Accurate eigensystem computations by Jacobi methods. *SIAM Journal on Matrix Analysis and Applications*, 16:977–1003, 1995.
- [173] R. Mathias. Spectral perturbation bounds for positive definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 18:959–980, 1997.
- [174] R. Mathias. Quadratic residual bounds for the Hermitian eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 18:541–550, 1998.
- [175] H. I. Medley and R. S. Varga. On smallest isolated Gershgorin disks for eigenvalues. III. *Numerische Mathematik*, 11:361–369, 1968.
- [176] K. Meerbergen and A. Spence. Implicitly restarted Arnoldi with purification for the shift-invert transformation. *Mathematics of Computation*, 66:667–689, 1997.
- [177] D. S. Mitrinović. *Analytic Inequalities*. Springer-Verlag, New York, 1970.
- [178] C. B. Moler and G. W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10:241–256, 1973.
- [179] R. B. Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra and Its Applications*, 154–156:289–309, 1991.
- [180] R. B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Mathematics of Computation*, 65:1213–1230, 1996.
- [181] R. B. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 21:1112–1135, 2000.
- [182] C. L. Müntz. Solution direct de l'équation séculaire et de quelques problèmes analogues transcendants. *Comptes Rendus de l'Académie des Sciences, Paris*, 156:43–46, 1913.

- [183] M. Newman and P. F. Flanagan. Eigenvalue extraction in NASTRAN by the tridiagonal reduction (FEER) method—Real eigenvalue analysis. NASA Contractor Report CR-2731, NASA, Washington DC, 1976. Cited in [236].
- [184] B. Nour-Omid, B. N. Parlett, T. Ericsson, and P. S. Jensen. How to implement the spectral transformation. *Mathematics of Computation*, 48:663–673, 1897.
- [185] W. Oettli and W. Prager. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numerische Mathematik*, 6:405–409, 1964.
- [186] J. Olsen, P. Jørgensen, and J. Simons. Passing the one-billion limit in full configuration-interaction (FCI) calculations. *Chemical Physics Letters*, 169:463–472, 1990.
- [187] E. E. Osborne. On pre-conditioning of matrices. *Journal of the ACM*, 7:338–345, 1960.
- [188] A. M. Ostrowski. On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. I. *Archive for Rational Mechanics and Analysis*, 1:233–241, 1958.
- [189] A. M. Ostrowski. On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. II. *Archive for Rational Mechanics and Analysis*, 2:423–428, 1958.
- [190] A. M. Ostrowski. On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. III (generalized Rayleigh quotient and characteristic roots with linear elementary divisors). *Archive for Rational Mechanics and Analysis*, 3:325–340, 1959.
- [191] A. M. Ostrowski. On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. IV (generalized Rayleigh quotient for nonlinear elementary divisors). *Archive for Rational Mechanics and Analysis*, 3:341–347, 1959.
- [192] A. M. Ostrowski. On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. VI (usual Rayleigh quotient for nonlinear elementary divisors). *Archive for Rational Mechanics and Analysis*, 4:153–165, 1959.
- [193] A. M. Ostrowski. On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. V (usual Rayleigh quotient for non-Hermitian matrices and linear elementary divisors). *Archive for Rational Mechanics and Analysis*, 3:472–481, 1959.
- [194] A. M. Ostrowski. *Solution of Equations and Systems of Equations*. Academic Press, New York, 1960.

- [195] C. C. Paige. Practical use of the symmetric Lanczos process with reorthogonalization. *BIT*, 10:183–195, 1970.
- [196] C. C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*. PhD thesis, University of London, 1971.
- [197] C. C. Paige. Computational variants of the Lanczos method for the eigenproblem. *Journal of the Institute of Mathematics and Its Applications*, 10:373–381, 1972.
- [198] C. C. Paige. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *Journal of the Institute of Mathematics and Its Applications*, 18:341–349, 1976. Cited in [25].
- [199] C. C. Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra and Its Applications*, 34:235–258, 1980. Reprinted in [27].
- [200] C. C. Paige. Computing the generalized singular value decomposition. *SIAM Journal on Scientific and Statistical Computing*, 7:1126–1146, 1986.
- [201] C. C. Paige, B. N. Parlett, and H. van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numerical Linear Algebra with Applications*, 2:115–134, 1995.
- [202] C. C. Paige and M. A. Saunders. Toward a generalized singular value decomposition. *SIAM Journal on Numerical Analysis*, 18:398–405, 1981.
- [203] C. C. Paige and P. Van Dooren. On the quadratic convergence of Kogbetliantz's algorithm for computing the singular value decomposition. *Linear Algebra and Its Applications*, 77:301–313, 1986.
- [204] B. N. Parlett. The origin and development of methods of the LR type. *SIAM Review*, 6:275–295, 1964.
- [205] B. N. Parlett. A new look at the Lanczos algorithm for solving symmetric systems and linear equations. *Linear Algebra and Its Applications*, 29:323–346, 1980.
- [206] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ, 1980. Reissued with revisions by SIAM, Philadelphia, 1998.
- [207] B. N. Parlett and J. Le. Forward instability of tridiagonal qr. *SIAM Journal on Matrix Analysis and Applications*, 14:279–316, 1993.
- [208] B. N. Parlett and C. Reinsch. Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numerische Mathematik*, 13:293–304, 1969. Also in [305, pages 315–326].

- [209] B. N. Parlett and D. S. Scott. The Lanczos algorithm with selective orthogonalization. *Mathematics of Computation*, 33:217–238, 1979.
- [210] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Mathematics of Computation*, 44:105–124, 1985.
- [211] G. Peters and J. H. Wilkinson. Inverse iteration, ill-conditioned equations, and Newton’s method. *SIAM Review*, 21:339–360, 1979. Cited in [94].
- [212] D. A. Pope and C. Tompkins. Maximizing functions of rotations: Experiments concerning speed of diagonalization of symmetric matrices using Jacobi’s method. *Journal of the ACM*, 4:459–466, 1957.
- [213] D. Porter and D. S. G. Stirling. *Integral Equations*. Cambridge University Press, Cambridge, 1990.
- [214] M. J. D. Powell. A new algorithm for unconstrained optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, New York, 1970. Cited in [64, page 196].
- [215] Lord Rayleigh (J. W. Strutt). On the calculation of the frequency of vibration of a system in its gravest mode, with an example from hydrodynamics. *The Philosophical Magazine*, 47:556–572, 1899. Cited in [206].
- [216] J. L. Rigal and J. Gaches. On the compatibility of a given solution with the data of a linear system. *Journal of the ACM*, 14:543–548, 1967.
- [217] W. Ritz. Über eine neue Method zur Lösung gewisser Variationsprobleme der mathematischen Physik. *Journal für die reine und angewandte Mathematik*, 135:1–61, 1909.
- [218] A. Ruhe. Perturbation bounds for means of eigenvalues and invariant subspaces. *BIT*, 10:343–354, 1970.
- [219] A. Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Mathematics of Computation*, 33:680–687, 1979.
- [220] A. Ruhe. Rational Krylov algorithms for nonsymmetric eigenvalue problems. II. Matrix pairs. *Linear Algebra and Its Applications*, 197–198:283–295, 1994.
- [221] A. Ruhe. Rational Krylov subspace method. In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems*, pages 246–249. SIAM, Philadelphia, 2000.
- [222] H. Rutishauser. Anwendungen Des Quotienten-Differenzen-Algorithmus. *Zeitschrift für angewandte Mathematik und Physik*, 5:496–508, 1954.
- [223] H. Rutishauser. Der Quotienten-Differenzen-Algorithmus. *Zeitschrift für angewandte Mathematik und Physik*, 5:233–251, 1954.

- [224] H. Rutishauser. Une méthode pour la determination des valeurs propres d'une matrice. *Comptes Rendus de l'Académie des Sciences, Paris*, 240:34–36, 1955.
- [225] H. Rutishauser. Solution of eigenvalue problems with the LR-transformation. *National Bureau of Standards Applied Mathematics Series*, 49:47–81, 1958.
- [226] H. Rutishauser. The Jacobi method for real symmetric matrices. *Numerische Mathematik*, 9:1–10, 1966. Also in [305, pages 202–211].
- [227] H. Rutishauser. Computational aspects of F. L. Bauer's simultaneous iteration method. *Numerische Mathematik*, 13:4–13, 1969.
- [228] H. Rutishauser. Simultaneous iteration method for symmetric matrices. *Numerische Mathematik*, 16:205–223, 1970. Also in [305, pages 284–301].
- [229] Y. Saad. On the rates of convergence of the Lanczos and the block Lanczos methods. *SIAM Journal on Numerical Analysis*, 17:687–706, 1980.
- [230] Y. Saad. Variations of Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra and Its Applications*, 34:269–295, 1980.
- [231] Y. Saad. Projection methods for solving large sparse eigenvalue problems. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 121–144. Springer-Verlag, New York, 1983.
- [232] Y. Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Mathematics of Computation*, 42:567–588, 1984.
- [233] Y. Saad. *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms*. John Wiley, New York, 1992.
- [234] J. Schur. Über die charakteristischen Wurzeln einer linearen Substitution mit einer Anwendung auf die Theorie der Integralgleichungen. *Mathematische Annalen*, 66:448–510, 1909.
- [235] H. R. Schwarz. Tridiagonalization of a symmetric band matrix. *Numerische Mathematik*, 12:231–241, 1968. Also in [305, pages 273–283].
- [236] D. S. Scott. The advantages of inverted operators in Rayleigh–Ritz approximations. *SIAM Journal on Scientific and Statistical Computing*, 3:68–75, 1982.
- [237] H. D. Simon. Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear Algebra and Its Applications*, 61:101–132, 1984.
- [238] H. D. Simon. The Lanczos algorithm with partial reorthogonalization. *Mathematics of Computation*, 42:115–142, 1984.
- [239] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and Henk van der Vorst. Jacobi–Davidson type methods for generalized eigenproblems and polynomial eigenproblems. *BIT*, 36:595–633, 1996.

- [240] G. L. G. Sleijpen and H. A. van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 17:401–425, 1996.
- [241] G. L. G. Sleijpen and H. A. van der Vorst. Generalized Hermitian eigenvalue problems: Jacobi–Davidson methods. In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems*, pages 123–127. SIAM, Philadelphia, 2000.
- [242] G. L. G. Sleijpen and H. A. van der Vorst. Generalized non-Hermitian eigenvalue problems: Jacobi–Davidson method. In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems*, pages 238–246. SIAM, Philadelphia, 2000.
- [243] G. L. G. Sleijpen and H. A. van der Vorst. The Hermitian eigenvalue problem: Jacobi–Davidson methods. In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems*, pages 88–105. SIAM, Philadelphia, 2000.
- [244] G. L. G. Sleijpen and H. A. van der Vorst. Non-Hermitian eigenvalue problems: Jacobi–Davidson methods. In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems*, pages 221–228. SIAM, Philadelphia, 2000.
- [245] G. L. G. Sleijpen, H. A. van der Vorst, and E. Meijerink. Efficient expansion of subspaces in the Jacobi–Davidson method for standard and generalized eigenproblems. *Electronic Transactions on Numerical Analysis*, 7:75–89, 1998.
- [246] D. C. Sorensen. Implicit application of polynomial filters in a  $k$ -step Arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, 13:357–385, 1992.
- [247] D. C. Sorensen and P. T. P. Tang. On the orthogonality of eigenvectors computed by divide and conquer techniques. *SIAM Journal on Numerical Analysis*, 28:1752–1775, 1991.
- [248] A. Stathopoulos, Y. Saad, and Kesheng Wu. Dynamic thick restarting of the Davidson and the implicitly restarted Arnoldi algorithms. *SIAM Journal on Scientific Computing*, 19:227–245, 1998.
- [249] G. W. Stewart. Accelerating the orthogonal iteration for the eigenvalues of a Hermitian matrix. *Numerische Mathematik*, 13:362–376, 1969.
- [250] G. W. Stewart. Error bounds for approximate invariant subspaces of closed linear operators. *SIAM Journal on Numerical Analysis*, 8:796–808, 1971.
- [251] G. W. Stewart. On the sensitivity of the eigenvalue problem  $Ax = \lambda Bx$ . *SIAM Journal on Numerical Analysis*, 4:669–686, 1972.

- [252] G. W. Stewart. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Review*, 15:727–764, 1973.
- [253] G. W. Stewart. *Introduction to Matrix Computations*. Academic Press, New York, 1973.
- [254] G. W. Stewart. Modifying pivot elements in Gaussian elimination. *Mathematics of Computation*, 28:537–542, 1974.
- [255] G. W. Stewart. Gershgorin theory for the generalized eigenvalue problem  $Ax = \lambda Bx$ . *Mathematics of Computation*, 29:600–606, 1975.
- [256] G. W. Stewart. Algorithm 407: HQR3 and EXCHNG: FORTRAN programs for calculating the eigenvalues of a real upper Hessenberg matrix in a prescribed order. *ACM Transactions on Mathematical Software*, 2:275–280, 1976.
- [257] G. W. Stewart. Simultaneous iteration for computing invariant subspaces of a non-Hermitian matrix. *Numerische Mathematik*, 25:123–136, 1976.
- [258] G. W. Stewart. On the perturbation of pseudo-inverses, projections, and linear least squares problems. *SIAM Review*, 19:634–662, 1977.
- [259] G. W. Stewart. Computing the CS decomposition of a partitioned orthogonal matrix. *Numerische Mathematik*, 40:297–306, 1982.
- [260] G. W. Stewart. A method for computing the generalized singular value decomposition. In B. Kågström and A. Ruhe, editors, *Matrix Pencils*, pages 207–220. Springer-Verlag, New York, 1983.
- [261] G. W. Stewart. Two simple residual bounds for the eigenvalues of Hermitian matrices. *SIAM Journal on Matrix Analysis and Applications*, 12:205–208, 1991.
- [262] G. W. Stewart. Error analysis of QR updating with exponential windowing. *Mathematics of Computation*, 59:135–140, 1992.
- [263] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35:551–566, 1993.
- [264] G. W. Stewart. *Afternotes Goes to Graduate School*. SIAM, Philadelphia, 1998.
- [265] G. W. Stewart. *Matrix Algorithms I: Basic Decompositions*. SIAM, Philadelphia, 1998.
- [266] G. W. Stewart. The decompositional approach to matrix computations. *Computing in Science and Engineering*, 2:50–59, 2000.
- [267] G. W. Stewart. A Krylov–Schur algorithm for large eigenproblems. Technical Report TR-4127, Department of Computer Science, University of Maryland, College Park, 2000. To appear in the *SIAM Journal on Matrix Analysis and Applications*.

- [268] G. W. Stewart. A generalization of Saad's theorem on Rayleigh–Ritz approximations. *Linear Algebra and Its Applications*, 327:115–120, 2001.
- [269] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory*. Academic Press, New York, 1990.
- [270] W. J. Stewart and A. Jennings. Algorithm 570: LOPSI a simultaneous iteration method for real matrices. *ACM Transactions on Mathematical Software*, 7:230–232, 1981.
- [271] W. J. Stewart and A. Jennings. A simultaneous iteration algorithm for real matrices. *ACM Transactions on Mathematical Software*, 7:184–198, 1981.
- [272] J. J. Sylvester. On a theory of the syzygetic relations etc. *Phil. Trans.*, pages 481–84, 1853. Cited in [28].
- [273] J. J. Sylvester. Sur l'équation en matrices  $px = xq$ . *Comptes Rendus de l'Académie des Sciences*, pages 67–71 and 115–116, 1884.
- [274] O. Taussky. A recurring theorem on determinants. *American Mathematical Monthly*, 46:672–675, 1949.
- [275] D. R. Taylor. *Analysis of the Look Ahead Lanczos Algorithm*. PhD thesis, Department of Mathematics, University of California, Berkeley, 1982. Cited in [79].
- [276] L. N. Trefethen. Computation of pseudospectra. In *Acta Numerica*, pages 247–295. Cambridge University Press, Cambridge, 1999.
- [277] L. N. Trefethen. *Spectral Methods in MATLAB*. SIAM, Philadelphia, 2000.
- [278] L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.
- [279] L. N. Trefethen. Pseudospectra of linear operators. *SIAM Review*, 39:383–406, 1997.
- [280] R. Underwood. An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems. Technical Report STAN-CS-75-496, Computer Science, Stanford University, Stanford, CA, 1975. PhD thesis.
- [281] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Linear Algebra and Its Applications*, 27:103–140, 1979.
- [282] J. M. Van Kats and H. A. van der Vorst. Automatic monitoring of Lanczos schemes for symmetric or skew-symmetric generalized eigenvalue problems. Technical Report TR 7, Academisc Computer Centre Utrecht, Budapestlaan 6, De Vithof-Utrecht, the Netherlands, 1977. Cited in [236].
- [283] C. F. Van Loan. Generalizing the singular value decomposition. *SIAM Journal on Numerical Analysis*, 13:76–83, 1976.

- [284] C. F. Van Loan. Computing the CS and the generalized singular value decompositions. *Numerische Mathematik*, 46:479–491, 1985.
- [285] R. S. Varga. On smallest isolated Gershgorin disks for eigenvalues. *Numerische Mathematik*, 6:366–376, 1964.
- [286] R. C. Ward. The combination shift QZ algorithm. *SIAM Journal on Numerical Analysis*, 12:835–853, 1975.
- [287] R. C. Ward. Balancing the generalized eigenvalue problem. *SIAM Journal on Scientific and Statistical Computing*, 2:141–152, 1981.
- [288] D. S. Watkins. *Fundamentals of Matrix Computations*. John Wiley & Sons, New York, 1991.
- [289] D. S. Watkins. Forward stability and transmission of shifts in the QR algorithm. *SIAM Journal on Matrix Analysis and Applications*, 16:469–487, 1995.
- [290] D. S. Watkins. On the transmission of shifts and shift blurring in the QR algorithm. *Linear Algebra and Its Applications*, 241–243:877–896, 1996.
- [291] D. S. Watkins. Performance of the QZ algorithm in the presence of infinite eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 22:364–375, 2000.
- [292] D. S. Watkins and L. Elsner. Convergence of algorithms of decomposition type for the eigenvalue problem. *Linear Algebra and Its Applications*, 143:19–47, 1991.
- [293] K. Weierstrass. Zur Theorie der bilinearen und quadratischen Formen. *Monatshefte Akademie Wissenschaften Berlin*, pages 310–338, 1868.
- [294] H. Wielandt. Beiträge zur mathematischen Behandlung komplexer Eigenwertprobleme V: Bestimmung höherer Eigenwerte durch gebrochene Iteration. Bericht B 44/J/37, Aerodynamische Versuchsanstalt Göttingen, Germany, 1944.
- [295] J. H. Wilkinson. The use of iterative methods for finding the latent roots and vectors of matrices. *Mathematical Tables and Other Aids to Computation*, 9:184–191, 1955.
- [296] J. H. Wilkinson. The calculation of eigenvectors by the method of Lanczos. *Computer Journal*, 1:148–152, 1959.
- [297] J. H. Wilkinson. Householder's method for the solution of the algebraic eigenvalue problem. *Computer Journal*, 3:23–27, 1960.
- [298] J. H. Wilkinson. Note on the quadratic convergence of the cyclic Jacobi process. *Numerische Mathematik*, 4:296–300, 1962.

- [299] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1963.
- [300] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965.
- [301] J. H. Wilkinson. Convergence of the LR, QR, and related algorithms. *Computer Journal*, 8:77–84, 1965.
- [302] J. H. Wilkinson. Global convergence of tridiagonal QR algorithm with origin shifts. *Linear Algebra and Its Applications*, 1:409–420, 1968.
- [303] J. H. Wilkinson. Note on matrices with a very ill-conditioned eigenproblem. *Numerische Mathematik*, 19:176–178, 1972.
- [304] J. H. Wilkinson. Kronecker’s canonical form and the QZ algorithm. *Linear Algebra and Its Applications*, 28:285–303, 1979.
- [305] J. H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation. Vol. II Linear Algebra*. Springer-Verlag, New York, 1971.
- [306] K. Wu and H. Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22:602–616, 2000.
- [307] T. J. Ypma. Historical development of the Newton–Raphson method. *SIAM Review*, 37:531–551, 1995.

# INDEX

Underlined page numbers indicate a defining entry. Slanted page numbers indicate an entry in a notes and references section. The abbreviation *me* indicates that there is more information at the main entry for this item. Only authors mentioned explicitly in the text are indexed.

- 1-norm, *see* norm  
2-norm, *see* norm
- $\bar{A}$  (conjugate of  $A$ ), 422  
 $A^H$  (conjugate transpose of  $A$ ), 422  
 $A^{-H}$  (conjugate transpose inverse of  $A$ ), 422  
 $A^{-1}$  (inverse of  $A$ ), 422  
 $A^T$  (transpose of  $A$ ), 422  
 $A^{-T}$  (transpose inverse of  $A$ ), 422  
Abel, N. H., 55  
absolute norm, *see* norm  
 $|A|$  (absolute value of  $A$ ), 423  
accumulation of transformations, 76  
Aitken, A. C., 69, 110  
 $A \leq B$ ,  $A < B$ , etc. (componentwise comparison), 423  
algebraic multiplicity, *see* eigenvalue angle, *see* canonical angle  
approximate Newton method, 396  
    analysis, 400–402  
    and inexact Newton method, 419  
    constant shift approximation, 403  
    correction equation, 397, 420  
        inexact solution, 403–404  
        solution, 398  
    correction formula, 397  
    derivation, 396–397  
    diagonally dominant  $A$ , 403  
    drawbacks, 404–405  
    equivalence of correction formula and equation, 398  
    error recurrence, 402  
    inner and outer iterations, 403
- local convergence, 402  
    convergence rates, 402  
natural approximation, 403  
orthogonal correction, 399  
    nomenclature, 399
- Arnoldi decomposition, 297, 299, 315  
alternate form, 300  
Arnoldi factorization, 315  
computation, *see* Arnoldi method  
computation of residuals, 336  
hat convention, 300  
QR factorization of Krylov sequence, 298  
Rayleigh quotient, 301  
reduced, 299, 302  
termination of Krylov sequence, 300  
uniqueness, 300  
uniqueness of starting vector, 301
- Arnoldi factorization, *see* Arnoldi decomposition  
Arnoldi method, 117, 344  
    Arnoldi process, 302, 303–304  
    block, 344  
    comparison of implicit and Krylov–Schur restarting, 332  
    computation of residuals, 336  
    convergence criteria, 334–336, 347  
        choice of tolerance, 341–342  
        shift-and-invert enhancement, 336  
    deflation, 337–340, 347  
        advantages, 337  
        in Arnoldi decomposition, 340  
        in Krylov–Schur method, 340  
    nonorthogonal bases, 339–340

- residual norm, 337–338
- stability, 338–339
- equivalence of implicit and Krylov–Schur restarting, 331
- filter polynomial, 317, 345
  - also see* filter polynomial
- implicit restarting, 318, 345
  - contraction phase, 318–322
  - double shift method, 320
  - exact (Rayleigh quotient) shifts, 324–325
  - example, 323
  - forward instability, 325, 346
  - operation count, 320–323
  - overview, 318
  - the cycle, 322
  - truncation index, 320
- Krylov–Schur restarting, 325–326, 346
  - exchanging eigenblocks, 326–328, *me*
  - operation count, 329
  - the cycle, 328–329
- loss of orthogonality, 117
- operation count, 304
- reorthogonalization, 303
- restarted, 316, 344–345
  - rationale, 316
  - also see* Arnoldi method, implicit restarting; Arnoldi method, Krylov–Schur restarting
- shift-and-invert enhancement, 305
  - convergence criteria, 336
  - stability, 334, 347
  - stagnation of Krylov subspace, 305–306, 316
- stability, 332–333, 346–347
  - shift-and-invert enhancement, 334, 347
  - storage requirements, 304–305, 316
- Arnoldi, W. E., 315, 344
- ARPACK, 325, 345
  - convergence criteria, 341
- arrowhead matrix, 201
- B* inner product, 371
- Cauchy inequality, 371
- norm, 371
- orthogonal matrix, 372
- orthogonality, 230, 372
- orthogonalization, 372–373
  - economizing *B* products, 372–373
  - maintaining orthogonality, 373
- orthonormal matrix, 372
- symmetry, 371
- B*-Arnoldi method, 373
  - deflation, 374
  - orthogonalization, 373
  - residual norms, 373–374
  - restarting, 374
  - Ritz pairs, 373
- B*-Lanczos method, 374–375
  - periodic reorthogonalization, 375
  - restarted, 375
- backward error
  - and unitary similarities, 10
  - as paradigm, 70
  - convergence criteria, 62
  - from residual, 61, 70, 196, 253, 265, 334–335, 369
  - in convergence testing, 10
  - inverse power method, 69
- backward stability
  - and complex eigenvalues of real matrices, 126
  - Arnoldi method, 332–333, 346–347
  - band tridiagonalization, 189
  - bidiagonal QR step, 220
  - deflation in Arnoldi method, 338–339
  - divide-and-conquer algorithm for the spectral decomposition, 183
  - double shift QR algorithm, 126
  - eigenvector computation, 102
  - Hessenberg QR algorithm, 94
  - Householder’s reduction to tridiagonal form, 162, 170
  - implicit tridiagonal QR step, 167
  - inertia of a tridiagonal matrix, 192–193
  - Lanczos algorithm with full orthogonalization, 351
  - QZ algorithm, 152
  - reduction to bidiagonal form, 217
  - reduction to Hessenberg-triangular form, 147
  - spectral decomposition updating, 179

- stability in the usual sense, 87  
 subspace iteration, 388  
 Wilkinson's contribution, 111
- Bai, Z., 23, 129, 346, 395  
 balancing, 107–108, 112  
     and grading, 110  
     matrix pencil, 152–153, 156  
     operation count, 108
- Bartels, R. H., 24
- Bau, D., 23
- Bauer, F. L., 394
- Bellman, R., 23
- Beltrami, E., 226
- Bhatia, R., 52
- bi-Lanczos algorithm, 283, 315, 367  
     look-ahead recurrence, 367
- bidiagonal matrix, 215  
     complex bidiagonal matrix to real, 217  
     reduction to, 215–217, 227  
         first column of the transformation, 217  
         operation count, 217  
         stability, 217  
     relative stability of singular values, 217, 223
- bidiagonal QR algorithm, 217, 227–228  
     combined with QR decomposition, 226, 228  
     operation count, 226  
     pivoting, 228  
     deflation, 224  
     detecting negligible superdiagonal elements, 223–224, 227  
     graded matrices, 224, 227–228  
     QR step, 219–220  
         operation count, 220  
         stability, 220  
     shift computation, 222–223  
         zero shift, 227
- biorthogonal bases, 245
- Björck, Å., 264
- block Krylov subspace, *see* Krylov subspace
- block triangularization of nearly block triangular matrix, 255
- Braman, K., 129
- Bunch, J. R., 201
- Byers, R., 129
- $\mathbb{C}$  (complex numbers), 421  
 $\mathbb{C}^n$  (complex  $n$ -space), 421  
 $\mathbb{C}^{m \times n}$  (space of complex  $m \times n$  matrices), 421
- canonical angle, 248, 264  
     angles between right and left eigenspaces, 251  
     between two vectors, 44  
     computation, 49  
     characterization of largest angle, 249  
     computation, 249, 264  
     of a combination of subspaces, 250–251  
     of a vector and a subspace, 250  
     subspaces of unequal dimensions, 249–250
- Cauchy, A., 23
- Cayley, A., 23
- Chan, T. F., 228
- Chandrasekaran, S., 235
- characteristic equation, 4, 24  
 characteristic polynomial, 4, 24  
     and companion matrix, 55  
     and terminating Krylov sequences, 279  
     in Lanczos's method, 315  
     matrix of order two, 4–5  
 characteristic value, 23
- Chatelin, F., 23, 279
- Chebyshev polynomial, 271, 280  
     as filter polynomial, 317  
     in complex plane, 280  
     in subspace iteration, 392
- Cholesky algorithm, 157
- Cholesky decomposition, 231, 426  
     pivoted, 234  
     updating, 171
- chordal distance, 138, 155
- Clint, M., 394
- column space ( $\mathcal{R}(X)$ ), 424
- column- and row-oriented algorithms, 102, 162
- companion matrix, 55
- complete system of eigenvectors, 7  
     and diagonalization, 8–9
- condition

- condition number, 48  
     limitations, 48  
 eigenvalue, 48, 53  
     condition number, 48  
     Hermitian matrix, 42–43, 51  
 eigenvector, 48–51, 53  
     condition number, 50  
     Hermitian matrix, 51–52  
     *also see* sep  
 generalized eigenvalue, 140  
 generalized eigenvector, 143  
 ill conditioning, 48  
 simple eigenblock, 260  
 simple eigenspace, 261  
 singular values, 206  
 singular vector, 209–210  
     condition number, 210  
 S/PD generalized eigenvalue, 231  
     well conditioning, 48  
 condition number, *see* condition  
 congruence transformation, 190  
 conjugate gradient method, 279  
 consistent norm, *see* norm  
 convergence ratios, 34  
 convergence, normwise and  
     componentwise, 30  
 Crawford, C. R., 236  
 cross-product algorithm for the SVD,  
     210–211  
     assessment, 214, 227  
     attractive features, 211  
     inaccuracy in the singular values,  
         211–212  
     inaccuracy in the left singular  
         vectors, 214  
     inaccuracy in the right singular  
         vectors, 212–213  
     refined Ritz vector, 291–292, 296  
     use by statisticians, 227  
 cross-product matrix, 205  
 CS decomposition, 236–237  
     algorithms, 237  
 Cullum, J., 280, 367  
 Cuppen, J. J. M., 201  
 Curtis, A. R., 156  
  
 Datta, B. N., 23  
 Davidson's algorithm, 419  
  
 Davidson, E. R., 419  
 Davis, C., 53, 265  
 defectiveness, 384  
     and Krylov subspaces, 276  
     dependence of eigenvectors, 8  
     dominant eigenvalues, 34  
     eigenvalue, 7  
     matrix, 7, 15  
     sensitivity of eigenvalues, 7–8, 38  
 definite generalized eigenvalue problem,  
     235–236  
     rotating to make  $B$  positive definite,  
         236  
 deflation, 12, 344  
     by inspection, 106–107  
     operation count, 107  
 complex eigenvector, 114  
     *also see* Hessenberg QR algorithm;  
         power method; QR algorithm;  
         etc.  
 Dembo, R. S., 419  
 Demmel, J. W., 23, 129, 227, 346  
 Dennis, J. E., 265, 418  
 $\det(A)$  (determinant of  $A$ ), 422  
 Dhillon, I. S., 202  
 $\text{diag}(\delta_1, \dots, \delta_n)$  (diagonal matrix), 423  
 diagonalization  
     block, 15, 19–20  
     assessment, 25  
     uniqueness, 20  
     complete system of eigenvectors, 8–9  
     distinct eigenvalues, 20  
 $\dim(\mathcal{X})$  (dimension of  $\mathcal{X}$ ), 424  
 divide-and-conquer algorithm for the  
     spectral decomposition,  
         181–183, 201  
     depth of the recursion, 183, 185  
     error analysis, 201  
     generalities, 181–182  
     operation count, 185  
     recursive vs. direct approach, 183  
     stability, 183  
     the effects of deflation, 185  
     unsuitability for graded matrices, 183  
 divide-and-conquer algorithms, 181–182  
 dominant eigenpair, 31  
 dominant eigenvalue, 31  
     defective, 34

- dominant eigenvector, 31  
     power method, 56
- Donath, W. E., 280, 367
- Dongarra, J. J., 23, 201
- Duff, I. S., 395
- $e_i$  ( $i$ th unit vector), 423
- eigenbasis, *see* eigenspace
- eigenblock, *see* eigenspace
- eigenpair, 2, 24  
     complete system, 7, *me*  
     diagonal matrix, 3  
     dominant, 31, *me*  
     eigenvalue, 2  
     eigenvector, 2  
     left, 2  
     normalization, 2  
     perturbation theory  
         first-order, 45–46, 53  
         rigorous bounds, 46–47, 53
- real matrix, 5–6
- right, 2
- simple, 8
- singular matrix, 3
- triangular matrix, 3  
     *also see* eigenspace
- eigenspace, 22, 115, 240, 247  
     deflation, 243  
     nonorthogonal, 243
- eigenbasis, 127, 242
- eigenblock, 242
- eigenpair, 242  
     behavior under similarity  
         transformations, 242
- left, 242
- orthonormal, 242
- right, 242
- examples, 240
- existence, 242
- left, 242  
     as orthogonal complement of right  
     eigenspace, 243
- nonuniqueness, 244
- representation by a basis, 240–241
- representation of  $A$  with respect to a  
     basis, 241
- residual, 251, 334–335  
     backward error, 253, 265
- optimal, 252, 264
- residual bounds for eigenspaces of  
     Hermitian matrices, 262–263,  
     265
- residual bounds for eigenvalues of  
     Hermitian matrices, 254,  
     263–264, 265
- simple, 244, *me*  
     *also see* simple eigenspace
- eigenvalue, 2  
     algebraic multiplicity, 4, 6, 12  
     block triangular matrix, 5  
     characteristic equation, 4, *me*  
     characteristic polynomial, 4, *me*  
     continuity, 37–38  
     defective, 7, *me*  
     diagonal matrix, 3  
     dominant, 31, *me*  
     Elsner's theorem, 38, 52  
     geometric multiplicity, 6, 12  
     Gershgorin's theorem, 39, *me*  
     Hermitian matrix, 13  
     history and nomenclature, 23–24  
     multiple, 2, *me*  
     nilpotent matrix, 12–13  
     normal matrix, 14  
     perturbation theory, 52  
         condition, 48, *me*  
         derivatives, 47  
         first-order, 45–46, 53  
         rigorous bounds, 46–47, 53  
         simple eigenvalue, 38
- simple, 8
- singular matrix, 3
- skew Hermitian matrix, 14
- triangular matrix, 3
- unitary matrix, 14  
     *also see* eigenpair; Hermitian matrix;  
         similarity transformation
- eigenvector, 2  
     complete system, 7, *me*  
     computing eigenvectors, 100  
     diagonal matrix, 3  
     dominant, 31, *me*  
     Hermitian matrix, 13  
     history and nomenclature, 23–24  
     left, 4  
     left and right, 44, 52–53

- normal matrix, 14  
 normalization, 2  
 null space characterization, 6  
 perturbation theory  
     condition, 48–50, *me*  
     first-order, 45–46, 53  
     rigorous bounds, 46–47, 53  
 real matrix, 5–6  
 right, 4  
 simple, 8  
 singular matrix, 3  
 triangular matrix, 3  
*also see* eigenpair; Hermitian matrix;  
     similarity transformation  
 eigenvectors of a symmetric band matrix,  
     195–200, 202  
     attaining a small residual, 196–197  
     convergence, 198–200  
     limitations of the algorithm, 200  
     loss of orthogonality, 197  
     orthogonalization, 197–198  
         Gram–Schmidt algorithm, 198  
         tradeoffs in clustering, 200  
     residual, 195–196  
     solution of inverse power equation,  
         200  
     starting vector, 200  
 Eisenstat, S. C., 201, 228, 419  
 EISPACK  
      $2 \times 2$  blocks in QZ algorithm, 156  
     eigenvalues by bisection, 202  
 elementary reflector, *see* Householder  
     transformation  
 Elsner’s theorem, 38, 52, 285  
 Elsner, L., 52, 111  
 $\epsilon_M$  (rounding unit), 27  
 Ericsson, T., 347, 379, 380  
 exchanging eigenblocks, 326–328,  
     345–346  
     stability, 328  
 exponent exception  
     computing bidiagonal QR shift, 223  
     computing inertia, 193  
     eigenvector computation, 102  
     implicit double shift, 119  
     plane rotation, 89  
 factorization, 67  
 Fernando, K. V., 228  
 filter polynomial, 317, 345  
     Chebyshev polynomial, 317, 365  
     in Arnoldi method, 317  
     in Lanczos algorithm, 352, 365  
     Leja points, 365  
     Ritz values as roots, 317  
 Fischer’s theorem, 41  
 fl (floating-point evaluation), 27  
 flam (floating-point add and multiply), 58  
     in complex arithmetic, 86–87  
 frot, 91  
 Fokkema, D. R., 346  
 Francis, J. G. F., 110–112, 128, 170  
 Frayssé, V., 380  
 Freund, R. W., 296  
 Frobenius norm, *see* norm  
 Gaches, J., 380  
 Galerkin method, *see* Rayleigh–Ritz  
     method  
 Gaussian elimination, 157  
     and inertia, 191  
     band matrix, 200  
 generalized eigenvalue problem, 129, 296  
     background, 155  
     characteristic polynomial, 133  
     chordal distance, 138, 155  
     eigenpair, 130  
         left, 130  
         perturbation of, 136–137  
         right, 130  
     eigenvalue, 130  
         algebraic multiplicity, 133  
         condition, 140  
         first-order expansion, 137  
         perturbation bound, 140  
         projective representation, 134, 231  
     eigenvector, 130  
         condition, 143  
         first-order expansion, 141–142  
         perturbation bound, 142–143  
     equivalence transformation, 131  
         effect on eigenvalues and  
             eigenvectors, 132  
     generalized Schur form, 132, *me*  
     generalized shift-and-invert  
         transform, 368, *me*

- Hessenberg-triangular form, 144–147, *me*  
infinite eigenvalue, 131  
algebraic multiplicity, 133  
condition, 140  
nonregular pencils, 130–131  
pencil, 130  
perturbation theory, 136, 155  
*also see* generalized eigenvalue problem, eigenpair; generalized eigenvalue problem, eigenvalue; generalized eigenvalue problem, eigenvector  
QZ algorithm, 147–148, *me*  
Rayleigh quotient, 136, 138  
real generalized Schur form, 143, *me*  
reduction to ordinary eigenproblem, 134, 368–369  
regular pair, 131  
regular pencil, 131  
residual, 369  
backward error, 369, 380  
right and left eigenvectors, 135  
shifted pencil, 134–135  
simple eigenvalue, 133  
Weierstrass form, 132  
generalized Schur form, 55, 132, 155  
order of eigenvalues, 133  
real generalized Schur form, 143  
generalized shift-and-invert transform, 368, 379–380  
bounding the residual, 370  
matrix-vector product, 369  
S/PD pencils, 370  
generalized singular value decomposition, 236–237  
computation via the CS decomposition, 236–237  
relation to the S/PD eigenproblem, 236  
geometric multiplicity, *see* eigenvalue  
Gershgorin's theorem, 39, 52, 193  
diagonal similarities, 40–41  
Gershgorin disks, 39  
Gershgorin, S. A., 52  
Givens rotation, 170  
*also see* plane rotation  
Givens, W., 170, 201  
Goldstine, H. H., 203  
Golub, G. H., 23, 24, 201, 227, 264, 367  
graded matrix, 108, 113, 200  
and balancing, 110  
and the QR algorithm, 108–110  
bidiagonal QR algorithm, 227–228  
Jacobi's method, 203  
types of grading, 108  
Gram–Schmidt algorithm, 198, 303  
*B* variants, 372  
*gsreorthog*, 198, 304  
modified, 307, 359  
Grcar, J. F., 365  
Greek-Latin equivalents, 421  
Grimes, R. G., 367  
*gsreorthog*, *see* Gram–Schmidt algorithm  
Gu, M., 201, 228  
Gutknecht, M. H., 367  
Hadamard, M. J., 110  
Handbook for Automatic Computation:  
Linear Algebra, 112  
balancing for eigenvalue computations, 112  
eigenvalues by bisection, 202  
eigenvectors by the inverse power method, 202  
Jacobi's method, 203  
singular value decomposition, 227  
S/PD eigenproblem, 235  
harmonic Rayleigh–Ritz method, 293, 296, 411  
computation, 293–294  
from Krylov decomposition, 314–315  
Hermitian matrix, 294  
ordinary eigenproblem, 294  
Rayleigh quotient, 294  
exact eigenspace, 293  
generalized eigenvalue problem, 296  
harmonic Ritz pair, 292  
Hermitian matrix, 296  
Rayleigh quotient preferred to harmonic Ritz value, 294, 336  
screening of spurious eigenpairs, 293  
Hermitian matrix  
eigenvalue, 13, 52  
condition, 42–43

- Fischer's theorem, 41  
 Hoffman–Wielandt theorem, 43  
 interlacing theorem, 42  
 min–max characterization, 41  
 perturbation, 42  
 principal submatrix, 42  
 eigenvector, 13  
 residual bounds for eigenspaces, 262–263, 265  
 residual bounds for eigenvalues, 254, 263–264, 265  
 sep, 51  
 special properties, 157  
*also see* symmetric matrix  
 Hessenberg form, 81  
 Householder reduction, 83–88, 110–111  
   accumulating transformations, 86  
   first column of the accumulated transformation, 87  
   operation count, 86  
   overwriting original matrix, 87  
   stability, 87, 111  
 implicit Q theorem, 116, *me*  
 nonorthogonal reduction, 111  
   of a symmetric matrix, 158  
   uniqueness of reduction, 116  
   unreduced, 116, *me*  
 Hessenberg QR algorithm  
   ad hoc shift, 100, 105  
   applying transformations to deflated problems, 95–96  
   basic QR step, 92–94  
    operation count, 92  
    stability, 94  
   complex conjugate shifts, 115  
   deflation, 94–95, 112  
    aggressive deflation, 105–106, 112, 123  
    applying transformations to deflated problems, 95–96  
    double shift algorithm, 123  
   detecting negligible subdiagonals, 94–95  
    backward stability, 94  
    normwise criterion, 94–95  
    relative criterion, 95  
 double shift algorithm, 123–126  
   2×2 blocks, 129  
   deflation, 123  
   operation count, 123  
   QR step, 120–123  
   stability, 126  
 explicit shift algorithm, 98–100  
   convergence, 98–100  
   eigenvalues only, 100  
   operation count, 100  
   searching for negligible subdiagonal elements, 96–97  
   stability, 98  
 forward instability, 325, 346  
 Francis double shift, 115, 148  
 implementation issues, 112  
 implicit double shift, 87, 110, 118–123, 128  
   general strategy, 118  
   real Francis shifts, 119  
   starting, 118–119  
 implicit Q theorem, 118, *me*  
 invariance of Hessenberg form, 92, 112  
 multishift, 128–129  
 transmission of shifts, 112  
 Wilkinson shift, 97, 112  
   computation, 97–98  
 Hessenberg-triangular form, 144–147, 156  
   operation count, 147  
   stability, 147  
 Higham, D. J., 380  
 Higham, N. J., 24, 228, 380  
 Hilbert, D., 23  
 Hoffman–Wielandt theorem, 43, 169  
 Horn, R. A., 23  
 Horowitz, L. P., 203  
 Hotelling, H., 69  
 Householder transformation, 12, 81, 111  
   product with a matrix, 81–82  
   reduction of a vector to multiple of  $e_1$ , 82–83  
   reduction of a vector to multiple of  $e_n$ , 82–83  
   reduction of real vectors, 83  
   reduction of row vectors, 82  
 Householder, A. S., 23, 111, 279  
 hump, the, 34

- $I, I_n$  (identity matrix), 423  
 $f$  (cross matrix), 423  
implicit Q theorem, 116, 118  
inertia, 190, 201  
and LU decomposition, 191  
computed by Gaussian elimination,  
191  
invariance under congruence  
transformations, 190  
of a tridiagonal matrix, 191–193  
operation count, 192  
stability, 192–193  
inf( $X$ ) (smallest singular value of  $X$ ), 205  
 $\infty$ -norm, *see* norm  
interlacing theorem, 42  
interval bisection, 193  
invariant subspace, *see* eigenspace  
inverse power method, 66, 70–71, 170  
backward error, 69  
convergence, 68  
dominant eigenvector, 56  
effects of rounding error, 68–69  
implementation, 67  
residual computation, 66–67  
symmetric band matrix, 195–200  
use of optimal residuals, 71  
Ipsen, I. C. F., 70, 265
- Jacobi's method, 202–203  
and graded matrices, 203  
as preprocessing step, 202  
convergence, 202  
cyclic method, 203  
quadratic, 203  
parallelism, 203  
rediscovery, 202  
threshold method, 203  
Jacobi, C. G. J., 201, 202, 403, 420  
Jacobi–Davidson method, 296, 381, 396,  
404, 405, 419–420  
compared with approximate Newton  
method, 406  
convergence testing, 408–409  
convergence to more than one vector,  
407  
retarded by inexact solution,  
417–418  
correction equation, 420
- Davidson's algorithm, 419  
derivation from approximate Newton  
method, 404–405  
exact solution of correction equation,  
415–416  
restriction on choice of shifts,  
415–416  
extending partial Schur  
decompositions, 407–411  
focal region, 405  
harmonic Ritz vectors, 411–412  
Hermitian matrices, 412–414, 420  
iterative solution of correction  
equation, 416–418  
free choice of shifts, 416–417  
Krylov sequence methods, 416  
preconditioning, 416  
stopping, 417–418  
Olsen's method, 419–420  
operation count, 409  
orthogonalization, 409  
partial Schur decomposition, 407  
residual computation, 408–409  
restarting, 411  
with harmonic Ritz vectors, 412  
shift parameters, 405–406  
solving projected systems, 414–415  
ill conditioning, 415  
multiple right-hand sides, 414–415  
storage requirements, 409  
updating the Rayleigh quotient, 409
- Jennings, A., 394, 395  
Jensen, P. S., 347, 380  
Jia, Z., 295, 296  
Jiang, E., 170, 265  
Johnson, C. R., 23  
Jordan block, *see* Jordan canonical form  
Jordan canonical form, 21, 24–25, 36  
assessment, 22, 25  
Jordan block, 7, 20  
powers of, 34–35  
principal vector, 22  
sensitivity of eigenvalues, 38  
uniqueness, 21  
Jordan, C., 155, 226, 264
- $K_k(A, u)$  (Krylov matrix), 267  
 $K_k(A, u)$  (Krylov subspace), 267

- Kahan, W., 53, 156, 170, 201, 227, 264, 265  
 Kaniel, S., 280  
 Kato, T., 52  
 Kline, M., 23  
 König, J., 110  
 Kogbetliantz, E. G., 228  
 Kollerstrom, N., 418  
 Krause, G., 52  
 Kronecker product, 24  
 Kronecker, L., 155  
 Krylov decomposition, 297, 309, 316
  - computation of harmonic Ritz vectors, 314–315
  - computation of refined Ritz vectors, 314
  - computation of residuals, 335
  - economics of transforming, 312–314
  - equivalence, 309
    - to an Arnoldi decomposition, 311–312
  - nomenclature, 310
  - orthonormal, 309
  - Rayleigh quotient, 309
  - reduction to Arnoldi form, 312–314
  - similarity, 310
    - translation, 311
- Krylov subspace, 267, 279
  - and defective matrices, 276
  - and scaling, 268
  - and similarity transformations, 268
  - block, 277, 280, 367
    - computation, 277
    - convergence, 277
    - non-Hermitian matrices, 278
  - block Krylov sequence, 277
  - compared with power method, 266
  - convergence, 269, 279–280, 315
    - assessment of polynomial bounds, 274
    - multiple eigenvalues, 273–274
    - polynomial bounds, 269–272
    - square root effect, 272–273
    - to inferior eigenvector, 273
    - to interior eigenpairs, 273
    - to the subdominant eigenvector, 266
  - degeneracy in Krylov sequence, 298
- elementary properties, 267–268  
 invariance under shifting, 268  
 Krylov matrix, 267  
 Krylov sequence, 266, 279
  - multiple eigenvalues, 277
    - convergence, 273–274
    - deflation, 278
  - non-Hermitian matrices
    - convergence, 274–275, 280
    - polynomial bounds, 275–276
  - polynomial representation, 268
  - termination, 268
    - and characteristic polynomial, 279
    - and eigenspaces, 269
  - uniqueness of starting vector, 301
- also see* Arnoldi decomposition; Krylov decomposition; Lanczos algorithm
- Krylov, A. N., 279  
 Kublanovskaya, V. N., 110
- Lagrange, J.-L., 23  
 $\Lambda(A)$  (the spectrum of  $A$ ), 2  
 Lanczos algorithm, 279, 315
  - adjusted Rayleigh quotient, 351
    - updating, 362–363
  - and characteristic polynomial, 315
  - and Rayleigh–Ritz method, 348
  - as iterative method, 365
  - block, 280, 367
  - computing residual norms, 363
  - estimating loss of orthogonality, 356–358
  - filter polynomial, 352
    - Chebyshev polynomial, 365
    - Leja points, 365
  - full reorthogonalization, 349, 365
    - convergence and deflation, 352
    - error analysis, 365
  - essential tridiagonality of Rayleigh quotient, 351
  - implicit restarting, 351–352, 365
  - Krylov-spectral restarting, 352, 365
  - stability, 351
  - global orthogonality, 350
  - Lanczos decomposition, 297, 307
    - Rayleigh quotient, 306
    - uniqueness, 307

- Lanczos recurrence, 307–308, 348  
 alternative, 307, 315  
 loss of orthogonality, 308  
 operation count, 308  
 storage considerations, 308
- local orthogonality, 350  
 loss of orthogonality, 348  
 local and global, 350
- method of minimized iterations, 279
- no reorthogonalization, 349, 366  
 software, 366
- partial reorthogonalization, 350,  
 365–366  
 software, 366
- periodic reorthogonalization, 350,  
 358–362, 365–366  
 example, 364  
 modified Gram–Schmidt algorithm,  
 359  
 software, 366
- Rayleigh quotient, 347  
 contamination by  
 reorthogonalization, 350
- reorthogonalization, 348, 365  
 and semiorthogonality, 358–359  
 contamination of the Rayleigh  
 quotient, 350  
 difficulties, 348–349  
 general algorithm, 349–350  
 updating adjusted Rayleigh  
 quotient, 362–363
- also see* Lanczos algorithm, full  
 reorthogonalization; Lanczos  
 algorithm, no  
 reorthogonalization; Lanczos  
 algorithm, partial  
 reorthogonalization; Lanczos  
 algorithm, periodic  
 reorthogonalization; Lanczos  
 algorithm, selective  
 reorthogonalization
- restarted, 280
- selective reorthogonalization, 350,  
 365–366
- semiorthogonality, 353, 365–366  
 and reorthogonalization, 358–359  
 effects of relaxing constraint, 353
- QR factorization, 353–354
- Rayleigh quotient, 355  
 residual norms, 355–356  
 Ritz vectors, 355–356  
 size of reorthogonalization  
 coefficients, 354
- singular value decomposition,  
 366–367
- termination and restarting, 350  
 thick restarting, 365
- Lanczos decomposition, *see* Lanczos  
 algorithm
- Lanczos recurrence, *see* Lanczos algorithm
- Lanczos, C., 279, 315, 365
- LAPACK, 112  
 2×2 blocks in QZ algorithm, 156  
 2×2 blocks in double shift QR  
 algorithm, 129  
 ad hoc shift in QR algorithm, 105  
 bidiagonal QR algorithm, 228  
 CS decomposition, 237  
 differential qd algorithm, 228  
 divide-and-conquer algorithm for the  
 spectral decomposition, 183  
 eigenvalues by bisection, 202  
 eigenvectors by the inverse power  
 method, 202
- eigenvectors of triangular matrices,  
 102
- exchanging eigenblocks, 328, 346
- multishift QR algorithm, 128, 129
- PWK method for the tridiagonal QR  
 algorithm, 171
- reduction to tridiagonal form, 170
- S/PD generalized eigenvalue  
 problem, 235
- spectral decomposition downdating,  
 201
- storage of band matrices, 187
- tridiagonal QR step, 170
- Wilkinson’s algorithm, 233
- large matrices, 239  
 storage of eigenvectors, 239  
 storage of sparse matrices, 239
- latent value, 24
- Le, J., 346
- left eigenpair, 2
- Lehoucq, R. B., 345, 346
- Lewis, J. G., 367

- Li, C. K., 235  
 local convergence ratio, *see* convergence ratios  
 LR algorithm, 110, 394  
     convergence, 111  
 LU decomposition, 193, 425  
     and inertia, 191  
 Lui, Z. A., 367  
 Martin, R. S., 113  
 Mathias, R., 129, 235, 265  
 matrix, 421  
     cross, 423  
     diagonal, 423  
     element, 421  
     full rank, 425  
     Hermitian, 423  
     history, 23  
     identity, 423  
     order, 422  
     orthogonal, 423  
     orthonormal, 423  
     permutation, 423  
     sum and product, 422  
     symmetric, 423  
     trapezoidal, 424  
     triangular, 424  
     unit triangular, trapezoidal, 424  
     unitary, 423  
     zero, 423  
 matrix powers ( $A^k$ ), 33–36  
     behavior when computed, 36  
     spectral radius bounds, 33  
     the hump, 34  
 matrix-vector product  
     effects of rounding error, 65  
     tridiagonal matrix, 58–59  
 Meerbergen, K., 380  
 method of minimized iterations, 279  
 Moler, C. B., 156  
 Moon, Y. S., 236  
 Moré, J. J., 265  
 Morgan, R. B., 296, 345  
 multiple eigenvalue, 2  
     algebraic multiplicity, 4, 6, 12  
     geometric multiplicity, 6, 12  
 multiset, 2  
 Murray, F. J., 203  
 $\mathcal{N}(X)$  (null space of  $X$ ), 424  
 Nash, S., 24  
 Newton's method, 395  
     application to eigenproblem, 418–419  
     derivation by perturbation expansion, 395–396  
     history, 418  
     relation to Rayleigh quotient method, 418  
     relation to the QR algorithm, 403, 418–419  
 Newton, I., 418  
 Newton-based methods, 381  
 Nielsen, C. P., 201  
 nilpotent matrix, 12–13  
 nonnormal matrix, 71  
 norm, 25, 36  
     1-norm, 26, 27  
     2-norm  
         and Frobenius norm, 27  
         and largest singular value, 205  
         matrix norm, 27  
         of orthonormal matrix, 27  
         properties, 27  
         unitary invariance, 27  
         vector norm, 26  
     absolute norm, 28  
     examples, 28  
     and spectral radius, 31–33, 36  
     comparable norm, 36  
     consistent norm, 29, 36  
         consistent vector norm, 29–30  
         construction by similarities, 30  
         examples, 29  
     equivalence of norms, 30  
     Euclidean norm, 26  
     family of norms, 26  
     Frobenius norm, 26  
         and 2-norm, 27  
         and singular values, 205  
         trace characterization, 26  
         unitary invariance, 27  
      $\infty$ -norm, 26, 27  
     limitations, 28, 33, 35–36, 43  
     matrix norm, 25  
     operator norm, 27  
     rounding a matrix, 28

- spectral norm, *see* norm, 2-norm  
 subordinate norm, 27  
 triangle inequality, 26  
 unitarily invariant norm, 27  
 vector norm, 26  
 normal matrix, 14  
     eigenvalues, 14  
     eigenvectors, 14  
 Nour-Omid, B., 347, 380  
 null space ( $\mathcal{N}(X)$ ), 424  
 nullity [null( $A$ )], 424  
 Oettli, W., 70  
 Olsen, J., 419  
 operation count  
     application of a plane rotation, 91  
     Arnoldi process, 304  
     balancing, 108  
     band tridiagonalization, 189  
     basic QR step in Hessenberg QR algorithm, 92  
     bidiagonal QR step, 220  
     complex arithmetic, 86–87  
     deflation by inspection, 107  
     divide-and-conquer algorithm for the spectral decomposition, 185  
     divide-and-conquer algorithms, 181–182  
     eigenvectors of a triangular matrix, 102  
     explicit shift Hessenberg QR algorithm, 100  
     Householder reduction to Hessenberg form, 86  
     Householder-matrix product, 82  
     hybrid QRD-SVD algorithm, 226  
     implicit double shift QR step, 123  
     implicit tridiagonal QR step, 167  
     implicitly restarted Arnoldi, 320–323  
     inertia of a tridiagonal matrix, 192  
     Jacobi–Davidson method, 409  
     Krylov–Schur restarting, 329  
     Lanczos recurrence, 308  
     QZ algorithm, 150–152  
     reduction to bidiagonal form, 217  
     reduction to Hessenberg-triangular form, 147  
     reduction to tridiagonal form, 162  
     spectral decomposition updating, 179–181  
     Wilkinson’s algorithm for S/PD eigenproblem, 232  
     orthogonal similarity, *see* unitary similarity  
     Osborne, E. E., 112  
     Ostrowski, A. M., 36, 71  
     overflow, *see* exponent exception  
     Paige, C. C., 237, 280, 296, 315, 366  
     Parlett, B. N., 24, 169, 170, 228, 265, 279, 295, 296, 346, 347, 365, 367, 379, 380  
     -pencil, *see* generalized eigenvalue problem  
     pentadiagonal matrix, 186  
     perturbation theory, *see* eigenpair; eigenvalue; eigenvector; generalized eigenvalue problem; Hermitian matrix; relative and structured perturbation theory; simple eigenspace; singular value decomposition; S/PD generalized eigenproblem  
     Peters, G., 418, 420  
     Petrov method, *see* Rayleigh–Ritz method  
     plane rotation, 88, 111  
         and Wilkinson diagrams, 91  
         application, 89–91  
         operation count, 91  
         pre- and postmultiplication, 91  
     exponent exception, 89  
     generation, 89  
     in the  $(i, j)$ -plane, 88  
     introducing a zero into a 2-vector, 88  
     introducing a zero into a matrix, 88–89  
     plane rotations  
         Givens’ method, 170  
     positive definite matrix, 425  
     positive semidefinite matrix, 425  
     Powell, M. J. D., 70  
     power method, 56, 69–70, 266, 381  
         assessment, 66  
         choice of starting vector, 58, 60  
         convergence, 57  
         convergence criteria, 60–62  
         deflation, 70  
         effects of rounding error, 65

- implementation, 64
- Markov chain, 69
- normalization, 59
- operation counts, 58–59
- rate of convergence, 58
- Rayleigh quotients, 63
- shifting, 64, 70
  - with two vectors, 381–382
- Prager, W., 70
- primitive Ritz vector, *see* Rayleigh–Ritz method
- principal vector, *see* Jordan canonical form
- projector, 427
- proper value, 24
- pseudospectrum, 36–37
- qd-algorithm, 110
- QL algorithm, *see* QR algorithm
- QR algorithm, 1, 24, 55, 394, 396
  - as factorization of a polynomial, 76
  - convergence, 111
    - alternate shifts, 75
    - cubic, 75
    - error bounds for a QR step, 73–74
    - multiple eigenvalues, 75
    - quadratic, 75
    - unshifted algorithm, 77–80
  - history, 110
  - impracticality on a full matrix, 80–81
  - QL algorithm, 111, 113, 169, 170
  - QR step, 71
  - Rayleigh quotient shift, 73
  - relation to inverse power method, 72–73, 111
  - relation to Newton’s method, 403, 418–419
  - relation to power method, 72, 77, 111
  - shifted, 71
  - unshifted, 77–80
    - convergence to Schur decomposition, 78–79
    - disorder in eigenvalues, 79
    - equimodular eigenvalues, 80
    - implications for the shifted algorithm, 80
    - rates of convergence, 80
  - variants (RQ, QL, LQ), 111, 113, 169, 170, 418
- Wilkinson shift, 75
- also see* bidiagonal QR algorithm; Hessenberg QR algorithm; QZ algorithm; subspace iteration; tridiagonal QR algorithm
- QR decomposition, 426–427
  - and QR algorithm, 110
- QR step, *see* QR algorithm
- quasi-triangular matrix, 115
  - computing eigenvectors, 126–128
- QZ algorithm, 147–148, 156
  - $2 \times 2$  blocks, 155
  - choice of double shift, 148
  - computation of eigenvectors, 155
  - deflation, 150, 153
  - eigenvalues only, 152
  - operation count, 150–152
  - QZ step, 148–152
  - real shifts, 156
  - stability, 152
  - starting the double shift, 148
  - treatment of infinite eigenvalues, 153–155, 156
- $\mathbb{R}$  (real numbers), 421
- $\mathbb{R}^n$  (real  $n$ -space), 421
- $\mathbb{R}^{m \times n}$  (space of real  $m \times n$  matrices), 421
- $\mathcal{R}(X)$  (column space of  $X$ ), 424
- rank, 424
- Raphson, 418
- rational Krylov transformation, 342–344, 347
  - no free lunch, 343–344
  - potential instability, 343
- Rayleigh (J. W. Strutt), 70, 295
- Rayleigh quotient, 47, 63, 70
  - accuracy of, 260
  - and optimal residuals, 70
  - for generalized eigenvalue problem, 136, 138
  - for subspaces, 252
  - in Arnoldi decomposition, 301
  - in Krylov decomposition, 309
  - in Lanczos decomposition, 306
- Rayleigh quotient method, 69, 71, 396
  - quadratic convergence, 69
  - relation to Newton’s method, 418
- Rayleigh quotient shift, *see* QR algorithm

- Rayleigh–Ritz method, 252, 280, 381, 396  
 and Lanczos algorithm, 348  
 as a generalized eigenvalue problem,  
 283  
 choosing a primitive Ritz basis, 282  
 convergence, 284, 295  
   Hermitian matrix, 286–288  
   residual, 289  
   Ritz blocks and bases, 288  
   Ritz value, 285–287  
   Ritz vector, 286–287  
   uniform separation condition, 287  
 exact eigenspace, 281  
 failure, 282–283  
 Galerkin method, 295–296  
 general procedure, 281  
 harmonic Rayleigh–Ritz method,  
293, me  
 history, 295  
 oblique, 283  
 optimal residual, 284  
 orthogonal, 284  
 Petrov method, 295–296  
 primitive Ritz basis, 282  
 primitive Ritz vector, 282  
 projection method, 295  
 residual bound, 288  
 Ritz basis, 282  
 Ritz block, 282  
 Ritz pair, 282  
 Ritz space, 282  
 Ritz value, 282  
 Ritz vector, 282  
 separation condition, 283  
 two levels of approximation,  
 280–281
- real matrix  
 eigenvectors, 5–6  
 perturbation of simple eigenvalue, 38
- real Schur form, 113, 126, 128  
 computation by double shift QR  
   algorithm, 123–126  
 computation by Householder  
   transformation, 115  
 computing eigenvectors, 126–128  
 in subspace iteration, 391
- refined Ritz vector, 289, 295–296  
 computation, 291–292
- basic algorithm, 291  
 compared with Ritz vector, 291  
 cross-product algorithm, 291–292,  
 296  
 from Krylov decomposition, 314  
 convergence, 289–290  
 use of Rayleigh quotient, 290
- Reid, J. K., 156
- Reinsch, C., 113
- relative and structured perturbation theory,  
 54
- relative error, 28  
 in eigenvalues, 43, 231  
 normwise, 28
- representation of symmetric matrices, 159  
 in two-dimensional arrays, 159  
 packed storage, 159
- residual, 61  
 backward error, 61, 70, 253, 265  
 computation in the inverse power  
 method, 66–67  
 convergence testing, 62, 195–196  
 optimal, 63, 252, 264
- resolvent, 247
- reverse communication, 345
- Rigal, J. L., 380
- right eigenpair, 2
- Ritz pair, *see* Rayleigh–Ritz method
- Ritz, W., 295
- rounding unit ( $\epsilon_M$ ), 27
- row orientation, *see* column- and  
 row-oriented algorithms
- Ruhe, A., 23, 295, 347, 367, 379, 380
- Rutishauser, H., 110, 203, 394, 395
- Saad, Y., 23, 279, 280, 295, 344–347
- scalar, 421
- Schnabel, R. B., 418
- Schur decomposition, 1, 10–12, 24, 55, 71  
 and Jordan form, 22, 24  
 by rowwise deflation, 72  
 computation by explicit shift QR  
   algorithm, 98–100
- computing eigenvectors, *see* triangular  
 matrix, computing eigenvectors
- partial, 407
- Schur vector, 12, *me*  
*also see* real Schur form

- Schur vector, 12  
     relation to eigenvectors, 12, 15
- Schur, I., 24
- Schwarz, H. R., 201
- Scott, D. S., 365, 379
- Scott, J. A., 395
- secular equation, 24
- semidefinite *B*-Lanczos and Arnoldi  
     algorithms, 375–379, 380  
     assessment, 379  
     geometry, 375–376  
     null-space error, 376  
         defective matrix, 380  
         effect on Rayleigh quotient, 376  
         effect on Ritz vectors, 376–377  
         growth, 378, 380  
         purging, 378–379, 380  
         rounding-error analysis, 380
- sep, 46, 53–54, 143, 255  
     and physical separation of  
         eigenvalues, 50–51, 256
- Hermitian matrices, 51  
     properties, 256
- shift-and-invert enhancement, 66  
     factorization, 67  
     solving linear systems, 67–68
- $\sigma_i(X)$  (the *i*th singular value of  $X$ ), 204
- similarity transformation, 8  
     eigenvalues and eigenvectors, 9  
     invariance of trace, 9  
     unitary similarity, 10
- Simon, H., 346, 365–367
- simple eigenspace, 244  
     angles between right and left  
         eigenspaces, 251  
     biorthogonal basis for left  
         eigenspace, 245
- block diagonalization, 244–246  
     several eigenspaces, 245–246
- complementary eigenspace, 245
- corresponding left eigenspace, 245
- deflation  
     of an approximate eigenspace,  
         254–255, 265
- perturbation theory, 265  
     accuracy of the Rayleigh quotient,  
         260, 265  
     alternate bound, 265
- bounds for eigenspace, 261  
     bounds in terms of the error, 259  
     condition of an eigenblock, 260  
     condition of an eigenspace, 261  
     main theorem, 258  
     normalization of bases, 259
- residual bounds, 257  
     limitations, 257
- resolvent, 247
- spectral projector, 247
- spectral representation, 245  
     several eigenspaces, 246  
     standard representation, 245
- uniqueness, 246, 247  
     *also see* eigenspace
- Simpson, 418
- singular value, 204  
     and 2-norm, 205  
     and eigenvalues of the cross-product  
         matrix, 157, 205  
     and Frobenius norm, 205  
     effects of rounding, 211  
     min-max characterization, 206  
     ordering convention, 204  
     perturbation theory, 206, 226–227  
         analogue of Rayleigh quotient, 209  
         condition, 206, *me*  
         first-order expansion, 208  
         second-order expansion for small  
             singular values, 226  
      $\sigma_i(X)$  (the *i*th singular value of  $X$ ),  
         204
- singular value decomposition, 55, 204  
     and 2-norm, 27  
     and spectral decomposition of  
         cross-product matrix, 205  
     differential qd algorithm, 228  
     divide-and-conquer algorithms, 228  
     downdating, 228  
     history, 226  
     Jacobi methods, 228–229  
         one-sided method, 229  
         two-sided method of Kogbetliantz,  
             228
- Lanczos algorithm, 366–367
- singular value, 204, *me*
- singular value factorization, 204
- singular vector, 204, *me*

- uniqueness, 205
- singular value factorization, *see* singular value decomposition
- singular vector, 204
  - and eigenvectors of the cross-product matrix, 157, 205
  - left, 204
  - perturbation theory, 206–210, 226–227
    - condition, 209–210, *me*
    - first-order expansion, 208
    - left singular vector, 210
    - right singular vector, 210
    - separation of singular values, 210
  - right, 204
    - and eigenvectors of the cross-product matrix, 205
- skew Hermitian matrix, 14
  - eigenvalues, 14
- skew symmetric matrix, *see* skew Hermitian matrix
- Sleijpen, G. L. G., 296, 346, 420
- solution of projected systems, 414–415
  - ill conditioning, 415
  - multiple right-hand sides, 414–415
- Sorensen, D. C., 201, 316, 345, 346, 365
- $\text{span}(\mathcal{X})$  (the space spanned by  $\mathcal{X}$ ), 424
- sparse matrix, 58, 239
- SPD generalized eigenproblem, 157, 229
  - $B$ -orthogonality, 230
  - banded problems, 235
  - Chandrasekaran's method, 235
  - congruence transformation, 229
  - diagonalization by congruences, 229
- eigenvalue, 230
  - condition number, 231
  - ill conditioned, 231
  - projective representation, 231
  - well conditioned eigenvalues not determined to high relative accuracy, 231
- eigenvector, 230
- generalized shift and invert transformation, 370
- ill-conditioned  $B$ , 233–235
  - use of pivoted Cholesky decomposition of  $B$ , 234
- use of spectral decomposition of  $B$ , 234
- no infinite eigenvalues, 230
- perturbation theory, 231, 234–235
- product problem ( $AB$ ), 235
- S/PD pencil, 229
- Wilkinson's algorithm, 231–233, 235
  - computation of  $R^{-T}AR^{-1}$ , 232–233
  - operation count, 232
  - storage and overwriting, 232
- spectral decomposition, 13, 158
  - and 2-norm, 27
  - effects of rounding, 211–212
  - updating, *see* spectral decomposition updating
- spectral decomposition updating, 171, 201
  - assessment, 181
  - basic algorithm, 171
  - computing eigenvectors, 177–181
    - stability, 179
    - the naive approach, 177–178
  - deflation, 173–175
    - tradeoffs, 175
  - operation count, 179–181
  - reduction to standard form, 171–173
    - accumulating transformations, 173
  - secular equation, 175
  - solving the secular equation, 176–177
- spectral enhancement, 64
  - shift-and-invert enhancement, 66, *me*
- spectral norm, *see* norm, 2-norm
- spectral projector, 53
- spectral radius, 31
  - norm bounds, 31–33, 36
- spectral representation, *see* simple eigenspace
- spectral transformation, 379
  - also see* generalized shift-and-invert transformation
- spectrum, 2
- Spence, A., 380
- SRR, *see* subspace iteration, Schur–Rayleigh–Ritz refinement
- stability in the usual sense, 87
- Stathopoulos, A., 346
- Steihaug, T., 419

Stewart, G. W., 24, 52, 155, 156, 237, 264, 265, 295, 316, 345, 346, 394, 395  
 Stewart, W. J., 395  
 Sturm sequence, 170, 202  
 subspace iteration, 381, 382, 394  
     and QR algorithm, 385  
     convergence testing, 387–388  
     stability, 388  
 convergence theory, 383  
     Schur–Rayleigh–Ritz refinement, 386  
     superiority to power method, 384  
     to dominant subspaces, 384  
 defectiveness, 384  
 deflation, 388  
 dependence in basis, 382  
 freedom in dimension of the basis, 384  
 general algorithm, 386–387  
 orthogonalization, 382, 384–385  
     frequent with shift and invert enhancement, 390–391  
     when to perform, 389–391, 393–394  
 Rayleigh–Ritz method, 394  
 real Schur form, 391  
 Schur–Rayleigh–Ritz refinement, 382, 386, 394  
     convergence, 386, 394  
     when to perform, 389, 394  
 shift-and-invert enhancement, 390  
 software, 395  
 symmetric matrix, 391  
     Chebyshev acceleration, 392–394, 395  
     economization of storage, 391–392  
*Treppeniteration*, 394  
 Sun, J.-G., 52  
 SVD, *see* singular value decomposition  
 Sylvester’s equation, 2, 16, 24, 128  
     conditions for solution, 16–17  
     Kronecker product form, 24  
     numerical solution, 18, 24  
     Sylvester operator, 16  
 Sylvester, J. J., 24, 201  
 symmetric band matrix, 186

band tridiagonalization, 187–189, 201  
 limitations, 189  
 operation count, 189  
 stability, 189  
 band width, 186  
 computing eigenvalues and eigenvectors, 186  
 eigenvectors, *see* eigenvectors of a symmetric band matrix  
 pentadiagonal matrix, 186  
 representation, 187  
 storage, 186  
 symmetric matrix  
     inertia, 190, *me*  
 representation, 159  
     in two-dimensional arrays, 159  
     packed storage, 159  
 simplifications in the QR algorithm, 158  
 special properties, 157  
     *also see* Hermitian matrix  
 symmetric positive definite generalized eigenvalue problem, *see* S/PD generalized eigenproblem  
 symmetric tridiagonal matrix, *see* tridiagonal matrix  
 systolic array, 203  
 Tang, P. T. P., 201  
 Taussky, O., 52  
 Taylor, D. R., 367  
 $\Theta(\mathcal{X}, \mathcal{Y})$  (canonical angle matrix), 248  
 Toumazou, V., 380  
 trace  
     as sum of eigenvalues, 9  
     invariance under similarity transformations, 9  
     use in debugging, 10  
 trace( $A$ ) (trace of  $A$ ), 422  
 Trefethen, L. N., 23, 37  
 triangle inequality, *see* norm  
 triangular matrix  
     computing eigenvectors, 100–104  
     operation count, 102  
     stability and residuals, 102–104  
     computing left eigenvectors, 104  
 tridiagonal matrix, 158, 186

- assumed symmetric, 158  
calculation of a selected eigenvalue,  
    193–195, 201–202  
accuracy, 195  
use of fast root finders, 195  
Givens' reduction to, 170  
Householder's reduction to, 159–162,  
    170, 189  
first column of the transformation,  
    162  
operation count, 162  
stability, 162, 170  
using symmetry, 160  
Wilkinson's contribution, 170  
inertia, 191–193  
    operation count, 192  
    stability, 192–193  
making a Hermitian matrix real,  
    162–163  
representation, 160  
tridiagonal QR algorithm, 128, 163,  
    170–171  
combination of Rayleigh quotient and  
    Wilkinson shifts, 170  
deflation, 168–169  
detecting negligible off-diagonal  
    elements, 168–169, 170  
    graded matrix, 169  
explicitly shifted, 164  
    graded matrix, 169  
        QL algorithm, 169, 170  
implicitly shifted QR step, 164–167  
    operation count, 167  
    PWK method, 169, 171  
    stability, 167  
    variations, 169  
local convergence, 167–168  
Rayleigh quotient shift, 167, 170  
    Wilkinson shift, 167, 168, 170  
Twain, Mark, 20  
  
underflow, *see* exponent exception  
Underwood, R., 280, 367  
unitarily invariant norm, *see* norm  
unitary matrix  
    eigenvalues, 14  
unitary similarity, 10  
    backward stability, 10  
  
unreduced Hessenberg matrix, 116  
van der Vorst, H. A., 23, 296, 346, 420  
Van Loan, C. F., 23, 24, 237  
vector, 421  
    component, 421  
    unit, 423  
    zero, 423  
von Neumann, J., 203  
Vu, P., 345  
  
Ward, R. C., 156  
Watkins, D. S., 23, 111, 112, 129, 156, 346  
Weierstrass form, 132  
Weierstrass, K., 155  
Wielandt, H., 70  
Wilkinson, 420  
Wilkinson diagram, 81, 424  
Wilkinson shift, *see* Hessenberg QR  
    algorithm; QR algorithm;  
    tridiagonal QR algorithm  
Wilkinson, J. H., 1, 23, 52, 53, 70, 71,  
    111–113, 169, 170, 229, 235,  
    264, 365, 394, 418  
Wu, K., 346, 365  
  
Yang, C., 345  
  
Zhang, Z., 170