

University of Alabama at Birmingham
Department of Mathematics

Numerical Linear Algebra

Lecture Notes for MA 660
(1997–2014)

Dr Nikolai Chernov

Summer 2014

Contents

0. Review of Linear Algebra	1
0.1 Matrices and vectors	1
0.2 Product of a matrix and a vector	1
0.3 Matrix as a linear transformation	2
0.4 Range and rank of a matrix	2
0.5 Kernel (nullspace) of a matrix	2
0.6 Surjective/injective/bijective transformations	2
0.7 Square matrices and inverse of a matrix	3
0.8 Upper and lower triangular matrices	3
0.9 Eigenvalues and eigenvectors	4
0.10 Eigenvalues of real matrices	4
0.11 Diagonalizable matrices	5
0.12 Trace	5
0.13 Transpose of a matrix	6
0.14 Conjugate transpose of a matrix	6
0.15 Convention on the use of conjugate transpose	6
1. Norms and Inner Products	7
1.1 Norms	7
1.2 1-norm, 2-norm, and ∞ -norm	7
1.3 Unit vectors, normalization	8
1.4 Unit sphere, unit ball	8
1.5 Images of unit spheres	8
1.6 Frobenius norm	8
1.7 Induced matrix norms	9
1.8 Computation of $\ A\ _1$, $\ A\ _2$, and $\ A\ _\infty$	9
1.9 Inequalities for induced matrix norms	9
1.10 Inner products	10
1.11 Standard inner product	10
1.12 Cauchy-Schwarz inequality	11
1.13 Induced norms	12
1.14 Polarization identity	12
1.15 Orthogonal vectors	12
1.16 Pythagorean theorem	12
1.17 Orthogonality and linear independence	12
1.18 Orthonormal sets of vectors	13
1.19 Orthonormal basis (ONB)	13
1.20 Fourier expansion	13
1.21 Orthogonal projection	13
1.22 Angle between vectors	14
1.23 Orthogonal projection onto a subspace	14
1.24 Degenerate case	14
1.25 Gram-Schmidt orthogonalization	15

1.26	Construction of ONB	15
1.27	Legendre polynomials (optional)	15
1.28	Orthogonal complement	16
1.29	Orthogonal direct sum	16
1.30	Some useful formulas	16
2.	Unitary Matrices	18
2.1	Isometries	18
2.2	Characterization of isometries - I	18
2.3	Characterization of isometries - II	18
2.4	Characterization of isometries - III	18
2.5	Identification of finite-dimensional inner product spaces	19
2.6	Unitary and orthogonal matrices	19
2.7	Lemma	19
2.8	Matrices of isometries	20
2.9	Group property	20
2.10	Orthogonal matrices in 2D	20
2.11	Characterizations of unitary and orthogonal matrices	21
2.12	Determinant of unitary matrices	21
2.13	Eigenvalues of unitary matrices	21
2.14	Invariance principle for isometries	22
2.15	Orthogonal decomposition for complex isometries	22
2.16	Lemma	23
2.17	Orthogonal decomposition for real isometries	23
2.18	Unitary and orthogonal equivalence	24
2.19	Unitary matrices in their simples form	24
2.20	Orthogonal matrices in their simples form	24
3.	Hermitian Matrices	26
3.1	Adjoint matrices	26
3.2	Adjoint transformations	26
3.3	Riesz representation theorem	27
3.4	Quasi-linearity	27
3.5	Remark	27
3.6	Existence and uniqueness of adjoint transformation	27
3.7	Relation between $\text{Ker } T^*$ and $\text{Range } T$	28
3.8	Selfadjoint operators and matrices	28
3.9	Examples	28
3.10	Hermitian property under unitary equivalence	28
3.11	Invariance principle for selfadjoint operators	29
3.12	Spectral Theorem	29
3.13	Characterization of Hermitian matrices	30
3.14	Eigendecomposition for Hermitian matrices	30
3.15	Inverse of a selfadjoint operator	30
3.16	Projections	31
3.17	Projections (alternative definition)	31

3.18	“Complimentary” projections	31
3.19	Orthogonal projections	32
3.20	Characterization of orthogonal projections	32
4.	Positive Definite Matrices	33
4.1	Positive definite matrices	33
4.2	Lemma	34
4.3	Sufficient condition for positive definiteness	34
4.4	Bilinear forms	34
4.5	Representation of bilinear forms	35
4.6	Corollary	35
4.7	Hermitian/symmetric forms	35
4.8	Quadratic forms	35
4.9	Theorem	35
4.10	Positive definite forms and operators	35
4.11	Theorem	35
4.12	Properties of Hermitian matrices	36
4.13	Eigenvalues of positive definite matrices	37
4.14	Inverse of a positive definite matrix	37
4.15	Characterization of positive definite matrices	38
4.16	Characterization of positive semi-definite matrices	38
4.17	Full rank and rank deficient matrices	38
4.18	Products A^*A and AA^*	38
4.19	Spectral radius	39
4.20	Spectral radius for Hermitian matrices	40
4.21	Theorem on the 2-norm of matrices	40
4.22	Example	42
4.23	Corollary for the 2-norm of matrices	42
5.	Singular Value Decomposition	44
5.1	Singular value decomposition (SVD)	44
5.2	Singular values and singular vectors	46
5.3	Real SVD	46
5.4	SVD analysis	47
5.5	Useful relations - I	47
5.6	Computation of SVD for small matrices	48
5.7	Reduced SVD	48
5.8	Rank-one expansion	49
5.9	Useful relations - II	49
5.10	Low-rank approximation	50
5.11	Distance to the nearest singular matrix	51
5.12	Small perturbations of matrices	51
5.13	Rank with tolerance ε	51
5.14	Computation of the numerical rank	51
5.15	Metric for matrices	52
5.16	Topological properties of full rank matrices	52

5.17	Topological property of diagonalizable matrices	52
6.	Schur Decomposition	54
6.1	Schur decomposition	54
6.2	Normal matrices	55
6.3	Normal matrices under unitary equivalence	55
6.4	Lemma	55
6.5	Theorem	56
6.6	Remark	56
6.7	Real Schur Decomposition	57
7.	LU Decomposition	59
7.1	Gaussian elimination	59
7.2	Principal minors	61
7.3	Criterion of failure	61
7.4	Gauss matrices	61
7.5	Main matrix formulas	62
7.6	Unit lower/upper triangular matrices	62
7.7	Main matrix formulas (continued)	63
7.8	Theorem (LU decomposition)	63
7.9	Forward and backward substitutions	64
7.10	Cost of computation	65
7.11	Computation of A^{-1}	65
7.12	“Manual” solution of systems of linear equations	66
7.13	Example	66
7.14	Warnings	66
7.15	Partial pivoting	67
7.16	Example	67
7.17	Complete pivoting	67
7.18	Example	67
7.19	Diagonally dominant matrices	68
8.	Cholesky Factorization	69
8.1	Theorem (LDM* Decomposition)	69
8.2	Corollary (LDL* Decomposition)	69
8.3	Sylvester’s Theorem	70
8.4	Corollary	70
8.5	Theorem (Cholesky Factorization)	71
8.6	Algorithm for Cholesky factorization	71
8.7	Cost of computation	72
8.8	Criterion of positive definiteness	72
9.	QR Decomposition	73
9.1	Gram-Schmidt orthogonalization (revisited)	73
9.2	Linearly dependent case	74
9.3	Extension of \hat{Q} to Q	74

9.4	Theorem (QR decomposition)	75
9.5	Real QR decomposition	75
9.6	Positivity of the diagonal of R	75
9.7	Uniqueness of the QR decomposition	76
9.8	Cost of QR	76
9.9	Cost of SVD	76
9.10	Modified Gram-Schmidt orthogonalization	77
10.	Least Squares	80
10.1	Definition	80
10.2	Conditions for existence and uniqueness of a solution	80
10.3	Least squares solution	80
10.4	Normal equations	80
10.5	Theorem	81
10.6	Linear least squares fit	82
10.7	Polynomial least squares fit	82
10.8	Continuous least squares fit	83
10.9	Algorithm 1, based on normal equations	83
10.10	Algorithm 2, based on QR decomposition	84
10.11	Algorithm 3, based on SVD	84
10.12	Rank deficient matrix A	85
11.	Machine Arithmetic	86
11.1	Decimal number system	86
11.2	Floating point representation	86
11.3	Normalized floating point representation	87
11.4	Binary number system	87
11.5	Other number systems	88
11.6	Machine number systems (an abstract version)	88
11.7	Basic properties of machine systems	89
11.8	Two standard machine systems	89
11.9	Rounding rules	89
11.10	Relative errors	90
11.11	Machine epsilon	90
11.12	Machine epsilon for the two standard machine systems	90
11.13	Example	91
11.14	Example	93
11.15	Computational errors	94
11.16	Multiplication and division	94
11.17	Addition and subtraction	95
12.	Condition Numbers	96
12.1	Introduction	96
12.2	Computational process as a function	96
12.3	Condition number of a function	97
12.4	Lemma	97

12.5	Condition number of a matrix	97
12.6	Main theorem for the condition number of a matrix	98
12.7	Corollary	99
12.8	Corollary	99
12.9	Practical interpretation	100
12.10	Maxmag and minmag	100
12.11	Properties of the condition numbers	101
12.12	Closeness to a singular matrix	102
12.13	A posteriori error analysis using the residual	102
12.14	Extension to rectangular matrices	102
13.	Numerical Stability	104
13.1	Introduction	104
13.2	Stable algorithms (definition)	105
13.3	Backward error analysis	105
13.4	Backward stable algorithms (definition)	106
13.5	Theorem	106
13.6	Theorem (without proof)	106
13.7	Example	107
13.8	Further facts (without proofs)	107
13.9	Example	108
13.10	Example	108
14.	Numerically Stable Least Squares	109
14.1	Normal equations revisited	109
14.2	QR-based algorithm revisited	111
14.3	SVD-based algorithm revisited	111
14.4	Hyperplanes	112
14.5	Reflections	112
14.6	Householder reflection matrices	112
14.7	Basic properties of Householder reflectors	113
14.8	Theorem	113
14.9	Remarks	113
14.10	Corollary	114
14.11	QR Decomposition via Householder reflectors	114
14.12	Givens rotation matrices	115
14.13	Geometric description of Givens rotators	115
14.14	QR decomposition via Givens rotators	116
14.15	Cost of QR via Givens rotators	117
15.	Computation of Eigenvalues: Theory	119
15.1	Preface	119
15.2	Rayleigh quotient	120
15.3	Restriction to the unit sphere	120
15.4	Properties of Rayleigh quotient	121
15.5	Theorem	121

15.6	Lemma	122
15.7	Courant-Fisher Minimax Theorem	123
15.8	Corollary	123
15.9	Theorem	124
15.10	Corollary	124
15.11	Approximation analysis using the residual	125
15.12	Bauer-Fike theorem	125
15.13	Corollary	126
15.14	Left eigenvectors (definition)	126
15.15	Lemma	126
15.16	Lemma	127
15.17	Lemma	127
15.18	Lemma	128
15.19	Theorem	128
15.20	Condition number of a simple eigenvalue	130
15.21	Remarks	130
15.22	Properties of condition numbers for simple eigenvalues	130
15.23	Relation to Schur decomposition	131
15.24	Multiple and ill-conditioned eigenvalues	131
15.25	Theorem (without proof)	131
15.26	First Gershgorin theorem	132
15.27	Second Gershgorin theorem	133
16.	Computation of Eigenvalues: Power Method	135
16.1	Preface	135
16.2	Power method: a general scheme	136
16.3	Linear, quadratic and cubic convergence	137
16.4	Remarks on convergence	137
16.5	Scaling problem in the power method	138
16.6	First choice for the scaling factor	138
16.7	Example	138
16.8	Example	139
16.9	Theorem (convergence of the power method)	140
16.10	Second choice for the scaling factor	141
16.11	Example	142
16.12	Example	143
16.13	Initial vector	144
16.14	Aiming at the smallest eigenvalue	144
16.15	Inverse power method	145
16.16	Practical implementation	145
16.17	Aiming at any eigenvalue	146
16.18	Power method with shift	147
16.19	Note on the speed of convergence	147
16.20	Power method with Rayleigh quotient shift	148
16.21	Example	148

16.22 Power method: pros and cons	149
16. Computation of Eigenvalues: QR Algorithm	151
17.1 “Pure” QR algorithm	151
17.2 Two basic facts	151
17.3 Theorem (convergence of the QR algorithm)	152
17.4 Example	153
17.5 Advantages of the QR algorithm	153
17.6 Hessenberg matrix	154
17.7 Lucky reduction of the eigenvalue problem	154
17.8 Theorem (Hessenberg decomposition)	155
17.9 Cost of Arnoldi algorithm	156
17.10 Preservation of Hessenberg structure	156
17.11 Cost of a QR step for Hessenberg matrices	157
17.12 The case of Hermitian matrices	157
17.13 Theorem (without proof)	157
17.14 QR algorithm with shift	158
17.15 QR algorithm with Rayleigh quotient shift	158
17.16 QR algorithm with deflation	159
17.17 Example	160
17.18 Wilkinson shift	160
17.19 Example 17.17 continued	160
17.20 Wilkinson shifts for complex eigenvalues of real matrices	161
17.21 Analysis of the double-step	162

Review of Linear Algebra

0.1 Matrices and vectors

The set of $m \times n$ matrices (m rows, n columns) with entries in a field \mathbb{F} is denoted by $\mathbb{F}^{m \times n}$. We will only consider two fields: complex ($\mathbb{F} = \mathbb{C}$) and real ($\mathbb{F} = \mathbb{R}$). For any matrix $A \in \mathbb{F}^{m \times n}$, we denote its entries by

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

The vector space \mathbb{F}^n consists of column vectors with n components:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{F}^n.$$

Note: all vectors, by default, are *column* vectors (not row vectors).

0.2 Product of a matrix and a vector

The product $y = Ax$ is a vector in \mathbb{F}^m :

$$y = Ax = \left[\begin{array}{c|c|c|c} a_1 & a_2 & \cdots & a_n \end{array} \right] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} + x_2 \begin{bmatrix} a_2 \\ \vdots \\ a_n \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_n \\ \vdots \\ a_n \end{bmatrix}$$

where a_1, \dots, a_n denote the columns of the matrix A .

Note: Ax is a linear combination of the columns of A . Thus

$$Ax \in \text{span}\{a_1, \dots, a_n\}.$$

0.3 Matrix as a linear transformation

Every matrix $A \in \mathbb{F}^{m \times n}$ defines a linear transformation

$$\mathbb{F}^n \rightarrow \mathbb{F}^m \quad \text{by} \quad x \mapsto Ax.$$

We will denote that transformation by the same letter, A .

0.4 Range and rank of a matrix

The *range* of $A \in \mathbb{F}^{m \times n}$ is a vector subspace of \mathbb{F}^m defined by

$$\text{Range } A = \{Ax : x \in \mathbb{F}^n\} \subset \mathbb{F}^m.$$

The range of A is a subspace of \mathbb{F}^m spanned by the columns of A :

$$\text{Range } A = \text{span}\{a_1, \dots, a_n\}.$$

The *rank* of A is defined by

$$\text{rank } A = \dim(\text{Range } A).$$

0.5 Kernel (nullspace) of a matrix

The *kernel* (also called *nullspace*) of $A \in \mathbb{F}^{m \times n}$ is a vector subspace of \mathbb{F}^n :

$$\text{Ker } A = \{x : Ax = 0\} \subset \mathbb{F}^n.$$

We have the following **matrix rank formula**:

$$\text{rank } A = \dim(\text{Range } A) = n - \dim(\text{Ker } A). \quad (0.1)$$

An intuitive explanation: n is the total number of dimensions in the space \mathbb{F}^n to which the matrix A is applied. The kernel collapses to zero, thus $\dim(\text{Ker } A)$ dimensions disappear. The rest survive, are carried over to \mathbb{F}^m , and make the range of A .

0.6 Surjective/injective/bijective transformations

The transformation from \mathbb{F}^n to \mathbb{F}^m defined by a matrix A is

- *surjective* iff $\text{Range } A = \mathbb{F}^m$. This is only possible if $n \geq m$.
- *injective* iff $\text{Ker } A = \{0\}$. This is only possible if $n \leq m$.
- *bijective* iff it is surjective and injective.

In the last case, $m = n$ and we call A an *isomorphism*.

0.7 Square matrices and inverse of a matrix

Every square matrix $A \in \mathbb{F}^{n \times n}$ defines a linear transformation $\mathbb{F}^n \rightarrow \mathbb{F}^n$, called an *operator* on \mathbb{F}^n . The *inverse* of a square matrix $A \in \mathbb{F}^{n \times n}$ is a square matrix $A^{-1} \in \mathbb{F}^{n \times n}$ uniquely defined by

$$A^{-1}A = AA^{-1} = I \quad (\text{identity matrix}).$$

A matrix $A \in \mathbb{F}^{n \times n}$ is said to be *invertible* (*nonsingular*) iff A^{-1} exists. The matrix is *noninvertible* (*singular*) iff A^{-1} does not exist.

The following are equivalent:

- (a) A is invertible
- (b) $\text{rank } A = n$
- (c) $\text{Range } A = \mathbb{F}^n$
- (d) $\text{Ker } A = \{0\}$
- (e) 0 is not an eigenvalue of A
- (f) $\det A \neq 0$

Note that

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

0.8 Upper and lower triangular matrices

A square matrix $A \in \mathbb{F}^{n \times n}$ is *upper triangular* if $a_{ij} = 0$ for all $i > j$.

A square matrix $A \in \mathbb{F}^{n \times n}$ is *lower triangular* if $a_{ij} = 0$ for all $i < j$.

A square matrix D is *diagonal* if $d_{ij} = 0$ for all $i \neq j$. In that case we write $D = \text{diag}\{d_{11}, \dots, d_{nn}\}$.

Note: if A and B are upper (lower) triangular, then so are their product AB and inverses A^{-1} and B^{-1} (if the inverses exist).

upper
triangular
matrices:

lower
triangular
matrices:

0.9 Eigenvalues and eigenvectors

We say that $\lambda \in \mathbb{F}$ is an *eigenvalue* for a square matrix $A \in \mathbb{F}^{n \times n}$ with an *eigenvector* $x \in \mathbb{F}^n$ if

$$Ax = \lambda x \quad \text{and} \quad x \neq 0.$$

Eigenvalues are the roots of the characteristic polynomial

$$C_A(\lambda) = \det(\lambda I - A) = 0.$$

By the fundamental theorem of algebra, every polynomial of degree n with complex coefficients has exactly n complex roots $\lambda_1, \dots, \lambda_n$ (counting multiplicities). This implies

$$C_A(\lambda) = \prod_{i=1}^n (\lambda - \lambda_i).$$

0.10 Eigenvalues of real matrices

A real-valued square matrix $A \in \mathbb{R}^{n \times n}$ can be treated in *two ways*:

First, we can treat it as a complex matrix $A \in \mathbb{C}^{n \times n}$ (we can do that because real numbers make a subset of the complex plane \mathbb{C}). Then A has n eigenvalues (real or non-real complex). An important note: since the characteristic polynomial $C_A(\lambda)$ has real coefficients, its non-real complex roots (i.e., non-real complex eigenvalues of A) come in conjugate pairs. Thus the eigenvalues of A can be listed as follows:

$$c_1, \dots, c_p, \quad a_1 \pm \mathbf{i}b_1, \dots, a_q \pm \mathbf{i}b_q$$

where c_i ($i = 1, \dots, p$) are all *real* eigenvalues and $a_j \pm \mathbf{i}b_j$ ($j = 1, \dots, q$) are all non-real complex eigenvalues (here a_j and $b_j \neq 0$ are real numbers); and we denote $\mathbf{i} = \sqrt{-1}$. Note that $p + 2q = n$.

Second, we can treat A as a matrix over the real field \mathbb{R} , then we can only use real numbers. In that case the eigenvalues of A are only

$$c_1, \dots, c_p$$

(from the previous list). Thus A may have fewer than n eigenvalues, and it may have none at all. For instance, $A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ has no (real) eigenvalues. However, if A does have real eigenvalues c_1, \dots, c_p , then for each eigenvalue c_i ($i = 1, \dots, p$) there is a real eigenvector, i.e., $Ax = c_i x$ for some $x \in \mathbb{R}^n$.

0.11 Diagonalizable matrices

A square matrix $A \in \mathbb{F}^{n \times n}$ is *diagonalizable* (over \mathbb{F}) iff

$$A = X\Lambda X^{-1}, \quad (0.2)$$

where $\Lambda = \text{diag}\{\lambda_1, \dots, \lambda_n\}$ is a diagonal matrix and $X \in \mathbb{F}^{n \times n}$. In this case $\lambda_1, \dots, \lambda_n$ are the eigenvalues of A and the columns x_1, \dots, x_n of the matrix X are the corresponding eigenvectors.

Indeed, we can rewrite (0.2) in the form $AX = X\Lambda$ and note that

$$AX = \begin{bmatrix} & & & & \\ & A & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Ax_1 & Ax_2 & \cdots & Ax_n \\ | & | & & | \end{bmatrix}$$

and

$$X\Lambda = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & & 0 \\ & \ddots & & \\ & & & \lambda_n \\ 0 & & & \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \lambda_1 x_1 & \lambda_2 x_2 & \cdots & \lambda_n x_n \\ | & | & & | \end{bmatrix},$$

therefore

$$Ax_1 = \lambda_1 x_1, \quad Ax_2 = \lambda_2 x_2, \quad \dots \quad Ax_n = \lambda_n x_n.$$

The formula (0.2) is often called the *eigenvalue decomposition* of A .

0.12 Trace

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

$$\text{tr } A = \sum_{i=1}^n a_{ii}.$$

Trace has the following properties:

- (a) $\text{tr } AB = \text{tr } BA$;
- (b) if $A = X^{-1}BX$, then $\text{tr } A = \text{tr } B$;
- (c) $\text{tr } A = \lambda_1 + \dots + \lambda_n$ (the sum of all complex eigenvalues).

0.13 Transpose of a matrix

For any matrix $A = (a_{ij}) \in \mathbb{F}^{m \times n}$ we denote by $A^T = (a_{ji}) \in \mathbb{F}^{n \times m}$ the *transpose*¹ of A . Note that

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

If A is a square matrix, then

$$\det A^T = \det A, \quad (A^T)^{-1} = (A^{-1})^T.$$

A matrix A is symmetric if $A^T = A$. Only square matrices can be symmetric.

0.14 Conjugate transpose of a matrix

For any matrix $A = (a_{ij}) \in \mathbb{C}^{m \times n}$ we denote by $A^* = (\bar{a}_{ji}) \in \mathbb{F}^{n \times m}$ the *conjugate transpose*² of A (also called *adjoint* of A):

$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \quad \Longrightarrow \quad A^* = \begin{bmatrix} \bar{a} & \bar{c} & \bar{e} \\ \bar{b} & \bar{d} & \bar{f} \end{bmatrix}$$

Note that

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

If A is a square matrix, then

$$\det A^* = \overline{\det A}, \quad (A^*)^{-1} = (A^{-1})^*. \quad (0.3)$$

For any real matrix $A \in \mathbb{R}^{m \times n}$, we have $A^* = A^T$. However, if $A \in \mathbb{C}^{m \times n}$ is a non-real complex matrix, then its conjugate transpose A^* is different from its transpose A^T .

0.15 Convention on the use of conjugate transpose

For non-real complex matrices A (and vectors x), just transpose A^T (respectively, x^T) usually does not give us anything good. For complex matrices and vectors, we should always use *conjugate* transpose.

In our formulas we will use A^* for both complex and real matrices, and we will use x^* for both complex and real vectors. We should just keep in mind that for real matrices A^* can be replaced with A^T and for real vectors x^* can be replaced with x^T .

¹Another popular notation for the transpose is A^t , but we will always use A^T .

²Another popular notation for the conjugate transpose is A^H , but we will use A^* .

Norms and Inner Products

1.1 Norms

A *norm* on a vector space V over \mathbb{C} is a real valued function $\|\cdot\|$ on V satisfying three axioms:

1. $\|v\| \geq 0$ for all $v \in V$ and $\|v\| = 0$ if and only if $v = 0$.
2. $\|cv\| = |c| \|v\|$ for all $c \in \mathbb{C}$ and $v \in V$.
3. $\|u + v\| \leq \|u\| + \|v\|$ for all $u, v \in V$ (**triangle inequality**).

If V is a vector space over \mathbb{R} , then a norm is a function $V \rightarrow \mathbb{R}$ satisfying the same axioms, except $c \in \mathbb{R}$.

Norm is a general version of *length* of a vector.

1.2 1-norm, 2-norm, and ∞ -norm

Some common norms on \mathbb{C}^n and \mathbb{R}^n are:

$$\|x\|_1 = \sum_{i=1}^n |x_i| \quad (1\text{-norm})$$

$$\|x\|_2 = \left(\sum_{i=1}^n |x_i|^2 \right)^{1/2} \quad (2\text{-norm})$$

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} \quad (p\text{-norm, } p \geq 1)$$

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i| \quad (\infty\text{-norm})$$

The 2-norm in \mathbb{R}^n corresponds to the Euclidean distance.

Some norms on $C[a, b]$, the space of continuous functions on $[a, b]$:

$$\|f\|_1 = \int_a^b |f(x)| dx \quad (1\text{-norm})$$

$$\|f\|_2 = \left(\int_a^b |f(x)|^2 dx \right)^{1/2} \quad (2\text{-norm})$$

$$\|f\|_p = \left(\int_a^b |f(x)|^p dx \right)^{1/p} \quad (p\text{-norm, } p \geq 1)$$

$$\|f\|_\infty = \max_{a \leq x \leq b} |f(x)| \quad (\infty\text{-norm})$$

1.3 Unit vectors, normalization

We say that $u \in V$ is a *unit vector* if $\|u\| = 1$.

For any $v \neq 0$, the vector $u = v/\|v\|$ is a unit vector.

We say that we *normalize* a vector $v \neq 0$ when we transform it to $u = v/\|v\|$.

1.4 Unit sphere, unit ball

The *unit sphere* in V is the set of all unit vectors:

$$\mathbb{S}_1 = \{v \in V : \|v\| = 1\} \quad (1.1)$$

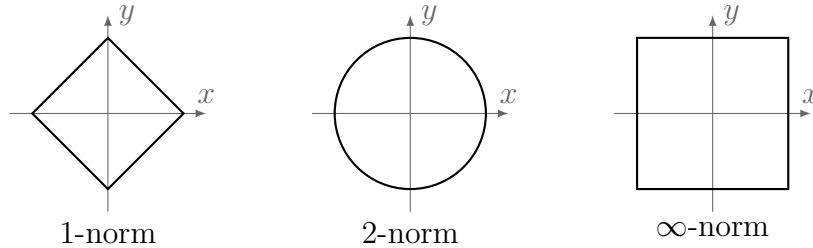
The *unit ball* in V is the set of all vectors whose norm is ≤ 1 :

$$\mathbb{B}_1 = \{v \in V : \|v\| \leq 1\} \quad (1.2)$$

The unit ball is the unit sphere with its interior.

1.5 Images of unit spheres

The figure shows unit spheres in \mathbb{R}^2 for three common norms: 1-norm, 2-norm (the Euclidean norm), and ∞ -norm.



1.6 Frobenius norm

The space $\mathbb{C}^{m \times n}$ of matrices can be naturally identified with \mathbb{C}^{mn} . Hence we can define norms on it in a similar way. In particular, an analogue of the 2-norm is

$$\|A\|_F = \left(\sum_i \sum_j |a_{ij}|^2 \right)^{1/2}.$$

It is called *Frobenius norm* of a matrix. It is easy to verify that

$$\|A\|_F^2 = \text{tr}(A^*A) = \text{tr}(AA^*).$$

1.7 Induced matrix norms

Recall that every matrix $A \in \mathbb{F}^{m \times n}$ defines a linear transformation $\mathbb{F}^n \rightarrow \mathbb{F}^m$. Suppose the spaces \mathbb{F}^n and \mathbb{F}^m are equipped with certain norms, $\|\cdot\|$. Then we define the *induced norm* (also called *operator norm*) of A as follows:

$$\|A\| = \sup_{\|x\|=1} \|Ax\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} \quad (1.3)$$

This norm is the maximum expansion factor, i.e., the largest factor by which the matrix A stretches vectors $x \in \mathbb{F}^n$.

Respectively, we obtain $\|A\|_1$, $\|A\|_2$, and $\|A\|_\infty$, if the spaces \mathbb{F}^n and \mathbb{F}^m are equipped with $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$.

For students familiar with topology: The supremum in (1.3) is always attained and can be replaced by maximum. This follows from the compactness of \mathbb{S}_1 and the continuity of $\|\cdot\|$. For the 2-norm, this can be also proved by an algebraic argument, see Corollary 4.23.

There are norms on $\mathbb{C}^{n \times n}$ that are not induced by any norm on \mathbb{C}^n , for example $\|A\| = \max_{i,j} |a_{ij}|$ (see Exercise 1.1).

1.8 Computation of $\|A\|_1$, $\|A\|_2$, and $\|A\|_\infty$

We have simple rules for computing $\|A\|_1$ and $\|A\|_\infty$:

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}| \quad (\text{“maximum column sum”})$$
$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}| \quad (\text{“maximum row sum”})$$

Unfortunately, there is no explicit formulas for $\|A\|_2$ in terms of the a_{ij} 's. A substantial part of this course will be devoted to the computation of $\|A\|_2$.

1.9 Inequalities for induced matrix norms

Any induced matrix norm satisfies

$$\|Ax\| \leq \|A\| \|x\| \quad \text{and} \quad \|AB\| \leq \|A\| \|B\|.$$

1.10 Inner products

Let V be a vector space over \mathbb{C} . An *inner product* on V is a complex valued function of two vector arguments, i.e., $V \times V \rightarrow \mathbb{C}$, denoted by $\langle \cdot, \cdot \rangle$, satisfying four axioms:

1. $\langle u, v \rangle = \overline{\langle v, u \rangle}$ for all $u, v \in V$.
2. $\langle cu, v \rangle = c\langle u, v \rangle$ for all $c \in \mathbb{C}$ and $u, v \in V$.
3. $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$ for all $u, v, w \in V$.
4. $\langle u, u \rangle \geq 0$ for all $u \in V$, and $\langle u, u \rangle = 0$ iff $u = 0$.

These axioms imply the following rules:

$$\langle u, v + w \rangle = \langle u, v \rangle + \langle u, w \rangle \quad \text{and} \quad \langle u, cv \rangle = \bar{c}\langle u, v \rangle. \quad (1.4)$$

i.e., the inner product is linear in the first argument (due to axioms 2 and 3) but “conjugate linear” in the second.

If V is a vector space over \mathbb{R} , then an inner product is a real valued function of two arguments, i.e., $V \times V \rightarrow \mathbb{R}$ satisfying the same axioms (except $c \in \mathbb{R}$, and there is no need to take conjugate).

1.11 Standard inner product

A standard (canonical) inner product in \mathbb{C}^n is

$$\langle x, y \rangle = \sum_{i=1}^n x_i \bar{y}_i = y^* x.$$

A standard (canonical) inner product in \mathbb{R}^n is

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

A standard inner product in $C([a, b])$, the space of complex-valued continuous functions, is:

$$\langle f, g \rangle = \int_a^b f(x) \bar{g}(x) dx$$

For real-valued functions, we can replace $\bar{g}(x)$ with $g(x)$.

1.12 Cauchy-Schwarz inequality

Let V be an inner product space. Then for any two vectors $u, v \in V$

$$|\langle u, v \rangle| \leq \langle u, u \rangle^{1/2} \langle v, v \rangle^{1/2} \quad (1.5)$$

The equality holds if and only if u and v are linearly dependent, i.e.,

$$|\langle u, v \rangle| = \langle u, u \rangle^{1/2} \langle v, v \rangle^{1/2} \iff au + bv = 0 \quad (1.6)$$

for some scalars a, b of which at least one is different from zero.

Proof. The easy case is where $v = 0$. Then $\langle u, v \rangle = 0$ and $\langle v, v \rangle = 0$, so both sides of (1.5) are zero, hence the claim is trivial.

We now turn to the difficult case: $v \neq 0$. Consider the function

$$\begin{aligned} f(z) &= \langle u - zv, u - zv \rangle \\ &= \langle u, u \rangle - z\langle v, u \rangle - \bar{z}\langle u, v \rangle + |z|^2\langle v, v \rangle \end{aligned}$$

of a complex variable z . Let $z = re^{i\theta}$ and $\langle u, v \rangle = se^{i\varphi}$ be the polar forms of the complex numbers z and $\langle u, v \rangle$, respectively. We fix $\theta = \varphi$ and note that

$$z\langle v, u \rangle = re^{i\varphi}se^{-i\varphi} = rs$$

and

$$\bar{z}\langle u, v \rangle = re^{-i\varphi}se^{i\varphi} = rs.$$

We assume that r varies from $-\infty$ to ∞ , then

$$0 \leq f(z) = \langle u, u \rangle - 2sr + r^2\langle v, v \rangle \quad (1.7)$$

Since this holds for all $r \in \mathbb{R}$, the discriminant of the above quadratic polynomial has to be ≤ 0 , i.e.

$$s^2 - \langle u, u \rangle \langle v, v \rangle \leq 0. \quad (1.8)$$

This proves the inequality (1.5).

The equality in (1.5) means the equality in (1.8), because $s = |\langle u, v \rangle|$. The equality in (1.8) means that the discriminant of the quadratic polynomial in (1.7) is zero, hence that polynomial assumes a zero value, i.e.,

$$f(z) = \langle u - zv, u - zv \rangle = 0$$

for some $z \in \mathbb{C}$. This, in turn, means $u - zv = 0$, thus $u = zv$ for some $z \in \mathbb{C}$, i.e., u and v are linearly dependent, as claimed. \square

1.13 Induced norms

If V is an inner product vector space, then

$$\|v\| = \langle v, v \rangle^{1/2} \quad (1.9)$$

defines a norm on V (all the axioms can be verified by direct inspection, except the triangle inequality is proved by Cauchy-Schwarz inequality 1.12).

1.14 Polarization identity

Some norms are induced by inner products, others are not. The 2-norm in \mathbb{R}^n and in \mathbb{C}^n (defined in Sect. 1.2) is induced by the standard inner product in those spaces (as defined in Sect. 1.11).

In vector spaces over \mathbb{R} , if a norm $\|\cdot\|$ is induced by an inner product $\langle \cdot, \cdot \rangle$, then the latter can be computed by *polarization identity*:

$$\langle u, v \rangle = \frac{1}{4} (\|u + v\|^2 - \|u - v\|^2) \quad (1.10)$$

A similar but more complicated polarization identity holds in vector spaces over \mathbb{C} .

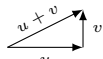
1.15 Orthogonal vectors

Two vectors $u, v \in V$ are said to be *orthogonal* if $\langle u, v \rangle = 0$.

1.16 Pythagorean theorem

If two vectors u and v are orthogonal, then

$$\|u + v\|^2 = \|u\|^2 + \|v\|^2 \quad (1.11)$$

Vectors u and v make two legs of a right triangle, and $u + v$ is its hypotenuse: 

Suppose u_1, \dots, u_k are mutually orthogonal, i.e., $\langle u_i, u_j \rangle = 0$ for each $i \neq j$. Then one can show, by induction, that $\|u_1 + \dots + u_k\|^2 = \|u_1\|^2 + \dots + \|u_k\|^2$.

1.17 Orthogonality and linear independence

If nonzero vectors u_1, \dots, u_k are mutually orthogonal, i.e., $\langle u_i, u_j \rangle = 0$ for each $i \neq j$, then they are linearly independent.

Proof. Suppose a linear combination of these vectors is zero, i.e.,

$$c_1 u_1 + \dots + c_k u_k = 0.$$

Then for every $i = 1, \dots, k$ we have

$$0 = \langle 0, u_i \rangle = \langle c_1 u_1 + \dots + c_k u_k, u_i \rangle = c_i \langle u_i, u_i \rangle$$

hence $c_i = 0$ (because $\langle u_i, u_i \rangle \neq 0$ for the non-zero vector u_i). Thus $c_1 = \dots = c_k = 0$. \square

1.18 Orthonormal sets of vectors

A set $\{u_1, \dots, u_k\}$ of vectors is said to be *orthonormal* if all the vectors u_i are mutually orthogonal and have unit length (i.e., $\|u_i\| = 1$ for all i).

The orthonormality can be expressed by a single formula:

$$\langle u_i, u_j \rangle = \delta_{ij},$$

where δ_{ij} denotes the *Kronecker delta symbol* defined as follows: $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$.

1.19 Orthonormal basis (ONB)

An orthonormal set of vectors $\{u_1, \dots, u_k\}$ that is a basis in V is called *orthonormal basis* (ONB).

An orthonormal set $\{u_1, \dots, u_k\}$ is an ONB in V iff $k = n = \dim V$.

1.20 Fourier expansion

If $\{u_1, \dots, u_n\}$ is an ONB, then for any vector v we have

$$v = \sum_{i=1}^n \langle v, u_i \rangle u_i. \quad (1.12)$$

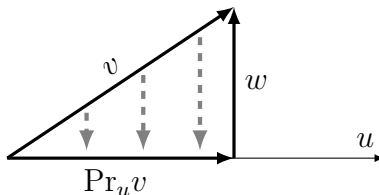
In other words, the numbers $\langle v, u_i \rangle$ (called *Fourier coefficients*) are the coordinates of the vector v in the basis $\{u_1, \dots, u_n\}$.

1.21 Orthogonal projection

Let $u, v \in V$, and $u \neq 0$. The *orthogonal projection* of v onto u is

$$\text{Pr}_u v = \frac{\langle v, u \rangle}{\|u\|^2} u.$$

One can easily check that the vector $w = v - \text{Pr}_u v$ is orthogonal to u . Thus v is the sum of two vectors: $\text{Pr}_u v$ (parallel to u) and w (orthogonal to u).



1.22 Angle between vectors

If V is a vector space over \mathbb{R} , then for any nonzero vectors $u, v \in V$ there is a unique angle $\theta \in [0, \pi]$ such that

$$\cos \theta = \frac{\langle v, u \rangle}{\|u\| \|v\|}. \quad (1.13)$$

Indeed, by Cauchy-Schwarz inequality 1.12, the above fraction is a real number in the interval $[-1, 1]$. Then recall that the function \cos^{-1} takes $[-1, 1]$ onto $[0, \pi]$ bijectively.

We call θ the *angle* between u and v .

Note that $\cos \theta = 0$ (i.e., $\theta = \pi/2$) if and only if u and v are orthogonal. Also, $\cos \theta = \pm 1$ if and only if u, v are proportional, i.e. $v = cu$. In that case the sign of c coincides with the sign of $\cos \theta$.

1.23 Orthogonal projection onto a subspace

Let $\{u_1, \dots, u_k\}$ be an orthonormal set of vectors. According to Sect. 1.17, they are linearly independent, hence they span a k -dimensional subspace $L = \text{span}\{u_1, \dots, u_k\}$. For any $v \in V$,

$$\text{Pr}_L v = \sum_{i=1}^k \langle v, u_i \rangle u_i$$

is the orthogonal projection of v onto L . One can easily check that the vector

$$w = v - \text{Pr}_L v$$

is orthogonal to all the vectors u_1, \dots, u_k , hence it is orthogonal to all vectors in L . In particular, the vectors u_1, \dots, u_k, w are mutually orthogonal, and one can easily check that

$$\text{span}\{u_1, \dots, u_k, v\} = \text{span}\{u_1, \dots, u_k, w\}.$$

1.24 Degenerate case

In the above construction, we have

$$w = 0 \quad \iff \quad v \in \text{span}\{u_1, \dots, u_k\}.$$

Note: In Section 1.23, the subspace L is spanned by an *orthonormal* set of vectors. What if we want to project a vector v onto a subspace L that is spanned by a non-orthonormal set of vectors $\{v_1, \dots, v_k\}$?

1.25 Gram-Schmidt orthogonalization

Let $\{v_1, \dots, v_k\}$ be a linearly independent set of vectors. They span a k -dimensional subspace $L = \text{span}\{v_1, \dots, v_k\}$. Our goal is to find an orthonormal set of vectors $\{u_1, \dots, u_k\}$ that spans L . We will modify and adjust the vectors v_1, \dots, v_k , one by one.

At our first step, we define

$$w_1 = v_1 \quad \text{and} \quad u_1 = w_1/\|w_1\|. \quad (1.14)$$

At our second step, we define

$$w_2 = v_2 - \langle v_2, u_1 \rangle u_1 \quad \text{and} \quad u_2 = w_2/\|w_2\|. \quad (1.15)$$

Then recursively, for each $p \geq 2$, we define

$$w_p = v_p - \sum_{i=1}^{p-1} \langle v_p, u_i \rangle u_i, \quad \text{and} \quad u_p = w_p/\|w_p\| \quad (1.16)$$

By induction on p , one can easily check that

$$\text{span}\{v_1, \dots, v_p\} = \text{span}\{u_1, \dots, u_p\}$$

and $w_p \neq 0$ (in accordance with Sect. 1.24).

The above procedure is called *Gram-Schmidt orthogonalization*. It gives us an orthonormal set $\{u_1, \dots, u_k\}$ which spans the same subspace:

$$L = \text{span}\{v_1, \dots, v_k\} = \text{span}\{u_1, \dots, u_k\}.$$

1.26 Construction of ONB

Suppose $\{v_1, \dots, v_n\}$ is a basis in V . Then the Gram-Schmidt orthogonalization gives us an orthonormal basis (ONB) $\{u_1, \dots, u_n\}$ in V .

As a result, every finite dimensional vector space with an inner product has an ONB. Furthermore, every set of orthonormal vectors $\{u_1, \dots, u_k\}$ can be extended to an ONB.

1.27 Legendre polynomials (optional)

Let $V = \mathbb{P}_n(\mathbb{R})$, the space of real polynomials of degree $\leq n$, with the inner product given by $\langle f, g \rangle = \int_0^1 f(x)g(x) dx$. Applying Gram-Schmidt orthogonalization to the basis $\{1, x, \dots, x^n\}$ gives the first $n + 1$ of the so called *Legendre polynomials*.

1.28 Orthogonal complement

Let $S \subset V$ be a subset (not necessarily a subspace). Then

$$S^\perp = \{v \in V : \langle v, w \rangle = 0 \text{ for all } w \in S\}$$

is called the *orthogonal complement* to S .

One can easily check that S^\perp is a vector subspace of V .

We also note that if $W \subset V$ is a subspace of V , then $(W^\perp)^\perp \subset W$. Moreover, if V is finite dimensional, then $(W^\perp)^\perp = W$ (see Exercise 1.4).

1.29 Orthogonal direct sum

If $W \subset V$ is a finite dimensional subspace of V , then $V = W \oplus W^\perp$.

Proof. Let $\{u_1, \dots, u_n\}$ be an ONB of W (one exists according to Sect. 1.26). For any $v \in V$ define

$$w = \sum_{i=1}^n \langle v, u_i \rangle u_i$$

One can easily verify that $w \in W$ and $v - w \in W^\perp$. Therefore $v = w + w'$ where $w \in W$ and $w' \in W^\perp$. Lastly, $W \cap W^\perp = \{0\}$ because any vector $w \in W \cap W^\perp$ is orthogonal to itself, i.e., $\langle w, w \rangle = 0$, hence $w = 0$.

1.30 Some useful formulas

Let $\{u_1, \dots, u_n\}$ be an orthonormal set (not necessarily an ONB) in V . Then for any vector $v \in V$ we have *Bessel's inequality*:

$$\|v\|^2 \geq \sum_{i=1}^n |\langle v, u_i \rangle|^2. \quad (1.17)$$

If $\{u_1, \dots, u_n\}$ is an ONB in V , then Bessel's inequality turns into an equality:

$$\|v\|^2 = \sum_{i=1}^n |\langle v, u_i \rangle|^2. \quad (1.18)$$

More generally, we have *Parseval's identity*:

$$\langle v, w \rangle = \sum_{i=1}^n \langle v, u_i \rangle \overline{\langle w, u_i \rangle}, \quad (1.19)$$

which easily follows from the Fourier expansion (1.12).

Parseval's identity (1.19) can be also written as follows. Suppose

$$v = \sum_{i=1}^n a_i u_i \quad \text{and} \quad w = \sum_{i=1}^n b_i u_i,$$

so that (a_1, \dots, a_n) and (b_1, \dots, b_n) are the coordinates of the vectors v and w , respectively, in the ONB $\{u_1, \dots, u_n\}$. Then

$$\langle v, w \rangle = \sum_{i=1}^n a_i \bar{b}_i. \quad (1.20)$$

In particular,

$$\|v\|^2 = \langle v, v \rangle = \sum_{i=1}^n a_i \bar{a}_i = \sum_{i=1}^n |a_i|^2. \quad (1.21)$$

Exercise 1.1. Show that the norm $\|A\| = \max_{i,j} |a_{ij}|$ on the space of $n \times n$ real matrices is not induced by any vector norm in \mathbb{R}^n . Hint: use inequalities from Section 1.7.

Exercise 1.2. Prove the *Neumann lemma*: if $\|A\| < 1$, then $I - A$ is invertible. Here $\|\cdot\|$ is a norm on the space of $n \times n$ matrices induced by a vector norm.

Exercise 1.3. Let V be an inner product space, and $\|\cdot\|$ denote the norm induced by the inner product. Prove the *parallelogram law*

$$\|u + v\|^2 + \|u - v\|^2 = 2\|u\|^2 + 2\|v\|^2.$$

Based on this, show that the norms $\|\cdot\|_1$ and $\|\cdot\|_\infty$ in \mathbb{C}^2 are not induced by any inner products.

Exercise 1.4. Let $W \subset V$ be a subspace of an inner product space V .

- (i) Prove that $W \subset (W^\perp)^\perp$.
- (ii) If, in addition, V is finite dimensional, prove that $W = (W^\perp)^\perp$.

Exercise 1.5. Let $\{u_1, \dots, u_n\}$ be an ONB in \mathbb{C}^n . Assuming that n is even, compute

$$\|u_1 - u_2 + u_3 - \dots + u_{n-1} - u_n\|$$

Unitary Matrices

2.1 Isometries

Let V and W be two inner product spaces (both real or both complex). An isomorphism $T: V \rightarrow W$ is called an *isometry* if it preserves the inner product, i.e.

$$\langle Tv, Tw \rangle = \langle v, w \rangle$$

for all $v, w \in V$. In this case the spaces V and W are said to be *isometric*.

2.2 Characterization of isometries - I

An isomorphism T is an isometry iff it preserves the induced norm, i.e., $\|Tv\| = \|v\|$ for all vectors $v \in V$.

Proof. This follows from Polarization Identity (1.10)

2.3 Characterization of isometries - II

An isomorphism T is an isometry iff $\|Tu\| = \|u\|$ for all *unit* vectors $u \in V$.

Proof. This easily follows from the previous section and the fact that every non-zero vector v can be normalized by $u = v/\|v\|$; cf. Sect. 1.3.

2.4 Characterization of isometries - III

Let $\dim V < \infty$. A linear transformation $T: V \rightarrow W$ is an isometry iff there exists an ONB $\{u_1, \dots, u_n\}$ in V such that $\{Tu_1, \dots, Tu_n\}$ is an ONB in W .

Proof. If T is an isometry, then for any ONB $\{u_1, \dots, u_n\}$ in V the set $\{Tu_1, \dots, Tu_n\}$ is an ONB in W . Now suppose T takes an ONB $\{u_1, \dots, u_n\}$ in V to an ONB $\{Tu_1, \dots, Tu_n\}$ in W . Since T takes a basis in V into a basis in W , it is an isomorphism. Now for each vector $v = c_1u_1 + \dots + c_nu_n \in V$ we have, by linearity, $Tv = c_1Tu_1 + \dots + c_nTu_n$, hence v and Tv have the same coordinates in the respective ONBs. Now by Parseval's Identity (1.21) we have

$$\|v\|^2 = |c_1|^2 + \dots + |c_n|^2 = \|Tv\|^2$$

Thus T is an isometry according to Sect. 2.2.

2.5 Identification of finite-dimensional inner product spaces

Finite dimensional inner product spaces V and W (over the same field) are *isometric* iff $\dim V = \dim W$.

(This follows from Section 2.4.)

As a result, we can make the following useful identifications:

- Ⓒ All complex n -dimensional inner product spaces can be identified with \mathbb{C}^n equipped with the standard inner product $\langle x, y \rangle = y^*x$.
- Ⓓ All real n -dimensional inner product spaces can be identified with \mathbb{R}^n equipped with the standard inner product $\langle x, y \rangle = y^T x$.

These identifications allow us to focus on the study of the standard spaces \mathbb{C}^n and \mathbb{R}^n equipped with the standard inner product.

Isometries $\mathbb{C}^n \rightarrow \mathbb{C}^n$ and $\mathbb{R}^n \rightarrow \mathbb{R}^n$ are operators that, in a standard basis $\{e_1, \dots, e_n\}$, are given by matrices of a special type, as defined below.

2.6 Unitary and orthogonal matrices

A matrix $Q \in \mathbb{C}^{n \times n}$ is said to be *unitary* if $Q^*Q = I$, i.e., $Q^* = Q^{-1}$.

A matrix $Q \in \mathbb{R}^{n \times n}$ is said to be *orthogonal* if $Q^T Q = I$, i.e., $Q^T = Q^{-1}$.

One can easily verify that

$$Q \text{ is unitary} \Leftrightarrow Q^* \text{ is unitary} \Leftrightarrow Q^T \text{ is unitary} \Leftrightarrow \bar{Q} \text{ is unitary.}$$

In the real case:

$$Q \text{ is orthogonal} \Leftrightarrow Q^T \text{ is orthogonal.}$$

2.7 Lemma

Let $A, B \in \mathbb{F}^{m \times n}$ be two matrices (here $\mathbb{F} = \mathbb{C}$ or $\mathbb{F} = \mathbb{R}$) such that $\langle Ax, y \rangle = \langle Bx, y \rangle$ for all $x \in \mathbb{F}^n$ and $y \in \mathbb{F}^m$. Then $A = B$.

Proof. For any pair of canonical basis vectors $e_j \in \mathbb{F}^n$ and $e_i \in \mathbb{F}^m$ we have $\langle Ae_j, e_i \rangle = a_{ij}$ and $\langle Be_j, e_i \rangle = b_{ij}$, therefore $a_{ij} = b_{ij}$. \square

2.8 Matrices of isometries

- Ⓒ The linear transformation of \mathbb{C}^n defined by a matrix $Q \in \mathbb{C}^{n \times n}$ is an isometry (preserves the standard inner product) iff Q is unitary.
- Ⓓ The linear transformation of \mathbb{R}^n defined by a matrix $Q \in \mathbb{R}^{n \times n}$ is an isometry (preserves the standard inner product) iff Q is orthogonal.

Proof If Q is an isometry, then for any pair of vectors x, y

$$\langle x, y \rangle = \langle Qx, Qy \rangle = (Qy)^* Qx = y^* Q^* Qx = \langle Q^* Qx, y \rangle$$

hence $Q^* Q = I$ due to Lemma 2.7. Conversely: if $Q^* Q = I$, then

$$\langle Qx, Qy \rangle = (Qy)^* Qx = y^* Q^* Qx = y^* x = \langle x, y \rangle,$$

hence Q is an isometry. □

2.9 Group property

If $Q_1, Q_2 \in \mathbb{C}^{n \times n}$ are unitary matrices, then so is $Q_1 Q_2$.

If $Q \in \mathbb{C}^{n \times n}$ is a unitary matrix, then so is Q^{-1} .

In terms of abstract algebra, unitary $n \times n$ matrices make a *group*, denoted by $U(n)$.

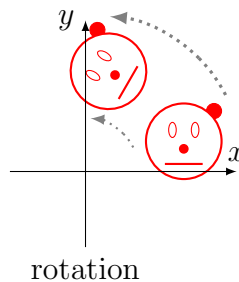
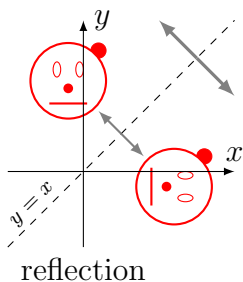
Similarly, orthogonal $n \times n$ matrices make a group, denoted by $O(n)$.

2.10 Orthogonal matrices in 2D

$Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ defines a reflection across the diagonal line $y = x$;

$Q = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$ is a counterclockwise rotation by angle θ .

For a complete description of 2D orthogonal matrices, see Exercises 2.3 and 2.4.



2.11 Characterizations of unitary and orthogonal matrices

A matrix $Q \in \mathbb{C}^{n \times n}$ is unitary iff its columns make an ONB in \mathbb{C}^n .

A matrix $Q \in \mathbb{C}^{n \times n}$ is unitary iff its rows make an ONB in \mathbb{C}^n .

A matrix $Q \in \mathbb{R}^{n \times n}$ is orthogonal iff its columns make an ONB in \mathbb{R}^n .

A matrix $Q \in \mathbb{R}^{n \times n}$ is orthogonal iff its rows make an ONB in \mathbb{R}^n .

Proof. The idea is illustrated by the following diagram:

$$Q^* \quad \times \quad Q \quad = \quad I \quad = \quad Q \quad \times \quad Q^*$$

$$\begin{bmatrix} q_1^* \\ q_2^* \\ \vdots \\ q_n^* \end{bmatrix} \begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} \tilde{q}_1 \\ \tilde{q}_2 \\ \vdots \\ \tilde{q}_n \end{bmatrix} \begin{bmatrix} | & | & & | \\ \tilde{q}_1^* & \tilde{q}_2^* & \cdots & \tilde{q}_n^* \\ | & | & & | \end{bmatrix}$$

Here q_i denotes the i th column of Q and \tilde{q}_i denotes the i th row of Q .

The proof is now a direct inspection. □

2.12 Determinant of unitary matrices

If Q is unitary/orthogonal, then $|\det Q| = 1$.

Proof. We have

$$1 = \det I = \det Q^*Q = \det Q^* \cdot \det Q = \overline{\det Q} \cdot \det Q = |\det Q|^2.$$

If Q is a real orthogonal matrix, then $\det A = \pm 1$. Orthogonal $n \times n$ matrices with determinant 1 make a subgroup of $O(n)$, denoted by $SO(n)$.

2.13 Eigenvalues of unitary matrices

If λ is an eigenvalue of a unitary/orthogonal matrix, then $|\lambda| = 1$.

Proof. If $Qx = \lambda x$ for some $x \neq 0$, then

$$\langle x, x \rangle = \langle Qx, Qx \rangle = \langle \lambda x, \lambda x \rangle = \lambda \bar{\lambda} \langle x, x \rangle = |\lambda|^2 \langle x, x \rangle;$$

therefore $|\lambda|^2 = 1$. □

Note: orthogonal matrices $Q \in \mathbb{R}^{n \times n}$ may not have any real eigenvalues; see the rotation matrix in Example 2.10. But if an orthogonal matrix has real eigenvalues, those equal ± 1 .

2.14 Invariance principle for isometries

Let $T: V \rightarrow V$ be an isometry and $\dim V < \infty$. If a subspace $W \subset V$ is invariant under T , i.e., $TW \subset W$, then so is its orthogonal complement W^\perp , i.e., $TW^\perp \subset W^\perp$.

Proof. The restriction of T to the invariant subspace W is an operator on W . It is an isometry of W , hence it is a bijection $W \rightarrow W$, i.e.,

$$\forall w \in W \quad \exists w' \in W: \quad Tw' = w.$$

Now, if $v \in W^\perp$, then

$$\forall w \in W: \quad \langle w, Tv \rangle = \langle Tw', Tv \rangle = \langle w', v \rangle = 0,$$

hence $Tv \in W^\perp$. This implies $TW^\perp \subset W^\perp$. \square

2.15 Orthogonal decomposition for complex isometries

For any isometry T of a finite dimensional **complex** space V there is an ONB of V consisting of eigenvectors of T .

Proof. Recall that eigenvalues always exist for operators on complex vector spaces (Sect. 0.10). Let λ_1 be an eigenvalue of T with an eigenvector $x_1 \neq 0$. Then $W_1 = \text{span}\{x_1\}$ is a one-dimensional subspace invariant under T . By Section 2.14, the $(n-1)$ -dimensional subspace W_1^\perp is also invariant under T . The restriction of T to W_1^\perp is an operator on W_1^\perp , which is an isometry.

Let λ_2 be an eigenvalue of the operator $T: W_1^\perp \rightarrow W_1^\perp$, with an eigenvector $x_2 \in W_1^\perp$. Then $W_2 = \text{span}\{x_2\}$ is a one-dimensional subspace of W_1^\perp invariant under T .

Since both W_1 and W_2 are invariant under T , we have that $W_1 \oplus W_2$ is a two-dimensional subspace of V invariant under T . By Section 2.14, the $(n-2)$ -dimensional subspace $(W_1 \oplus W_2)^\perp$ is also invariant under T . The restriction of T to $(W_1 \oplus W_2)^\perp$ is an isometry, too.

Then we continue, inductively: at each step we “split off” a one-dimensional subspace invariant under T and reduce the dimensionality of the remaining subspace. Since V is finite dimensional, we eventually exhaust all the dimensions of V and complete the proof. \square

Note: the above theorem is **not true** for **real** vector spaces. An isometry of a real vector space may not have any eigenvectors; see again the rotation matrix in Example 2.10.

2.16 Lemma

Every operator $T: V \rightarrow V$ on a finite dimensional **real** space V has either a one-dimensional or a two-dimensional invariant subspace $W \subset V$.

Proof. Let T be represented by a matrix $A \in \mathbb{R}^{n \times n}$ in some basis. If A has a real eigenvalue, then $Ax = \lambda x$ with some $x \neq 0$, and we get a one-dimensional invariant subspace $\text{span}\{x\}$. If A has no real eigenvalues, then the matrix A , treated as a complex matrix (cf. Sect. 0.10), has a complex eigenvalue $\lambda = a + \mathbf{i}b$, with $a, b \in \mathbb{R}$ and $\mathbf{i} = \sqrt{-1}$, and a complex eigenvector $x + \mathbf{i}y$, with $x, y \in \mathbb{R}^n$. The equation

$$A(x + \mathbf{i}y) = (a + \mathbf{i}b)(x + \mathbf{i}y) = (ax - by) + (bx + ay)\mathbf{i}$$

can be written as

$$\begin{aligned} Ax &= ax - by \\ Ay &= bx + ay \end{aligned}$$

This implies that the subspace $\text{span}\{x, y\}$ is invariant. \square

Note: The above subspace $\text{span}\{x, y\}$ is two-dimensional, unless x and y are linearly dependent. With some extra effort, one can verify that x and y can be linearly dependent only if $b = 0$, i.e. $\lambda \in \mathbb{R}$.

2.17 Orthogonal decomposition for real isometries

Let $T: V \rightarrow V$ be an isometry of a finite dimensional **real** space V . Then $V = V_1 \oplus \cdots \oplus V_p$ for some $p \leq n$, where V_i are mutually orthogonal subspaces, each V_i is T -invariant, and either $\dim V_i = 1$ or $\dim V_i = 2$.

Proof. Use induction on $\dim V$ and apply Sections 2.14 and 2.16. \square

Note: the restriction of the operator T to each two-dimensional invariant subspace V_i is simply a rotation by some angle (as in Example 2.10); this follows from Exercises 2.3 and 2.4.

Recall that two $n \times n$ matrices A and B are *similar* (usually denoted by $A \sim B$) if there exists an invertible matrix X such that $B = X^{-1}AX$. Two matrices are similar if they represent the same linear operator on an n -dimensional space, but under two different bases. In that case X is the change of basis matrix.

2.18 Unitary and orthogonal equivalence

- Ⓒ Two complex matrices $A, B \in \mathbb{C}^{n \times n}$ are said to be *unitary equivalent* if $B = P^{-1}AP$ for some unitary matrix P . This can be also written as $B = P^*AP$ (because $P^{-1} = P^*$).
- Ⓓ Two real matrices $A, B \in \mathbb{R}^{n \times n}$ are said to be *orthogonally equivalent* if $B = P^{-1}AP$ for some orthogonal matrix P . This can be also written as $B = P^TAP$ (because $P^{-1} = P^T$).

Two complex/real matrices are unitary/orthogonally equivalent if they represent the same linear operator on a complex/real n -dimensional inner product space under two different orthonormal bases (ONBs). Then P is the change of basis matrix, which must be unitary/orthogonal, because it changes an ONB to another ONB (cf. Sect. 2.4).

In this course we mostly deal with ONBs, thus unitary/orthogonal equivalence will play the same major role as similarity plays in Linear Algebra. In particular, for any type of matrices we will try to find simplest matrices which are unitary/orthogonal equivalent to matrices of the given type.

2.19 Unitary matrices in their simplest form

Any unitary matrix $Q \in \mathbb{C}^{n \times n}$ is unitary equivalent to a diagonal matrix $D = \text{diag}\{d_1, \dots, d_n\}$, whose diagonal entries belong to the unit circle, i.e. $|d_i| = 1$ for $1 \leq i \leq n$.

Proof. This readily follows from Section 2.15.

2.20 Orthogonal matrices in their simplest form

Any orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ is orthogonally equivalent to a block-diagonal matrix

$$B = \begin{bmatrix} R_{11} & 0 & \cdots & 0 \\ 0 & R_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{mm} \end{bmatrix}$$

where R_{ii} are 1×1 and 2×2 diagonal blocks. Furthermore, each 1×1 block is either $R_{ii} = +1$ or $R_{ii} = -1$, and each 2×2 block is a rotation matrix

$$R_{ii} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{bmatrix}$$

Proof. This readily follows from Section 2.17.

Exercise 2.1. Let $A \in \mathbb{C}^{m \times n}$. Show that

$$\|UA\|_2 = \|AV\|_2 = \|A\|_2$$

for any unitary matrices $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$.

Exercise 2.2. Let $A \in \mathbb{C}^{m \times n}$. Show that

$$\|UA\|_F = \|AV\|_F = \|A\|_F$$

for any unitary matrices $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$. Here $\|\cdot\|_F$ stands for the Frobenius norm.

Exercise 2.3. Let Q be a real orthogonal 2×2 matrix and $\det Q = 1$. Show that

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

for some $\theta \in [0, 2\pi)$.

In geometric terms, Q represents a rotation of \mathbb{R}^2 by angle θ .

Exercise 2.4. Let Q be a real orthogonal 2×2 matrix and $\det Q = -1$. Show that

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

for some $\theta \in [0, 2\pi)$.

Also prove that $\lambda_1 = 1$ and $\lambda_2 = -1$ are the eigenvalues of Q .

In geometric terms, Q represents a reflection of \mathbb{R}^2 across the line spanned by the eigenvector corresponding to $\lambda_1 = 1$.

Chapter 3

Hermitian Matrices

Beginning with this chapter, we will always deal with finite dimensional inner product spaces (unless stated otherwise). Recall that all such spaces can be identified with \mathbb{C}^n or \mathbb{R}^n equipped with the standard inner product (Sect. 2.5). Thus we will mostly deal with these standard spaces.

3.1 Adjoint matrices

Recall that every complex matrix $A \in \mathbb{C}^{m \times n}$ defines a linear transformation $\mathbb{C}^n \rightarrow \mathbb{C}^m$. Its conjugate transpose $A^* \in \mathbb{C}^{n \times m}$ defines a linear transformation $\mathbb{C}^m \rightarrow \mathbb{C}^n$, i.e.,

$$\boxed{\mathbb{C}^n} \begin{array}{c} \xrightarrow{A} \\ \xleftarrow{A^*} \end{array} \boxed{\mathbb{C}^m}$$

Furthermore, for any $x \in \mathbb{C}^n$ and $y \in \mathbb{C}^m$ we have

$$\langle Ax, y \rangle = y^* Ax = (A^* y)^* x = \langle x, A^* y \rangle$$

Likewise, $\langle y, Ax \rangle = \langle A^* y, x \rangle$. In plain words, A can be moved from one argument of the inner product to the other, but it must be changed to A^* .

Likewise, every real matrix $A \in \mathbb{R}^{m \times n}$ defines a linear transformation $\mathbb{R}^n \rightarrow \mathbb{R}^m$. Its transpose $A^T \in \mathbb{R}^{n \times m}$ defines a linear transformation $\mathbb{R}^m \rightarrow \mathbb{R}^n$. Furthermore, for any $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ we have

$$\langle Ax, y \rangle = y^T Ax = (A^T y)^T x = \langle x, A^T y \rangle$$

Thus in the real case, too, A can be moved from one argument of the inner product to the other, but it must be changed to $A^* = A^T$.

3.2 Adjoint transformations

More generally, for any linear transformation $T: V \rightarrow W$ of two inner product vector spaces V and W (both real or both complex) there exists a unique linear transformation $T^*: W \rightarrow V$ such that $\forall v \in V, \forall w \in W$

$$\langle Tv, w \rangle = \langle v, T^* w \rangle.$$

T^* is often called the *adjoint* of T .

T^* generalizes the conjugate transpose of a matrix.

The existence and uniqueness of T^* can be proved by a general argument, avoiding identification of the spaces V and W with \mathbb{C}^n or \mathbb{R}^n . The argument is outlined below, in Sections 3.3 to 3.6. (This part of the course can be skipped.)

3.3 Riesz representation theorem

Let $f \in V^*$, i.e., let f be a linear functional on V . Then there is a unique vector $u \in V$ such that

$$f(v) = \langle v, u \rangle \quad \forall v \in V$$

Proof. Let $B = \{u_1, \dots, u_n\}$ be an ONB in V . Then for any $v = \sum c_i u_i$ we have $f(v) = \sum c_i f(u_i)$ by linearity. Also, for any $u = \sum d_i u_i$ we have $\langle v, u \rangle = \sum c_i \bar{d}_i$. Hence, the vector $u = \sum \frac{f(u_i)}{\bar{f(u_i)}} u_i$ will suffice. To prove the uniqueness of u , assume $\langle v, u \rangle = \langle v, u' \rangle$ for all $v \in V$. Setting $v = u - u'$ gives $\langle u - u', u - u' \rangle = 0$, hence $u = u'$.

3.4 Quasi-linearity

The identity $f \leftrightarrow u$ established in the previous theorem is “quasi-linear” in the following sense: $f_1 + f_2 \leftrightarrow u_1 + u_2$ and $cf \leftrightarrow \bar{c}u$. In the real case, it is perfectly linear, though, and hence it is an isomorphism between V^* and V .

3.5 Remark

If $\dim V = \infty$, then Theorem 3.3 fails. Consider, for example, $V = C[0, 1]$ (real functions) with the inner product $\langle F, G \rangle = \int_0^1 F(x)G(x) dx$. Pick a point $t \in [0, 1]$. Let $f \in V^*$ be a linear functional defined by $f(F) = F(t)$. It does not correspond to any $G \in V$ so that $f(F) = \langle F, G \rangle$. In fact, the lack of such functions G has led mathematicians to the concept of *delta-functions*: a delta-function $\delta_t(x)$ is “defined” by three requirements: $\delta_t(x) \equiv 0$ for all $x \neq t$, $\delta_t(t) = \infty$ and $\int_0^1 F(x)\delta_t(x) dx = F(t)$ for every $F \in C[0, 1]$.

3.6 Existence and uniqueness of adjoint transformation

Let $T: V \rightarrow W$ be a linear transformation. Then there is a unique linear transformation $T^*: W \rightarrow V$ such that $\forall v \in V$ and $\forall w \in W$

$$\langle Tv, w \rangle = \langle v, T^*w \rangle.$$

Proof. Let $w \in W$. Then $f(v) := \langle Tv, w \rangle$ defines a linear functional $f \in V^*$. By the Riesz representation theorem, there is a unique $v' \in V$ such that $f(v) = \langle v, v' \rangle$. Then we define T^* by setting $T^*w = v'$. The linearity of T^* is a routine check. Note that in the complex case the conjugating bar appears twice and thus cancels out. The uniqueness of T^* is obvious. \square

We now return to the main line of our course.

3.7 Relation between $\text{Ker } T^*$ and $\text{Range } T$

Let $T: V \rightarrow W$ be a linear transformation. Then

$$\text{Ker } T^* = (\text{Range } T)^\perp$$

Proof. Suppose $y \in \text{Ker } T^*$, i.e., $T^*y = 0$. Then for every $x \in V$

$$0 = \langle x, 0 \rangle = \langle x, T^*y \rangle = \langle Tx, y \rangle.$$

Thus y is orthogonal to all vectors Tx , $x \in V$, hence $y \in (\text{Range } T)^\perp$.

Conversely, if $y \in (\text{Range } T)^\perp$, then y is orthogonal to all vectors Tx , $x \in V$, hence

$$0 = \langle Tx, y \rangle = \langle x, T^*y \rangle$$

for all $x \in V$. In particular, we can put $x = T^*y$ and see that $\langle T^*y, T^*y \rangle = 0$, hence $T^*y = 0$, i.e., $y \in \text{Ker } T^*$. \square

Next we will assume that $V = W$, in which case T is an operator.

3.8 Selfadjoint operators and matrices

A linear operator $T: V \rightarrow V$ is said to be *selfadjoint* if $T^* = T$.

A square matrix A is said to be *selfadjoint* if $A^* = A$.

In the real case, this is equivalent to $A^T = A$, i.e., A is a symmetric matrix.

In the complex case, selfadjoint matrices are called *Hermitian matrices*.

3.9 Examples

The matrix $\begin{bmatrix} 1 & 3 \\ 3 & 2 \end{bmatrix}$ is symmetric. The matrix $\begin{bmatrix} 1 & 3+i \\ 3-i & 2 \end{bmatrix}$ is Hermitian.

The matrices $\begin{bmatrix} 1 & 3+i \\ 3+i & 2 \end{bmatrix}$ and $\begin{bmatrix} 1+i & 3+i \\ 3-i & 2 \end{bmatrix}$ are **not** Hermitian (why?).

Note: the diagonal components of a Hermitian matrix must be real numbers!

3.10 Hermitian property under unitary equivalence

- Ⓒ If A is a complex Hermitian matrix unitary equivalent to B , then B is also a complex Hermitian matrix.
- Ⓓ If A is a real symmetric matrix orthogonally equivalent to B , then B is also a real symmetric matrix.

Proof. If $A^* = A$ and $B = P^*AP$, then $B^* = P^*A^*P = P^*AP = B$. \square

3.11 Invariance principle for selfadjoint operators

Let T be a selfadjoint operator and a subspace W be T -invariant, i.e., $TW \subset W$. Then W^\perp is also T -invariant, i.e., $TW^\perp \subset W^\perp$.

Proof. If $v \in W^\perp$, then for any $w \in W$ we have $\langle Tv, w \rangle = \langle v, Tw \rangle = 0$, so $Tv \in W^\perp$.

3.12 Spectral Theorem

Let $T: V \rightarrow V$ be a selfadjoint operator. Then there is an ONB consisting of eigenvectors of T , and all the eigenvalues of T are real numbers.

Proof. First let V be a *complex* space. Then we apply Principle 3.11 to construct an ONB of eigenvectors, exactly as we did in Section 2.15.

Now T is represented in the canonical basis $\{e_1, \dots, e_n\}$ by a Hermitian matrix A . In the (just constructed) ONB of eigenvectors, T is represented by a diagonal matrix $D = \text{diag}\{d_1, \dots, d_n\}$, and d_1, \dots, d_n are the eigenvalues of T . Since A and D are unitary equivalent, D must be Hermitian, too (by 3.10). Hence the diagonal components d_1, \dots, d_n of D are real numbers. This completes the proof of Spectral Theorem for complex spaces.

Before we proceed to real spaces, we need to record a useful fact about complex Hermitian matrices. Every Hermitian matrix $A \in \mathbb{C}^{n \times n}$ defines a selfadjoint operator on \mathbb{C}^n . As we just proved, there exists an ONB of eigenvectors. Hence A is unitary equivalent to a diagonal matrix $D = \text{diag}\{d_1, \dots, d_n\}$. Since D is Hermitian, its diagonal entries d_1, \dots, d_n are real numbers. Therefore

$$\boxed{\text{Every Hermitian matrix has only real eigenvalues}} \quad (3.1)$$

Now let V be a *real* space. In the canonical basis $\{e_1, \dots, e_n\}$, the selfadjoint operator $T: V \rightarrow V$ is represented by a real symmetric matrix A . The latter, *considered as a complex matrix*, is Hermitian, thus it has only real eigenvalues (see above). Hence the construction of an ONB of eigenvectors, as done in Section 2.15, works again. The proof is now complete. \square

Note: For every real symmetric matrix A , the characteristic polynomial is $C_A(x) = \prod_i (x - \lambda_i)$, where all λ_i 's are real numbers.

3.13 Characterization of Hermitian matrices

- Ⓒ A complex matrix is Hermitian iff it is unitary equivalent to a diagonal matrix with real diagonal entries.
- Ⓓ A real matrix is symmetric iff it is orthogonally equivalent to a diagonal matrix (whose entries are automatically real).

More generally: if an operator $T: V \rightarrow V$ has an ONB of eigenvectors, and all its eigenvalues are real numbers, then T is self-adjoint.

3.14 Eigendecomposition for Hermitian matrices

Let $A = QDQ^*$, where Q is a unitary matrix and D is a diagonal matrix. Denote by q_i the i th column of Q and by d_i the i th diagonal entry of D . Then

$$Aq_i = d_i q_i, \quad 1 \leq i \leq n$$

i.e. the columns of Q are eigenvectors of A whose eigenvalues are the corresponding diagonal components of D (cf. Section 0.11).

3.15 Inverse of a selfadjoint operator

- (a) If an operator T is selfadjoint and invertible, then so is T^{-1} .
- (b) If a matrix A is selfadjoint and nonsingular, then so is A^{-1} .

Proof. (a) By Spectral Theorem 3.12, there is an ONB $\{u_1, \dots, u_n\}$ consisting of eigenvectors of T , and the eigenvalues $\lambda_1, \dots, \lambda_n$ of T are real numbers. Since T is invertible, $\lambda_i \neq 0$ for all $i = 1, \dots, n$. Now

$$Tu_i = \lambda_i u_i \quad \implies \quad T^{-1}u_i = \lambda_i^{-1}u_i$$

hence T^{-1} has the same eigenvectors $\{u_1, \dots, u_n\}$, and its eigenvalues are the reciprocals of those of T , so they are real numbers, too. Therefore, T^{-1} is selfadjoint.

(b) If A is selfadjoint, then $A = P^*DP$ with a unitary matrix P and a diagonal matrix $D = \text{diag}\{d_1, \dots, d_n\}$ with real diagonal entries. Now

$$A^{-1} = P^{-1}D^{-1}(P^*)^{-1} = P^*D^{-1}P$$

where $D^{-1} = \text{diag}\{d_1^{-1}, \dots, d_n^{-1}\}$ is also a diagonal matrix with real diagonal entries. Therefore A^{-1} is selfadjoint. \square

3.16 Projections

Let $V = W_1 \oplus W_2$, i.e., suppose our vector space V is a direct sum of two subspaces. Recall that for each $v \in V$ there is a unique decomposition $v = w_1 + w_2$ with $w_1 \in W_1$ and $w_2 \in W_2$.

The operator $P: V \rightarrow V$ defined by

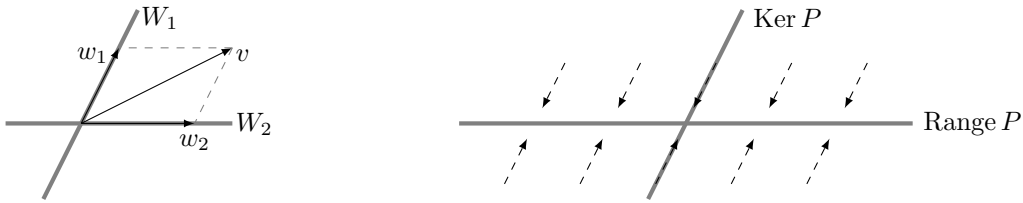
$$Pv = P(w_1 + w_2) = w_2$$

is called *projection* (or *projector*) of V on W_2 along W_1 .

Note that

$$\text{Ker } P = W_1 \quad \text{and} \quad \text{Range } P = W_2.$$

Also note that $P^2 = P$.



3.17 Projections (alternative definition)

An operator $P: V \rightarrow V$ is a projection iff $P^2 = P$.

Proof. $P^2 = P$ implies that for every $v \in V$ we have $P(v - Pv) = 0$, so

$$w_1 := v - Pv \in \text{Ker } P$$

Denoting $w_2 = Pv$ we get $v = w_1 + w_2$ with $w_1 \in \text{Ker } P$ and $w_2 \in \text{Range } P$. Furthermore,

$$Pv = Pw_1 + Pw_2 = 0 + P(Pv) = P^2v = Pv = w_2$$

We also note that $\text{Ker } P \cap \text{Range } P = \{0\}$. Indeed, for any $v \in \text{Range } P$ we have $Pv = v$ (as above) and for any $v \in \text{Ker } P$ we have $Pv = 0$. Thus $v \in \text{Ker } P \cap \text{Range } P$ implies $v = 0$. Hence $V = \text{Ker } P \oplus \text{Range } P$. \square

3.18 “Complimentary” projections

Let $V = W_1 \oplus W_2$. There is a unique projection P_1 on W_1 along W_2 and a unique projection P_2 on W_2 along W_1 . They “complement” each other adding up to the identity:

$$P_1 + P_2 = I.$$

3.19 Orthogonal projections

Let V be an inner product vector space (not necessarily finite dimensional) and $W \subset V$ a finite dimensional subspace. Then the projection on W along W^\perp is called *orthogonal projection* on W .

The assumption $\dim W < \infty$ is made to ensure that $V = W \oplus W^\perp$ (cf. Sect. 1.29).

3.20 Characterization of orthogonal projections

Let P be a projection. Then P is an orthogonal projection if and only if P is selfadjoint.

Proof. Let P be a projection on W_2 along W_1 , and $V = W_1 \oplus W_2$. For any vectors $v, w \in V$ we have $v = v_1 + v_2$ and $w = w_1 + w_2$ with some $v_i, w_i \in W_i$, $i = 1, 2$.

Now, if P is an orthogonal projection, then

$$\langle Pv, w \rangle = \langle v_2, w \rangle = \langle v_2, w_2 \rangle = \langle v, w_2 \rangle = \langle v, Pw \rangle.$$

If P is not an orthogonal projection, then there are $v_1 \in W_1$ and $w_2 \in W_2$ so that $\langle v_1, w_2 \rangle \neq 0$. Then $\langle v_1, Pw_2 \rangle \neq 0 = \langle Pv_1, w_2 \rangle$. \square

Exercise 3.1. Let V be an inner product space and $W \subset V$ a finite dimensional subspace with ONB $\{u_1, \dots, u_n\}$. For every $x \in V$ define

$$P(x) = \sum_{i=1}^n \langle x, u_i \rangle u_i$$

- (i) Prove that $x - P(x) \in W^\perp$, hence P is the orthogonal projection onto W .
- (ii) Prove that $\|x - P(x)\| \leq \|x - z\|$ for every $z \in W$, and that if $\|x - P(x)\| = \|x - z\|$ for some $z \in W$, then $z = P(x)$.

Exercise 3.2. (JPE, May 1999) Let $P \in \mathbb{C}^{n \times n}$ be a projector. Show that $\|P\|_2 \geq 1$ with equality if and only if P is an orthogonal projector.

Positive Definite Matrices

Recall that an inner product $\langle \cdot, \cdot \rangle$ is a complex-valued function of two vector arguments, cf. Section 1.10. It is linear in the first argument and “conjugate linear” in the second.

In the space \mathbb{C}^n , any function $f: \mathbb{C}^n \times \mathbb{C}^n \rightarrow \mathbb{C}$ with the last two properties can be defined by

$$f(x, y) = \langle Ax, y \rangle = y^* Ax,$$

where $A \in \mathbb{C}^{n \times n}$ is a matrix. Then $f(x, y)$ automatically satisfies axioms 2 and 3 of the inner product (Sect. 1.10), as well as the rules (1.4).

But axiom 1 of the inner product (Sect. 1.10) may not hold for any such $f(x, y)$. It holds if and only if

$$\langle Ax, y \rangle = f(x, y) = \overline{f(y, x)} = \overline{\langle Ay, x \rangle} = \langle x, Ay \rangle = \langle A^* x, y \rangle,$$

i.e., iff $A = A^*$, which means that A must be a Hermitian matrix.

Furthermore, if we want $f(x, y)$ satisfy *all* the axioms of the inner product, then we need to ensure the last axiom 4:

$$f(x, x) = \langle Ax, x \rangle > 0 \quad \forall x \neq 0.$$

4.1 Positive definite matrices

A matrix A is said to be *positive definite* if it is Hermitian and

$$\langle Ax, x \rangle = x^* Ax > 0 \quad \forall x \neq 0.$$

In the real case, a matrix A is *positive definite* if it is symmetric and

$$\langle Ax, x \rangle = x^T Ax > 0 \quad \forall x \neq 0.$$

If we replace “ > 0 ” with “ ≥ 0 ” in the above formulas, we get the definition of *positive semi-definite*³ matrices.

³Perhaps, a more natural term would be “non-negative definite”, but we will use the official name *positive semi-definite*.

Symmetric positive definite matrices play important role in optimization theory (the minimum of a function of several variables corresponds to a critical point where the matrix of second order partial derivatives is positive definite) and probability theory (the covariance matrix of a random vector is symmetric positive definite).

Interestingly, the condition $\langle Ax, x \rangle > 0$ for all $x \in \mathbb{C}^n$ implies that the matrix A is Hermitian, and therefore positive definite. We will prove this in Sections 4.2 and 4.3. (It is just a curious fact, with little importance, so the next two sections can be skipped.)

4.2 Lemma

Let A, B be complex matrices. If $\langle Ax, x \rangle = \langle Bx, x \rangle$ for all $x \in \mathbb{C}^n$, then $A = B$. (Proof: see exercises.)

Note: this lemma holds only in complex spaces, it fails in real spaces (see exercises).

4.3 Sufficient condition for positive definiteness

If A is a complex matrix such that $\langle Ax, x \rangle \in \mathbb{R}$ for all $x \in \mathbb{C}^n$, then A is Hermitian. In particular, if $\langle Ax, x \rangle > 0$ for all non-zero vectors $x \in \mathbb{C}^n$, then A is positive definite.

Proof. We have

$$\langle Ax, x \rangle = \overline{\langle Ax, x \rangle} = \langle x, Ax \rangle = \langle A^*x, x \rangle,$$

hence $A = A^*$ by Lemma 4.2. Now the condition $\langle Ax, x \rangle > 0$ for all $x \neq 0$ implies that A is positive definite. \square

Next we define positive definite and positive semi-definite operators in general spaces in a more abstract way; see Sections 4.4–4.11 below. (These sections can be skipped in class and assigned to more advanced students as a reading project.)

4.4 Bilinear forms

A *bilinear form* on a complex vector space V is a complex-valued function of two vector arguments, i.e., $f: V \times V \rightarrow \mathbb{C}$ such that

$$\begin{aligned} f(u_1 + u_2, v) &= f(u_1, v) + f(u_2, v) \\ f(cu, v) &= cf(u, v) \\ f(u, v_1 + v_2) &= f(u, v_1) + f(u, v_2) \\ f(u, cv) &= \bar{c}f(u, v) \end{aligned}$$

for all vectors $u, v, u_i, v_i \in V$ and scalars $c \in \mathbb{C}$. In other words, f must be linear in the first argument and “conjugate linear” in the second.

A bilinear form on a real vector space V is a mapping $f: V \times V \rightarrow \mathbb{R}$ that satisfies the same properties, except c is a real scalar, i.e., $\bar{c} = c$.

4.5 Representation of bilinear forms

Let V be a finite dimensional inner product space. Then for every bilinear form f on V there is a unique linear operator $T: V \rightarrow V$ such that

$$f(u, v) = \langle Tu, v \rangle \quad \forall u, v \in V$$

Proof. For every $v \in V$ the function $g(u) = f(u, v)$ is linear in u , so by Riesz Representation Theorem 3.3 there is a vector $w \in V$ such that $f(u, v) = \langle u, w \rangle$. Define a map $S: V \rightarrow V$ by $Sv = w$. It is then a routine check that S is linear. Setting $T = S^*$ proves the existence. The uniqueness is obvious. \square

4.6 Corollary

- © Every bilinear form on \mathbb{C}^n can be represented by $f(x, y) = \langle Ax, y \rangle$ with some $A \in \mathbb{C}^{n \times n}$.
- Ⓜ Every bilinear form on \mathbb{R}^n can be represented by $f(x, y) = \langle Ax, y \rangle$ with some $A \in \mathbb{R}^{n \times n}$.

Bilinear forms generalize the notion of inner product in abstract spaces. In order for a bilinear form to become an inner product, though, it needs two additional properties: conjugate symmetry $f(x, y) = \overline{f(y, x)}$ and the positivity $f(x, x) > 0$ for all $x \neq 0$.

4.7 Hermitian/symmetric forms

A bilinear form f on a complex space V is *Hermitian* if $f(u, v) = \overline{f(v, u)} \quad \forall u, v \in V$.
A bilinear form f on a real space V is *symmetric* if $f(u, v) = f(v, u) \quad \forall u, v \in V$.

4.8 Quadratic forms

For a Hermitian bilinear form f , the function $q: V \rightarrow \mathbb{R}$ defined by $q(u) = f(u, u)$ is called the *quadratic form* associated with f . Note that $q(u) \in \mathbb{R}$ even in the complex case, because $f(u, u) = \overline{f(u, u)}$.

4.9 Theorem

A linear operator $T: V \rightarrow V$ is selfadjoint if and only if the bilinear form $f(u, v) = \langle Tu, v \rangle$ is Hermitian (in the real case, symmetric).

Proof. If T is selfadjoint, then $f(u, v) = \langle Tu, v \rangle = \langle u, Tv \rangle = \overline{\langle Tv, u \rangle} = \overline{f(v, u)}$. If f is Hermitian, then $\langle u, Tv \rangle = \overline{\langle Tv, u \rangle} = \overline{f(v, u)} = f(u, v) = \langle Tu, v \rangle = \langle u, T^*v \rangle$, therefore $T = T^*$. \square

Thus, Hermitian bilinear forms on \mathbb{C}^n are defined by Hermitian matrices.

4.10 Positive definite forms and operators

A Hermitian (symmetric) bilinear form f on a vector space V is said to be *positive definite* if $f(u, u) > 0$ for all $u \neq 0$. A selfadjoint operator $T: V \rightarrow V$ is said to be *positive definite* if $\langle Tu, u \rangle > 0$ for all $u \neq 0$.

If we replace “ > 0 ” with “ ≥ 0 ” in the above formulas, we get the definition of *positive semi-definite* bilinear forms and operators.

4.11 Theorem

The following are equivalent:

- (a) a bilinear form $f(u, v)$ is an inner product.
- (b) $f(u, v) = \langle Tu, v \rangle$, where T is a positive definite operator.

We now return to the main line of our course.

Special note: The rest of Chapter 4 and the entire Chapter 5 are extremely important. The instructor should go slow, cover every minor detail, and not hesitate spending extra time explaining things. The students are advised to attend classes, closely follow the lectures, and do their best to fully grasp the material. Covering this part of the course well will allow the class to move faster through subsequent chapters. On the other hand, failure to fully understand this part of the course will cause major troubles later, and it might be virtually impossible to learn some of the subsequent topics.

Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix. By (3.1), all its eigenvalues $\lambda_1, \dots, \lambda_n$ are real numbers, hence they can be ordered. In particular, we can define

$$\lambda_{\min} = \min_{1 \leq i \leq n} \lambda_i \quad \text{and} \quad \lambda_{\max} = \max_{1 \leq i \leq n} \lambda_i. \quad (4.1)$$

4.12 Properties of Hermitian matrices

Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix. Then

(a) For every vector $x \in \mathbb{C}^n$ we have

$$\langle Ax, x \rangle \in \mathbb{R}. \quad (4.2)$$

(b) For every vector $x \in \mathbb{C}^n$ we have

$$\lambda_{\min} \|x\|_2^2 \leq \langle Ax, x \rangle \leq \lambda_{\max} \|x\|_2^2. \quad (4.3)$$

(c) We have

$$\langle Ax, x \rangle = \lambda_{\min} \|x\|_2^2 \iff Ax = \lambda_{\min} x \quad (4.4)$$

and

$$\langle Ax, x \rangle = \lambda_{\max} \|x\|_2^2 \iff Ax = \lambda_{\max} x \quad (4.5)$$

Part (c) implies that the left and right inequalities in (4.3) cannot be improved – they turn into equalities for certain nonzero vectors x .

Proof. By Spectral Theorem 3.12, there is an ONB $\{u_1, \dots, u_n\}$ of eigenvectors of A , corresponding to the eigenvalues $\lambda_1, \dots, \lambda_n$. Then for any vector $x = \sum c_i u_i$ we have $Ax = \sum \lambda_i c_i u_i$. Due to (1.20), we have

$$\begin{aligned} \langle Ax, x \rangle &= \lambda_1 c_1 \bar{c}_1 + \dots + \lambda_n c_n \bar{c}_n \\ &= \lambda_1 |c_1|^2 + \dots + \lambda_n |c_n|^2. \end{aligned}$$

Since $\lambda_i \in \mathbb{R}$, we conclude that $\langle Ax, x \rangle \in \mathbb{R}$ thus proving part (a). Next,

$$\langle Ax, x \rangle \geq \lambda_{\min}(|c_1|^2 + \cdots + |c_n|^2) = \lambda_{\min}\|x\|_2^2,$$

where we used (1.21). We also see that

$$\langle Ax, x \rangle - \lambda_{\min}\|x\|_2^2 = \sum_{i=1}^n |c_i|^2(\lambda_i - \lambda_{\min}),$$

where all the terms are nonnegative because $\lambda_i \geq \lambda_{\min}$. Thus, the left inequality in (4.3) turns into an equality if and only if $c_i = 0$ for all i 's such that $\lambda_i > \lambda_{\min}$. This means exactly that x is an eigenvector corresponding to the smallest eigenvalue λ_{\min} . The left inequality in (4.3) and the claim (4.4) are now proved.

Similarly,

$$\langle Ax, x \rangle \leq \lambda_{\max}(|c_1|^2 + \cdots + |c_n|^2) = \lambda_{\max}\|x\|_2^2$$

and

$$\lambda_{\max}\|x\|_2^2 - \langle Ax, x \rangle = \sum_{i=1}^n |c_i|^2(\lambda_{\max} - \lambda_i).$$

Thus, the right inequality in (4.3) turns into an equality if and only if $c_i = 0$ for all i 's such that $\lambda_{\max} > \lambda_i$, which means exactly that x is an eigenvector corresponding to the largest eigenvalue λ_{\max} . The right inequality in (4.3) and the claim (4.5) are now proved, too. \square

4.13 Eigenvalues of positive definite matrices

Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix. It is positive definite iff all its eigenvalues are positive.

Similarly, a Hermitian matrix $A \in \mathbb{C}^{n \times n}$ is positive semi-definite iff all its eigenvalues are nonnegative.

Proof. This readily follows from Section 4.12. \square

4.14 Inverse of a positive definite matrix

If a matrix A is positive definite, then so is A^{-1} .

4.15 Characterization of positive definite matrices

A matrix $A \in \mathbb{C}^{n \times n}$ is positive definite iff there is a nonsingular matrix B such that $A = B^*B$.

Proof. “ \Leftarrow ” If $A = B^*B$, then $A^* = B^*(B^*)^* = A$, so A is Hermitian. Now

$$\langle Ax, x \rangle = \langle Bx, Bx \rangle > 0 \quad \forall x \neq 0,$$

so A is positive definite (note that $x \neq 0$ implies $Bx \neq 0$, because B is nonsingular).

“ \Rightarrow ” According to Sections 3.13 and 4.13, $A = P^{-1}DP$, where $D = \text{diag}\{d_1, \dots, d_n\}$ is a diagonal matrix with positive diagonal entries $d_i > 0$, and P is a unitary matrix. We construct a new diagonal matrix

$$D_{1/2} = \text{diag}\{\sqrt{d_1}, \dots, \sqrt{d_n}\}.$$

Now $A = B^2$ where $B = P^{-1}D_{1/2}P$. Lastly we note that B is Hermitian, due to Section 3.13, hence $A = B^*B$. \square

4.16 Characterization of positive semi-definite matrices

A matrix $A \in \mathbb{C}^{n \times n}$ is positive semi-definite if and only if there is a matrix B (not necessarily nonsingular) such that $A = B^*B$.

The proof is similar to the above, so we omit it.

4.17 Full rank and rank deficient matrices

A matrix $A \in \mathbb{C}^{m \times n}$ is said to have *full rank* if

$$\text{rank } A = \min\{m, n\}$$

Otherwise, A is said to be *rank deficient*.

4.18 Products A^*A and AA^*

Let A be a rectangular $m \times n$ matrix. Then the matrices A^*A and AA^* are Hermitian and positive semi-definite. If $m \neq n$ and A has full rank, then the smaller of the two matrices A^*A and AA^* is positive definite.

Proof. First we verify the Hermitian property:

$$(A^*A)^* = A^*(A^*)^* = A^*A$$

and similarly $(AA^*)^* = AA^*$. Next we verify positive semidefiniteness:

$$\langle A^*Ax, x \rangle = \langle Ax, Ax \rangle \geq 0$$

for any $x \in \mathbb{C}^n$ and similarly

$$\langle AA^*y, y \rangle = \langle A^*y, A^*y \rangle \geq 0$$

for any $y \in \mathbb{C}^m$.

Now let A have full rank. If $m \geq n$, then $\text{Ker}(A) = \{0\}$, hence $Ax \neq 0$ for any $0 \neq x \in \mathbb{C}^n$, so that $\langle A^*Ax, x \rangle = \langle Ax, Ax \rangle > 0$, which implies that A^*A is positive definite. If $m < n$, then $\text{Range}(A) = \mathbb{C}^m$, hence for any $0 \neq y \in \mathbb{C}^m$ there is $0 \neq x \in \mathbb{C}^n$ such that $y = Ax$. Therefore

$$\langle A^*y, x \rangle = \langle y, Ax \rangle = \langle y, y \rangle > 0$$

which implies $A^*y \neq 0$. Therefore

$$\langle AA^*y, y \rangle = \langle A^*y, A^*y \rangle > 0$$

which implies that AA^* is positive definite. □

4.19 Spectral radius

Let $A \in \mathbb{C}^{n \times n}$ be a square matrix and $\lambda_1, \dots, \lambda_n$ its eigenvalues. Then

$$\rho_A = \max_{1 \leq i \leq n} |\lambda_i|$$

is called the *spectral radius* of A . It is the radius of the smallest disk $\{|z| \leq r\}$ in the complex plane that contains all the eigenvalues of A .

Note: if A is a Hermitian matrix, then

$$\rho_A = \max\{|\lambda_{\max}|, |\lambda_{\min}|\}$$

in the notation of (4.1).

4.20 Spectral radius for Hermitian matrices

If $A \in \mathbb{C}^{n \times n}$ is a Hermitian matrix, then

$$\|A\|_2 = \rho_A.$$

Proof. We use the notation of the proof in Section 4.12:

$$\begin{aligned} \|Ax\|_2^2 &= \langle Ax, Ax \rangle = \lambda_1^2 |c_1|^2 + \cdots + \lambda_n^2 |c_n|^2 \\ &\leq \left[\max_{1 \leq i \leq n} \lambda_i^2 \right] \sum_{j=1}^n |c_j|^2 = \rho_A^2 \|x\|_2^2. \end{aligned}$$

Taking the square root and assuming $x \neq 0$ we get

$$\|Ax\|_2 / \|x\|_2 \leq \rho_A.$$

Now there exists $k \in [1, n]$ such that $|\lambda_k| = \rho_A$. Let $x = u_k$, i.e., x is an eigenvector corresponding to λ_k . Then $c_k = 1$ and $c_i = 0$ for all $i \neq k$. Therefore

$$\|Ax\|_2^2 = \lambda_k^2 = \rho_A^2.$$

Combining the above estimates gives us

$$\|A\|_2 = \max_{x \neq 0} \|Ax\|_2 / \|x\|_2 = \rho_A$$

as desired. □

The next theorem is particularly important.

4.21 Theorem on the 2-norm of matrices

For any matrix $A \in \mathbb{C}^{m \times n}$ we have

$$\|A\|_2^2 = \|A^*\|_2^2 = \|A^*A\|_2 = \|AA^*\|_2 = \lambda_{\max}$$

where λ_{\max} denotes the largest eigenvalue of both A^*A and AA^* .

Note: This theorem gives a practical method for computing $\|A\|_2$ for small matrices, where $m = 2$ or $n = 2$. In that case one of the two matrices A^*A and AA^* is of size 2×2 , thus its eigenvalues are the roots of a quadratic polynomial.

Proof of Theorem 4.21 is long and will be done in four steps. In the proof, $\|\cdot\|$ will always denote the 2-norm.

Lemma. For every vector $z \in \mathbb{C}^n$ we have $\|z\| = \max_{\|y\|=1} |\langle y, z \rangle|$.

Proof. Indeed, by the Cauchy-Schwarz inequality

$$|\langle y, z \rangle| \leq \langle y, y \rangle^{1/2} \langle z, z \rangle^{1/2} = \|z\|$$

and the equality is attained whenever y is parallel to z . So we can set $y = \pm \frac{z}{\|z\|}$ and achieve the maximum. \square

Step 1. To prove that $\|A\| = \|A^*\|$ we write

$$\begin{aligned} \|A\| &= \sup_{\|x\|=1} \|Ax\| = \sup_{\|x\|=1} \sup_{\|y\|=1} |\langle y, Ax \rangle| = \sup_{\|x\|=1} \sup_{\|y\|=1} |\langle A^*y, x \rangle| \\ &= \sup_{\|y\|=1} \sup_{\|x\|=1} |\langle x, A^*y \rangle| = \sup_{\|y\|=1} \|A^*y\| = \|A^*\|. \end{aligned}$$

Step 2. To prove that $\|A\|^2 = \|A^*A\|$ we write

$$\|A^*A\| = \sup_{\|x\|=1} \|A^*Ax\| = \sup_{\|x\|=1} \sup_{\|y\|=1} |\langle y, A^*Ax \rangle| = \sup_{\|x\|=1} \sup_{\|y\|=1} |\langle Ay, Ax \rangle|.$$

Then we again use the Cauchy-Schwarz inequality:

$$|\langle Ay, Ax \rangle| \leq \|Ax\| \|Ay\| \leq \|A\| \|A\| = \|A\|^2$$

hence $\|A^*A\| \leq \|A\|^2$. On the other hand,

$$\|A^*A\| = \sup_{\|x\|=1} \sup_{\|y\|=1} |\langle Ay, Ax \rangle| \geq \sup_{\|x\|=1} |\langle Ax, Ax \rangle| = \|A\|^2.$$

Combining the upper and lower bounds gives us $\|A^*A\| = \|A\|^2$.

Step 3. Using an obvious symmetry we conclude that $\|A^*\|^2 = \|AA^*\|$.

Step 4. According to Section 4.20, we have

$$\|A^*A\|_2 = \max |\lambda_i(A^*A)|.$$

Recall that A^*A is a Hermitian positive semi-definite matrix, so its eigenvalues $\lambda_i(A^*A)$ are real and ≥ 0 , hence $\max |\lambda_i(A^*A)| = \lambda_{\max}(A^*A)$, the largest eigenvalue of A^*A . The same argument applies to AA^* . In particular, we see that

$$\lambda_{\max}(A^*A) = \lambda_{\max}(AA^*).$$

This completes the proof of Theorem 4.21. \square

4.22 Example

Let $A = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \in \mathbb{C}^{2 \times 1}$. For any unit vector $x \in \mathbb{C}^1$ we can write $x = [e^{i\theta}]$, therefore $Ax = \begin{bmatrix} 3e^{i\theta} \\ 4e^{i\theta} \end{bmatrix}$ and

$$\|Ax\| = \sqrt{|3e^{i\theta}|^2 + |4e^{i\theta}|^2} = \sqrt{3^2 + 4^2} = 5$$

which implies $\|A\| = 5$. Now let us find the norm of $A^* = [3 \ 4] \in \mathbb{C}^{1 \times 2}$, i.e., $\|A^*\| = \sup_{\|y\|=1} \|A^*y\|$. For simplicity, we will only use *real* unit vectors $y \in \mathbb{C}^2$, which can be described by $y = \begin{bmatrix} \cos \varphi \\ \sin \varphi \end{bmatrix}$ for $\varphi \in [0, 2\pi]$. We have $A^*y = 3 \cos \varphi + 4 \sin \varphi$, thus $\|A^*y\| = |3 \cos \varphi + 4 \sin \varphi|$. Finding the maximum of this function (over the interval $0 \leq \varphi \leq 2\pi$) is a Calculus-I problem: the maximum is achieved at $\cos \varphi = \pm 3/5$ and $\sin \varphi = \pm 4/5$, and we get

$$\|A^*\| = \max_{\|y\|=1} \|A^*y\| = \frac{9}{5} + \frac{16}{5} = \frac{25}{5} = 5.$$

We see that, indeed, $\|A\| = \|A^*\|$. Note that $A^*A = [25] \in \mathbb{C}^{1 \times 1}$, so obviously $\|A^*A\| = 25$, in full agreement with Theorem 4.21.

4.23 Corollary for the 2-norm of matrices

Let λ_{\max} again denote the largest eigenvalue of A^*A . Then we have

$$\|Ax\|_2 = \|A\|_2 \|x\|_2 \iff A^*Ax = \lambda_{\max}x.$$

Hence, the supremum in Section 1.7 is attained (on the eigenvectors of A^*A corresponding to λ_{\max}) and can be replaced by maximum. Moreover, this implies that the 2-norm of a real matrix is the same, whether it is computed in the complex space or in the real space.

Proof. On the one hand

$$\|Ax\|_2^2 = \langle Ax, Ax \rangle = \langle A^*Ax, x \rangle$$

and on the other hand

$$\|A\|_2^2 = \lambda_{\max},$$

so for any vector x with $\|x\| = 1$ we have

$$\|Ax\|_2^2 = \|A\|_2^2 \iff \langle A^*Ax, x \rangle = \lambda_{\max},$$

and lastly we use Section 4.12. □

Exercise 4.1. Let $A \in \mathbb{C}^{n \times n}$ satisfy $A^* = -A$. Show that the matrix $I - A$ is invertible. Then show that the matrix $(I - A)^{-1}(I + A)$ is unitary.

Exercise 4.2. Let $A = (a_{ij})$ be a complex $n \times n$ matrix. Assume that $\langle Ax, x \rangle = 0$ for all $x \in \mathbb{C}^n$. Prove that

(a) $a_{ii} = 0$ for $1 \leq i \leq n$ by substituting $x = e_i$

(b) $a_{ij} = 0$ for $i \neq j$ by substituting $x = pe_i + qe_j$ then using (a) and putting $p, q = \pm 1, \pm \mathbf{i}$ (here $\mathbf{i} = \sqrt{-1}$) in various combinations.

Conclude that $A = 0$.

Exercise 4.3. Let A, B be complex $n \times n$ matrices such that $\langle Ax, x \rangle = \langle Bx, x \rangle$ for all $x \in \mathbb{C}^n$. Use the previous exercise to prove that $A = B$.

Exercise 4.4. Find a real 2×2 matrix $A \neq 0$ such that $\langle Ax, x \rangle = 0$ for all $x \in \mathbb{R}^2$. Thus find two real 2×2 matrices A and B such that $\langle Ax, x \rangle = \langle Bx, x \rangle$ for all $x \in \mathbb{R}^2$, but $A \neq B$.

Exercise 4.5. Find a real 2×2 matrix A such that $\langle Ax, x \rangle > 0$ for all $x \in \mathbb{R}^2$, but A is not positive definite.

Singular Value Decomposition

Recall that every matrix $A \in \mathbb{C}^{m \times n}$ defines a linear transformation $\mathbb{C}^n \rightarrow \mathbb{C}^m$. Let B be an ONB in \mathbb{C}^n and B' be ONB in \mathbb{C}^m . Then T is represented in the bases B and B' by the matrix U^*AV , where $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ are unitary matrices. The following theorem shows that one can always find two bases B and B' so that the matrix U^*AV will be diagonal. This can be written as $U^*AV = D$ or equivalently, $A = UDV^*$.

$$\begin{array}{c}
 \text{SVD } (m > n) \\
 \begin{array}{ccccccc}
 \boxed{} & = & \boxed{} & \times & \begin{array}{|c|} \hline 0 \\ \hline \\ \hline 0 \\ \hline \end{array} & \times & \boxed{} \\
 A & & U & & D & & V^*
 \end{array} \\
 \\
 \text{SVD } (m < n) \\
 \begin{array}{ccccccc}
 \boxed{} & = & \boxed{} & \times & \begin{array}{|c|} \hline \\ \hline 0 \\ \hline \\ \hline \end{array} & \times & \boxed{} \\
 A & & U & & D & & V^*
 \end{array}
 \end{array}$$

Note: $D \in \mathbb{C}^{m \times n}$ is said to be diagonal if $D_{ij} = 0$ for $i \neq j$. It has exactly $p = \min\{m, n\}$ diagonal entries and can be denoted by $D = \text{diag}\{d_1, \dots, d_p\}$.

5.1 Singular value decomposition (SVD)

Let $A \in \mathbb{C}^{m \times n}$ and denote $p = \min\{m, n\}$. Denote the rank of A by r ($0 \leq r \leq p$). Then there are unitary matrices $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ and a real diagonal matrix $D = \text{diag}\{\sigma_1, \dots, \sigma_p\} \in \mathbb{R}^{m \times n}$ such that

$$A = UDV^* \tag{5.1}$$

and

$$\sigma_1 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_p = 0,$$

The matrix D is uniquely determined by A .

The matrices U and V are *not* unique, for every A there are many choices of U and V .

Proof. Observe that (5.1) is equivalent to $A^* = VD^TU^*$, which is an SVD for A^* . Therefore, A has an SVD if and only if A^* does. This allows us to assume that $m \geq n$, without loss of generality.

We now fix $l = m - n \geq 0$ and use induction on n (for every $n \geq 1$ we automatically take $m = n + l$).

Denote $\sigma_1 = \|A\|_2$. If $\sigma_1 = 0$, then A is a zero matrix ($a_{ij} = 0$ for all i, j), and an SVD is constructed trivially: D is a zero matrix and U and V are arbitrary unitary matrices. So let us assume that $\sigma_1 > 0$.

Due to Corollary 4.23, there is a unit vector $v_1 \in \mathbb{C}^n$ such that $\|Av_1\|_2 = \|A\|_2 = \sigma_1$. Then the vector $u_1 = Av_1/\sigma_1$ is a unit vector in \mathbb{C}^m .

According to Section 1.26, we can extend v_1 to an ONB $\{v_1, v'_2, \dots, v'_n\}$ in \mathbb{C}^n and u_1 to an ONB $\{u_1, u'_2, \dots, u'_m\}$ in \mathbb{C}^m . Let V_1 denote the unitary matrix whose columns are v_1, v'_2, \dots, v'_n and U_1 denote the unitary matrix whose columns are u_1, u'_2, \dots, u'_m . Then the product $S = U_1^*AV_1$ has the following structure (as one can verify directly):

$$U_1^*AV_1 = S = \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix} \quad (*)$$

If $n = 1$, then $S = \begin{bmatrix} \sigma_1 \\ 0 \end{bmatrix}$ is a diagonal matrix yielding an SVD

$$A = U_1SV_1^*$$

for A . This is the basis for our induction.

Suppose now $n > 1$, observe that

$$\|S\|_2 = \|S^*\|_2 \geq \left\| \begin{bmatrix} \sigma_1 & 0 \\ w & B^* \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} \sigma_1 \\ w \end{bmatrix} \right\|_2 = \sqrt{\sigma_1^2 + w^*w}.$$

On the other hand, the matrices U_1^* and V_1 are unitary, hence we have

$$\|S\|_2 = \|U_1^*AV_1\|_2 = (\text{Exercise 2.1}) = \|A\|_2 = \sigma_1.$$

Comparing the above two formulas shows that $w = 0$. The matrix B has dimensions $(m - 1) \times (n - 1)$, thus by our inductive assumption it has an SVD, let us denote it by $B = \hat{U}\hat{D}\hat{V}^*$. Now it can be easily verified that

$$A = \underbrace{U_1 \begin{bmatrix} 1 & 0 \\ 0 & \hat{U} \end{bmatrix}}_U \underbrace{\begin{bmatrix} \sigma_1 & 0 \\ 0 & \hat{D} \end{bmatrix}}_D \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & \hat{V}^* \end{bmatrix} V_1^*}_{V^*}$$

which gives us an SVD for A .

To prove the uniqueness of D , observe that

$$A^*A = VD^TDV^* \quad \text{and} \quad AA^* = UDD^TU^*$$

i.e., $\sigma_1^2, \dots, \sigma_p^2$ are the eigenvalues of both A^*A and AA^* , hence they are uniquely determined by A . \square

Incidentally, the last part of the proof shows that the matrices A^*A and AA^* have common non-zero eigenvalues.

Note also that the columns of U are eigenvectors of AA^* and the columns of V are eigenvectors of A^*A , according to Section 3.14.

5.2 Singular values and singular vectors

- The positive numbers $\sigma_1, \dots, \sigma_r$ are called *singular values* of A .
- The columns v_1, \dots, v_n of the matrix V (not those of V^*) are called *right singular vectors* for A .
- The columns u_1, \dots, u_m of the matrix U are called *left singular vectors* for A .

5.3 Real SVD

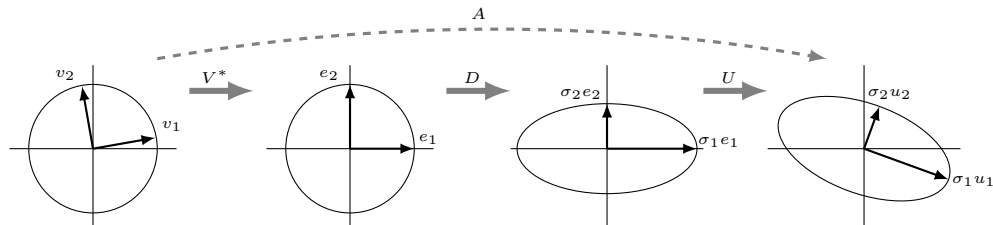
If $A \in \mathbb{R}^{m \times n}$ is a *real* matrix, then it has a *real* SVD

$$A = UDV^T \tag{5.2}$$

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are *real* orthogonal matrices.

Proof is just a repetition of the proof in Section 5.1, except v_1, v'_2, \dots, v'_n and u_1, u'_2, \dots, u'_m must be real vectors. \square

The following diagram illustrates the SVD; see also Section 5.4:



5.4 SVD analysis

For $1 \leq i \leq r$ we have

$$Av_i = \sigma_i u_i, \quad A^* u_i = \sigma_i v_i.$$

We also have

$$\begin{aligned} \text{Ker } A &= \text{span}\{v_{r+1}, \dots, v_n\}, & \text{Ker } A^* &= \text{span}\{u_{r+1}, \dots, u_m\} \\ \text{Range } A &= \text{span}\{u_1, \dots, u_r\}, & \text{Range } A^* &= \text{span}\{v_1, \dots, v_r\} \end{aligned}$$

and

$$\text{rank } A = \text{rank } A^* = r.$$

Here is a diagram illustrating the previous relations:

$$\begin{array}{ccccccc} & & A & & A^* & & \\ & & & & & & \\ & & \times \sigma_1 & & \times \sigma_1 & & \\ v_1 & \xrightarrow{\quad} & u_1 & \xrightarrow{\quad} & v_1 & & \\ & & \times \sigma_2 & & \times \sigma_2 & & \\ v_2 & \xrightarrow{\quad} & u_2 & \xrightarrow{\quad} & v_2 & & \\ \vdots & & \vdots & & \vdots & & \\ & & \times \sigma_r & & \times \sigma_r & & \\ v_r & \xrightarrow{\quad} & u_r & \xrightarrow{\quad} & v_r & & \\ v_{r+1} & \rightarrow & 0 & \rightarrow & 0 & & \\ \vdots & & \vdots & & \vdots & & \\ v_n & \rightarrow & 0 & \rightarrow & 0 & & \\ & & & & u_m & & \end{array}$$

5.5 Useful relations - I

For any matrix $A \in \mathbb{C}^{m \times n}$, an SVD for A^* is

$$A^* = VD^T U^*,$$

hence A and A^* have the same singular values. In particular,

$$\|A\|_2 = \|A^*\|_2 = \|D\|_2 = \|D^T\|_2 = \sigma_1.$$

If $A \in \mathbb{C}^{n \times n}$ is a *square* invertible matrix, then

$$A^{-1} = VD^{-1}U^*$$

where $D^{-1} = \text{diag}\{\sigma_1^{-1}, \dots, \sigma_n^{-1}\}$, and

$$\|A^{-1}\|_2 = \|D^{-1}\|_2 = \sigma_n^{-1}.$$

If $A \in \mathbb{C}^{n \times n}$ is a Hermitian matrix with eigenvalues $\lambda_1, \dots, \lambda_n$, then its singular values are $|\lambda_1|, \dots, |\lambda_n|$ (this follows from Section 3.13).

5.6 Computation of SVD for small matrices

One can manually compute an SVD of an $m \times 2$ matrix as follows. First we form the 2×2 matrix A^*A . Then we find its eigenvalues $\lambda_1 \geq \lambda_2 \geq 0$ by solving the corresponding quadratic equation. (Since A^*A is a Hermitian positive semi-definite matrix, its eigenvalues are real non-negative numbers.) Then we find the corresponding unit eigenvectors v_1, v_2 of the matrix A^*A . Then we compute $\sigma_1 = \sqrt{\lambda_1}$, $\sigma_2 = \sqrt{\lambda_2}$ and $u_1 = \sigma_1^{-1}Av_1$, $u_2 = \sigma_2^{-1}Av_2$. (Recall that u_1 and u_2 must be unit vectors orthogonal to each other.) Lastly, if $m > 2$, we extend the orthonormal set $\{u_1, u_2\}$ to an ONB in \mathbb{C}^m arbitrarily.

5.7 Reduced SVD

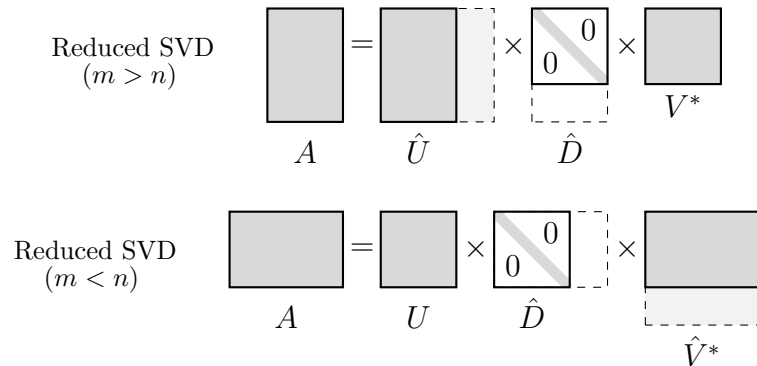
Let $A \in \mathbb{C}^{m \times n}$ with $m > n$. Then there is a matrix $\hat{U} \in \mathbb{C}^{m \times n}$ with orthonormal columns, a unitary matrix $V \in \mathbb{C}^{n \times n}$ and a square diagonal matrix $\hat{D} = \text{diag}\{\sigma_1, \dots, \sigma_n\}$ such that

$$A = \hat{U}\hat{D}V^*. \tag{5.3}$$

For $m < n$, the reduced SVD is similar: there exist a unitary matrix $U \in \mathbb{C}^{m \times m}$, a matrix $\hat{V} \in \mathbb{C}^{n \times m}$ with orthonormal columns, and a square diagonal matrix $\hat{D} = \text{diag}\{\sigma_1, \dots, \sigma_m\}$ such that

$$A = U\hat{D}\hat{V}^*, \tag{5.4}$$

Proof. Suppose $m > n$. Then we take the (full) SVD $A = UDV^*$ given by (5.1) and erase the last $m - n$ columns of U and the bottom $m - n$ rows of D (consisting of zeros). For $m < n$, the construction is similar. \square



5.8 Rank-one expansion

We have the following expansion of A :

$$A = \sum_{i=1}^r \sigma_i u_i v_i^* \quad (5.5)$$

Each term of this sum is an $m \times n$ matrix of rank one (Exercise 5.1).

Proof. It is enough to observe that for every v_j , $1 \leq j \leq n$,

$$\left(\sum_{i=1}^r \sigma_i u_i v_i^* \right) v_j = \sigma_j u_j = A v_j$$

because $v_i^* v_j = \delta_{ij}$ (the Kronecker delta symbol defined in Sect. 1.18). \square

5.9 Useful relations - II

Recall the Frobenius norm of a matrix (Section 1.6). We have

$$\|A\|_F^2 = (\text{Exercise 2.2}) = \|D\|_F^2 = \sigma_1^2 + \cdots + \sigma_r^2.$$

Comparing this with Section 5.5 gives

$$\|A\|_2 \leq \|A\|_F, \quad (5.6)$$

The value of $\|A\|_F^2$ can be interpreted as the *mass* (or *energy*) of the matrix A . The mass is conserved under multiplication by unitary matrices (Exercise 2.2), and the SVD pulls all the mass of a matrix onto its diagonal.

For any $1 \leq k \leq r$, let A_k denote the partial k -sum of (5.5):

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^*. \quad (5.7)$$

It is easy to see that $A_k v_i = \sigma_i u_i$ for $i = 1, \dots, k$ and $A_k v_i = 0$ for $i > k$, therefore $\text{rank } A_k = k$.

Comparing (5.7) with (5.5) we see that A_k has singular values $\sigma_1, \dots, \sigma_k$ and singular vectors u_1, \dots, u_k and v_1, \dots, v_k . Thus it agrees with A on the k largest singular values and the corresponding singular vectors.

Also, for any unit vectors u and v we have $\|uv^*\|_F = 1$ (Exercise 5.1), hence $\|\sigma_i u_i v_i^*\|_F^2 = \sigma_i^2$ and $\|A_k\|_F^2 = \sigma_1^2 + \cdots + \sigma_k^2$.

5.10 Low-rank approximation

For each $1 \leq k \leq r$ we have

$$\sigma_{k+1} = \|A - A_k\|_2 = \inf_{\substack{B \in \mathbb{C}^{m \times n} \\ \text{rank } B \leq k}} \|A - B\|_2 \quad (5.8)$$

(for $k = r$, we assume here that $\sigma_{r+1} = 0$).

Thus, A_k is the best approximation to A by matrices of rank $\leq k$.

Proof. Note that

$$A - A_k = \sum_{i=k+1}^r \sigma_i u_i v_i^* \quad (5.9)$$

hence the singular values of $A - A_k$ are $\sigma_{k+1}, \dots, \sigma_r$, of which σ_{k+1} is the largest. Thus $\|A - A_k\|_2 = \sigma_{k+1}$, according to Sect. 5.5.

Now suppose that there is some matrix B with $\text{rank } B \leq k$ such that $\|A - B\|_2 < \sigma_{k+1}$. For each nonzero vector $v \in \text{Ker } B$ we have

$$\|Av\|_2 = \|(A - B)v\|_2 \leq \|A - B\|_2 \|v\|_2 < \sigma_{k+1} \|v\|_2. \quad (5.10)$$

On the other hand, let us consider the subspace $L = \text{span}\{v_1, \dots, v_{k+1}\}$. For any nonzero vector $v \in L$ we have

$$v = c_1 v_1 + \dots + c_{k+1} v_{k+1}.$$

Due to Section 5.4

$$Av_2 = c_1 \sigma_1 u_1 + \dots + c_{k+1} \sigma_{k+1} u_{k+1},$$

therefore by (1.21)

$$\|Av\|_2^2 = (\sigma_1^2 |c_1|^2 + \dots + \sigma_{k+1}^2 |c_{k+1}|^2) \geq \sigma_{k+1}^2 \|v\|_2^2. \quad (5.11)$$

Note that $\dim L = (k + 1)$. On the other hand, due to (0.1) we have

$$\dim(\text{Ker } B) = n - \text{rank } B \geq n - k.$$

Since $(k + 1) + (n - k) > n$, the subspaces L and $\text{Ker } B$ must have a common nonzero vector v . That vector must satisfy both (5.10) and (5.11), which is impossible \Rightarrow a contradiction. \square

5.11 Distance to the nearest singular matrix

Let $A \in \mathbb{C}^{n \times n}$ be a nonsingular square matrix. Then

$$\min \{ \|A - A_s\|_2 : A_s \text{ is singular} \} = \sigma_n,$$

i.e., the smallest singular value σ_n of A is the “distance” from A to the “nearest” singular matrix.

Proof. Recall: A_s is singular iff $\text{rank} A_s \leq n - 1$, then use (5.8). \square

5.12 Small perturbations of matrices

We say that a matrix $E \in \mathbb{C}^{m \times n}$ is *small* if all its components are small. Equivalently, its norm $\|E\|_2$ is small. To see the equivalence, note that if $|e_{ij}| \leq \varepsilon$ for all i, j , then $\|E\|_2 \leq \|E\|_F \leq \varepsilon\sqrt{mn}$. On the other hand, if $\|E\|_2 \leq \varepsilon$, then $|e_{ij}| \leq \|E\mathbf{e}_j\|_2 \leq \|E\|_2 \leq \varepsilon$ for any i, j (here \mathbf{e}_j denotes the j th canonical basis vector).

For a matrix A and a small matrix E , we call $A + E$ a *small perturbation* of A .

5.13 Rank with tolerance ε

The *rank* of $A \in \mathbb{C}^{m \times n}$ with *tolerance* $\varepsilon > 0$ (also called *numerical rank*) is defined by

$$\text{rank}(A, \varepsilon) = \min_{\|E\|_2 \leq \varepsilon} \text{rank}(A + E).$$

Note: $\text{rank}(A, \varepsilon) \leq \text{rank} A$. The rank with tolerance ε is the minimum rank of A under small perturbations by matrices that have 2-norm $\leq \varepsilon$.

If A has full rank, i.e., $\text{rank} A = p = \min\{m, n\}$, but $\text{rank}(A, \varepsilon) < p$ for a small ε , then A is “nearly rank deficient”.

5.14 Computation of the numerical rank

The rank of A with tolerance ε equals the number of singular values of A (counted with multiplicity) that are *greater* than ε :

$$\text{rank}(A, \varepsilon) = \{k : \sigma_k > \varepsilon, \sigma_{k+1} \leq \varepsilon\}.$$

We also note that for each $\varepsilon > 0$

$$\text{rank}(A^*, \varepsilon) = \text{rank}(A, \varepsilon),$$

because A and A^* have the same singular values (Section 5.5).

The rest of this chapter uses certain topological terms (open and dense sets, metric). For students who are not familiar with these terms, explanations should be given, with simple examples.

5.15 Metric for matrices

We can measure the “distance” between matrices $A, B \in \mathbb{C}^{m \times n}$ as follows:

$$\text{dist}(A, B) = \|A - B\|_2$$

Then the space of matrices $\mathbb{C}^{m \times n}$ becomes a *metric space*. As a result, topological notions, like open sets, dense sets, etc., apply.

5.16 Topological properties of full rank matrices

Full rank matrices make an open and dense subset of $\mathbb{C}^{m \times n}$.

Openness means that for any full rank matrix A there is an $\varepsilon > 0$ such that all perturbations $A + E$ by small matrices E with norm $\|E\|_2 < \varepsilon$ have full rank. In plain words, any full rank matrix A is “surrounded” by full rank matrices.

Denseness means that for any rank-deficient matrix A and any $\varepsilon > 0$ there exists a small matrix E with norm $\|E\|_2 < \varepsilon$ such that $A + E$ has full rank. In other words, full rank matrices are scattered everywhere, they are in “every corner” of the space $\mathbb{C}^{m \times n}$. Yet in other words, every matrix is “arbitrarily close” to full rank matrices.

Proof. Openness follows from Section 5.14. Indeed, for any full rank matrix A we have $\sigma_p > 0$, where $p = \min\{m, n\}$. Hence every matrix B such that $\|A - B\|_2 < \sigma_p$ also has full rank.

To prove denseness, let A be a rank deficient matrix and let $A = UDV^*$ be its SVD. For any $\varepsilon > 0$, we construct $D_\varepsilon = \varepsilon I$ and $E = UD_\varepsilon V^*$. Then $\|E\|_2 = \|D_\varepsilon\|_2 = \varepsilon$ and

$$\text{rank}(A + E) = \text{rank}(U(D + D_\varepsilon)V^*) = \text{rank}(D + D_\varepsilon) = \min\{m, n\}$$

so $A + E$ has full rank. □

5.17 Topological property of diagonalizable matrices

Diagonalizable matrices make a dense subset of $\mathbb{C}^{n \times n}$.

Proof. See Exercise 6.3.

Exercise 5.1. Let $x \in \mathbb{C}^n$ and $y \in \mathbb{C}^m$. Consider the $m \times n$ matrix defined by $A = yx^*$.

- (a) Show that $\text{rank } A = 1$.
- (b) Show that $\|A\|_2 = \|x\|_2 \|y\|_2$.
- (c) Show that $\|A\|_F = \|x\|_2 \|y\|_2$ (which is the same as $\|x\|_F \|y\|_F$).

Exercise 5.2. (JPE, September 1996) Compute the singular values of

$$A = \begin{pmatrix} 0 & -1.6 & 0.6 \\ 0 & 1.2 & 0.8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Exercise 5.3. (JPE, May 2003) Determine the singular value decomposition for the matrix

$$A = \begin{pmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{pmatrix}$$

Exercise 5.4. Find the numerical rank with tolerance 0.9 of the matrix

$$A = \begin{pmatrix} 3 & 2 \\ -4 & -5 \end{pmatrix}$$

Exercise 5.5. Let $Q \in \mathbb{C}^{n \times n}$ be unitary. Find all singular values of Q .

Exercise 5.6. Show that if two matrices $A, B \in \mathbb{C}^{n \times n}$ are unitary equivalent, then they have the same singular values. Is the converse true? (Prove or give a counterexample.)

Schur Decomposition

Recall that every complex matrix $A \in \mathbb{C}^{n \times n}$ is similar to a Jordan matrix. In other words, for every linear operator $\mathbb{C}^n \rightarrow \mathbb{C}^n$ there exists a basis in which the operator is represented by a Jordan matrix. How can we represent linear operators if we use orthonormal bases only? In other words, to what extent one can simplify a complex matrix by using unitary equivalence (instead of similarity)?

6.1 Schur decomposition

Any matrix $A \in \mathbb{C}^{n \times n}$ is unitary equivalent to an upper triangular matrix T . That is, there exists a unitary matrix Q such that

$$A = QTQ^*.$$

Note that the diagonal of T consists of the eigenvalues of A . Moreover, one can find Q and T in such a way that the eigenvalues of A appear in any given order on the diagonal of T .

Proof. We use induction on n . For $n = 1$, we just write $A = I \cdot A \cdot I^*$ and note that I is unitary and $A = [a_{11}]$ is upper triangular.

Now let $n > 1$. Let λ be an eigenvalue of A and let x be a unit eigenvector corresponding to λ . Let Q_1 be a unitary matrix whose first column is x . Such a matrix exists, because there is an ONB in \mathbb{C}^n whose first vector is x , according to Section 1.26; then Q_1 can be constructed as a matrix whose columns are the vectors of that ONB.

Note that $Q_1 e_1 = x$, hence $Q_1^* x = e_1$. Therefore

$$Q_1^* A Q_1 e_1 = Q_1^* A x = \lambda Q_1^* x = \lambda e_1.$$

In other words, e_1 is an eigenvector of the matrix $Q_1^* A Q_1$ corresponding to the eigenvalue λ . Now the above formula implies

$$Q_1^* A Q_1 = \begin{bmatrix} \lambda & w^* \\ 0 & B \end{bmatrix}$$

with some $w \in \mathbb{C}^{n-1}$ and $B \in \mathbb{C}^{(n-1) \times (n-1)}$.

By our inductive assumption, the smaller matrix B has a Schur decomposition, i.e., $\hat{Q}^* B \hat{Q} = \hat{T}$ for a unitary matrix $\hat{Q} \in \mathbb{C}^{(n-1) \times (n-1)}$ and an upper triangular matrix \hat{T} . By Section 2.11, the enlarged $n \times n$ matrix $\begin{bmatrix} 1 & 0 \\ 0 & \hat{Q} \end{bmatrix}$ is also unitary. Thus

$$Q = Q_1 \begin{bmatrix} 1 & 0 \\ 0 & \hat{Q} \end{bmatrix}$$

is a unitary matrix, too (Section 2.9). Now one can verify directly that

$$\begin{aligned} Q^* A Q &= \begin{bmatrix} 1 & 0 \\ 0 & \hat{Q}^* \end{bmatrix} \begin{bmatrix} \lambda & w^* \\ 0 & B \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \hat{Q} \end{bmatrix} \\ &= \begin{bmatrix} \lambda & w^* \hat{Q} \\ 0 & \hat{Q}^* B \hat{Q} \end{bmatrix} = \begin{bmatrix} \lambda & w^* \hat{Q} \\ 0 & \hat{T} \end{bmatrix}. \end{aligned}$$

The last matrix is an $n \times n$ upper triangular matrix, as required. \square

6.2 Normal matrices

A matrix $A \in \mathbb{C}^{n \times n}$ is said to be *normal* if $AA^* = A^*A$.

One can easily check that unitary and Hermitian matrices are normal.

6.3 Normal matrices under unitary equivalence

If A is normal and Q unitary, then $B = Q^* A Q$ is normal. In other words, the class of normal matrices is closed under unitary equivalence.

Proof. Note that $B^* = Q^* A^* Q$ and $BB^* = Q^* A A^* Q = Q^* A^* A Q = B^* B$. \square

6.4 Lemma

If A is normal and upper triangular, then A is diagonal.

Proof. We use induction on n . For $n = 1$ the lemma is trivial. Assume that $n > 1$. Let us compute the top left element of the matrix $AA^* = A^*A$. On the one hand, it is

$$\sum_{i=1}^n a_{1i} \bar{a}_{1i} = \sum_{i=1}^n |a_{1i}|^2.$$

On the other hand, it is just $|a_{11}|^2$. Hence, $a_{12} = \cdots = a_{1n} = 0$, i.e.,

$$A = \begin{bmatrix} a_{11} & 0 \\ 0 & B \end{bmatrix}$$

One can easily check that $AA^* = A^*A$ implies $BB^* = B^*B$, i.e., B is a normal $(n-1) \times (n-1)$ matrix. Since A is upper triangular, so is B . By our inductive assumption, B is diagonal, hence A is diagonal, too. \square

Note: Any diagonal matrix is normal, because diagonal matrices commute.

6.5 Theorem

A matrix $A \in \mathbb{C}^{n \times n}$ is normal if and only if it is unitary equivalent to a diagonal matrix. In that case the Schur decomposition takes form

$$A = QDQ^* \tag{6.1}$$

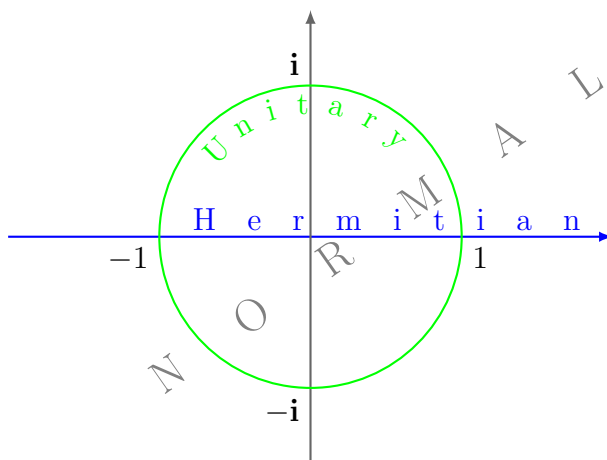
where D is a diagonal matrix. Whenever the Schur decomposition takes form (6.1), the columns of Q (Schur vectors) become eigenvectors of A .

Proof. This readily follows from Sections 6.1–6.4; see also Section 3.14. □

6.6 Remark

Three classes of complex matrices (unitary, Hermitian, and normal) have the same property: they are unitary equivalent to a diagonal matrix (in other words, they admit an ONB consisting of eigenvectors).

The difference between these classes lies in restrictions on the eigenvalues: unitary matrices have eigenvalues on the unit circle ($|\lambda| = 1$), Hermitian matrices have real eigenvalues ($\lambda \in \mathbb{R}$), and now normal matrices have arbitrary complex eigenvalues.



6.7 Real Schur Decomposition

If $A \in \mathbb{R}^{n \times n}$, then there exists an orthogonal matrix Q such that

$$Q^T A Q = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ 0 & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{mm} \end{bmatrix}$$

where each diagonal block R_{ii} is either a 1×1 matrix or a 2×2 matrix.

Proof. We use the induction on n . Our induction will be somewhat unusual – its increment will be either 1 or 2, i.e., our proof of the theorem for any given n will be based on its validity for either $n - 1$ or $n - 2$ (depending on the matrix A). Thus the basis for our induction must consist of *two* smallest values of n , i.e., $n = 1$ and $n = 2$.

For $n = 1$ and $n = 2$, we just write $A = I \cdot A \cdot I^*$ and note that I is unitary and A is either a single 1×1 block (if $n = 1$) or a single 2×2 block (if $n = 2$).

Now let $n > 2$. If the matrix A has a real eigenvalue, then we can reduce the dimension by one just like in the proof in Section 6.1 and then use our inductive assumption of the validity of the theorem for $n - 1$.

If A has no real eigenvalues, then by Lemma 2.16 there is a two-dimensional subspace $W \subset \mathbb{R}^n$ invariant under A . Let $\{x_1, x_2\}$ be an ONB of W (one exists due to Sect. 1.26). The invariance of W under A implies that $Ax_1 \in W$ and $Ax_2 \in W$, hence

$$\begin{aligned} Ax_1 &= r_{11} x_1 + r_{21} x_2 \\ Ax_2 &= r_{12} x_1 + r_{22} x_2 \end{aligned}$$

with some $r_{ij} \in \mathbb{R}$. Due to Sect. 1.26, the ONB $\{x_1, x_2\}$ in W can be extended to an ONB $\{x_1, \dots, x_n\}$ in \mathbb{R}^n . Let \tilde{Q} denote the orthogonal matrix with columns x_1, \dots, x_n . Observe that $\tilde{Q}e_1 = x_1$ and $\tilde{Q}e_2 = x_2$, hence $\tilde{Q}^T x_1 = e_1$ and $\tilde{Q}^T x_2 = e_2$. Therefore,

$$\tilde{Q}^T A \tilde{Q} e_1 = \tilde{Q}^T A x_1 = \tilde{Q}^T (r_{11} x_1 + r_{21} x_2) = r_{11} e_1 + r_{21} e_2$$

and similarly

$$\tilde{Q}^T A \tilde{Q} e_2 = \tilde{Q}^T A x_2 = \tilde{Q}^T (r_{12} x_1 + r_{22} x_2) = r_{12} e_1 + r_{22} e_2$$

Thus,

$$\tilde{Q}^T A \tilde{Q} = \begin{bmatrix} R_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{bmatrix}$$

where

$$R_{11} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}$$

\tilde{R}_{12} is some $2 \times (n - 2)$ matrix and \tilde{R}_{22} is some $(n - 2) \times (n - 2)$ matrix. Now we can apply our inductive assumption to the $(n - 2) \times (n - 2)$ matrix \tilde{R}_{22} and finish the proof in the same way as the proof in Section 6.1. Note that we need the validity of the theorem for $n - 2$, i.e., the increment of our induction is 2 in the above case. \square

Exercise 6.1. (combined from JPE, October 1990 and May 1997) Let $A \in \mathbb{C}^{n \times n}$ be a normal matrix.

- (a) Prove that $A - \lambda I$ is normal for any $\lambda \in \mathbb{C}$.
- (b) Prove that $\|Ax\| = \|A^*x\|$ for all x .
- (c) Prove that (λ, x) is an eigenpair of A if and only if $(\bar{\lambda}, x)$ is an eigenpair of A^* . (Hence, A and A^* have the same eigenvectors.)

Exercise 6.2. (JPE, September 2002) A matrix $A \in \mathbb{C}^{n \times n}$ is said to be *skew Hermitian* if $A^* = -A$.

- (a) Prove that if A is skew Hermitian and B is unitary equivalent to A , then B is also skew Hermitian.
- (b) Prove that the eigenvalues of a skew Hermitian matrix are purely imaginary, i.e. they satisfy $\bar{\lambda} = -\lambda$.
- (c) What special form does the Schur decomposition take for a skew Hermitian matrix A ?

Exercise 6.3. (JPE, September 1998). Show that diagonalizable complex matrices make a dense subset of $\mathbb{C}^{n \times n}$. That is, for any $A \in \mathbb{C}^{n \times n}$ and $\varepsilon > 0$ there is a diagonalizable $B \in \mathbb{C}^{n \times n}$ such that $\|A - B\|_2 < \varepsilon$.

Exercise 6.4 (Bonus). (JPE, May 1996). Let T be a linear operator on a finite dimensional complex inner product space V , and let T^* be the adjoint of T . Prove that $T = T^*$ if and only if $T^*T = T^2$.

LU Decomposition

First we review Gaussian elimination from Linear Algebra:

7.1 Gaussian elimination

Let $A \in \mathbb{C}^{n \times n}$ be a square matrix. We will modify it consecutively, in $n - 1$ steps, as described below, to obtain an upper triangular matrix. For the first step we denote $A^{(1)} = A$ and $a_{ij}^{(1)} = a_{ij}$ for all $1 \leq i, j \leq n$.

Step 1. Assume that $a_{11}^{(1)} \neq 0$. We define *multipliers*

$$m_{i1} = a_{i1}^{(1)} / a_{11}^{(1)} \quad \text{for } i = 2, \dots, n$$

Then we replace the i -th row $\tilde{a}_i^{(1)}$ of the matrix $A^{(1)}$ with $\tilde{a}_i^{(1)} - m_{i1}\tilde{a}_1^{(1)}$ for all $i = 2, \dots, n$, and obtain a new matrix

$$A^{(2)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}$$

where

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} \quad \text{for } 2 \leq i, j \leq n$$

Note that the first column of $A^{(2)}$ below the diagonal consists of zeros, i.e., $a_{i1}^{(2)} = 0$ for all $i = 2, \dots, n$.

Step 2. Assume that $a_{22}^{(2)} \neq 0$. We define *multipliers*

$$m_{i2} = a_{i2}^{(2)} / a_{22}^{(2)} \quad \text{for } i = 3, \dots, n$$

Then we replace the i -th row $\tilde{a}_i^{(2)}$ of the matrix $A^{(2)}$ with $\tilde{a}_i^{(2)} - m_{i2}\tilde{a}_2^{(2)}$ for

all $i = 3, \dots, n$, and obtain a new matrix

$$A^{(3)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} \end{bmatrix}$$

where

$$a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i1}a_{2j}^{(2)} \quad \text{for } 3 \leq i, j \leq n$$

Note that the first two columns of $A^{(3)}$ below the diagonal consists of zeros, i.e., $a_{ij}^{(3)} = 0$ for $j = 1, 2$ and all $i = j + 1, \dots, n$.

Then we continue this process. At each step we create zeros in one more column below the diagonal. In $n - 1$ steps all the columns below the main diagonal will consist of zeros, hence the resulting matrix A^n will be upper triangular, and we will denote it by U :

$$A^{(n)} = U = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} \end{bmatrix}$$

Remember that the above procedure requires $a_{kk}^{(k)} \neq 0$ for all $k = 1, \dots, n - 1$. These numbers are called *pivots*.

The procedure has to stop prematurely if (and only if) one of the pivots happens to be zero. In that case we say that the Gaussian elimination fails.

Note that

$$a_{kk}^{(k)} \neq 0 \quad \text{for all } k = 1, \dots, n - 1$$

is equivalent to

$$a_{11}^{(1)} \cdots a_{kk}^{(k)} \neq 0 \quad \text{for all } k = 1, \dots, n - 1$$

This will be convenient for the criterion of failure below.

7.2 Principal minors

Let $A \in \mathbb{C}^{n \times n}$. For $1 \leq k \leq n - 1$, the k -th principal minor of A is the $k \times k$ matrix formed by the entries in the first k rows and the first k columns of A . In other words, A_k is the top left $k \times k$ block of A . We denote the k -th principal minor of A by A_k . We will also denote $A_n = A$.

7.3 Criterion of failure

Gaussian elimination fails if and only if

$$\det A_k = 0 \quad \text{for some } k = 1, \dots, n - 1.$$

This is because for each $k = 1, \dots, n$

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

Next we present the Gaussian elimination in matrix form. All our formulas can be verified by direct inspection, so we give them without proof.

7.4 Gauss matrices

For each $j = 1, \dots, n - 1$ we denote

$$m^{(j)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m_{j+1,j} \\ \vdots \\ m_{n,j} \end{bmatrix}$$

and then we define the so called *Gauss matrix* G_j by

$$G_j = I - m^{(j)} e_j^* = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 1 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & -m_{j+1,j} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & 1 & 0 \\ 0 & 0 & \cdots & -m_{n,j} & \cdots & 0 & 1 \end{bmatrix}.$$

7.5 Main matrix formulas

For each $k = 1, \dots, n - 1$ we have

$$A^{(k+1)} = G_k A^{(k)}$$

and therefore

$$U = A^{(n)} = G_{n-1} \cdots G_2 G_1 A.$$

This can be rewritten as

$$A = G_1^{-1} G_2^{-1} \cdots G_{n-1}^{-1} U.$$

Next let us denote

$$L_j = I + m^{(j)} e_j^* = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & m_{j+1,j} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & 1 & 0 \\ 0 & 0 & \cdots & m_{n,j} & \cdots & 0 & 1 \end{bmatrix}.$$

For each $j = 1, \dots, n - 1$ we have $L_j G_j = I$, hence $G_j^{-1} = L_j$, and thus

$$A = L_1 L_2 \cdots L_{n-1} U. \quad (7.1)$$

7.6 Unit lower/upper triangular matrices

A matrix L is said to be *unit lower triangular* if it is lower triangular and has ones on its main diagonal. Note that $\det L = 1$. The following slight extensions of the rules given in Section 0.8 can be verified directly:

- If L_1 and L_2 are both unit lower triangular matrices, then so is their product $L_1 L_2$ and so are their inverses L_1^{-1} and L_2^{-1} .

Unit lower
triangular
matrices:

$$\begin{bmatrix} 1 & & 0 \\ & \ddots & \\ & & 1 \end{bmatrix} \times \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ & & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & & 0 \\ & \ddots & \\ & & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ & & 1 \end{bmatrix}$$

In a similar way we can define *unit upper triangular* matrices. They obey similar rules.

7.7 Main matrix formulas (continued)

Note that L_1, \dots, L_{n-1} in (7.1) are unit lower triangular matrices, thus their product is a unit lower triangular matrix, which we denote by L . More precisely, we have

$$\begin{aligned} L &= L_1 L_2 \cdots L_{n-1} \\ &= (I + m^{(1)} e_1^*) (I + m^{(2)} e_2^*) \cdots (I + m^{(n-1)} e_{n-1}^*) \\ &= I + m^{(1)} e_1^* + m^{(2)} e_2^* + \cdots + m^{(n-1)} e_{n-1}^* \\ &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n-1,1} & m_{n-1,2} & \cdots & 1 & 0 \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} & 1 \end{bmatrix}. \end{aligned}$$

We summarize our matrix formulas in the following theorem:

7.8 Theorem (LU decomposition)

Let $A \in \mathbb{C}^{n \times n}$ have non-singular principal minors up to the order $n - 1$. Then there is a decomposition

$$A = LU$$

where L is a unit lower triangular matrix and U is an upper triangular matrix:

$$\begin{array}{c} \boxed{} \\ A \end{array} = \begin{array}{c} \boxed{\begin{array}{cc} 1 & 0 \\ & 1 & \\ & & \ddots & \\ & & & 1 \end{array}} \\ L \end{array} \times \begin{array}{c} \boxed{\begin{array}{c} \\ \\ \\ 0 \end{array}} \\ U \end{array}$$

Note that $\det L = 1$, hence

$$\det A = \det U = u_{11} \cdots u_{nn}.$$

If, in addition, A is non-singular, then the LU decomposition is unique.

Proof: The existence follows from our matrix formulas. To prove the uniqueness, we argue by way of contradiction. Let $A = \tilde{L}\tilde{U} = LU$ be two LU decompositions. Since A is non-singular, it follows that \tilde{U} is also non-singular, and hence

$$L^{-1}\tilde{L} = U\tilde{U}^{-1}.$$

By the rules of Section 7.6, the inverse L^{-1} is unit lower triangular, so the product $L^{-1}\tilde{L}$ is unit lower triangular. On the other hand, $U\tilde{U}^{-1}$ is upper triangular. The only matrix that is both unit lower triangular and upper triangular is the identity matrix I , hence

$$L^{-1}\tilde{L} = U\tilde{U}^{-1} = I.$$

This implies $\tilde{L} = L$ and $\tilde{U} = U$. □

7.9 Forward and backward substitutions

The LU decomposition can be used to solve a system of linear equations

$$Ax = b,$$

where $A \in \mathbb{C}^{n \times n}$ is a given nonsingular matrix, $b \in \mathbb{C}^n$ is a given vector, and $x \in \mathbb{C}^n$ is an unknown vector to be computed.

Suppose we have an LU decomposition $A = LU$, where L is lower triangular and U is upper triangular. Then we can find x in two steps.

$$\begin{array}{c} \boxed{\begin{array}{c} \text{0} \\ \diagdown \\ \end{array}} \\ L \end{array} \times \underbrace{\begin{array}{c} \boxed{\begin{array}{c} \text{0} \\ \diagdown \\ \end{array}} \\ U \end{array}}_y \times \begin{array}{c} \boxed{} \\ x \end{array} = \begin{array}{c} \boxed{} \\ b \end{array}$$

Step 1. Denote $Ux = y$ and solve the lower triangular system $Ly = b$ for y via “forward substitution” (computing y_1, \dots, y_n subsequently):

$$\begin{aligned} \ell_{11}y_1 = b_1 & \Rightarrow y_1 = b_1/\ell_{11} \\ \ell_{21}y_1 + \ell_{22}y_2 = b_2 & \Rightarrow y_2 = (b_2 - \ell_{21}y_1)/\ell_{22} \\ \ell_{31}y_1 + \ell_{32}y_2 + \ell_{33}y_3 = b_3 & \Rightarrow y_3 = (b_3 - \ell_{31}y_1 - \ell_{32}y_2)/\ell_{33} \\ & \vdots \\ \ell_{n1}y_1 + \dots + \ell_{nn}y_n = b_n & \Rightarrow y_n = (b_n - \ell_{n1}y_1 - \dots - \ell_{n(n-1)}y_{n-1})/\ell_{nn} \end{aligned}$$

Note: if L is *unit* lower triangular, then $\ell_{11} = \dots = \ell_{nn} = 1$, and therefore the divisions can be avoided.

Step 2. Solve the system $Ux = y$ for x via “backward substitution” (computing x_n, \dots, x_1 subsequently):

$$\begin{aligned} u_{nn}x_n = y_n & \Rightarrow x_n = y_n/u_{nn} \\ u_{(n-1)(n-1)}x_{n-1} + u_{(n-1)n}x_n = y_{n-1} & \Rightarrow x_{n-1} = (y_{n-1} - u_{(n-1)n}x_n)/u_{(n-1)(n-1)} \\ & \vdots \\ u_{11}x_1 + \dots + u_{1n}x_n = y_1 & \Rightarrow x_1 = (y_1 - u_{12}x_2 - \dots - u_{1n}x_n)/u_{11} \end{aligned}$$

7.10 Cost of computation

The cost of computation is measured in “flops”, where a flop (*f*loating *o*perat*o*n) is an arithmetic operation (addition, subtraction, multiplication, division, or a root extraction).

Let us estimate the cost of the LU decomposition. The cost of computation of $A^{(2)}$ is $n - 1$ divisions to compute the multipliers and then $2n(n - 1)$ flops to update the rows ($n - 1$ rows with $2n$ flops per row), i.e. total of approximately $2n^2$ flops. The computation of $A^{(3)}$ then takes $2(n - 1)^2$ flops, and so on.

Thus the total computational cost for the LU factorization is

$$2(n^2 + (n - 1)^2 + \cdots + 1^2) = \frac{2n(n + 1)(2n + 1)}{6} \approx \frac{2n^3}{3} \quad \text{flops}$$

If one solves a system of linear equations $Ax = b$, then the LU decomposition is followed by solving two triangular systems (Section 7.9). The cost to solve one triangular system is about n^2 flops. So there is an additional cost of $2n^2$ flops, which is negligible compared to $2n^3/3$. Hence, the total cost is still $\approx 2n^3/3$.

Note that the LU decomposition takes most of the computations required for solving a system $Ax = b$. Thus, this method is particularly well suited to very common situations in which one is solving systems $Ax = b$ for more than one vector b , but with the same matrix A . In that case the LU decomposition is done just once, and then each additional b will require $\approx 2n^2$ flops.

7.11 Computation of A^{-1}

Assume that $A \in \mathbb{C}^{n \times n}$ is non-singular and has non-singular principal minors up to the order $n - 1$, so that the Gaussian elimination works. One can find the matrix $X = A^{-1}$ by solving the system $AX = I$ for $X \in \mathbb{C}^{n \times n}$. This amounts to solving n systems of linear equations $Ax_k = e_k$, for $k = 1, \dots, n$, where x_k stands for the k -th column of the matrix X . The computational cost of this procedure is

$$\frac{2n^3}{3} + n \times 2n^2 = \frac{8n^3}{3} \quad \text{flops}$$

As a matter of fact, this is the fastest way of computing the inverse A^{-1} .

7.12 “Manual” solution of systems of linear equations

One needs to remember that the LU decomposition (with the subsequent forward and backward substitutions) is just a convenient matrix representation of the standard and more familiar Gaussian elimination. These two procedures are mathematically equivalent. When one solves a system of linear equations $Ax = b$ practically (on paper), one should definitely apply the Gaussian elimination to both A and b , rather than forming the L and U matrices.

7.13 Example

Suppose one needs to solve the following system by LU decomposition:

$$\begin{bmatrix} 0.01 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Then one multiplies the first row by 100 and subtracts it from the second row. This gives

$$\begin{bmatrix} 0.01 & 2 \\ 0 & -197 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ -196 \end{bmatrix}$$

Then one solves the second equation for y and finds $y = \frac{196}{197}$. Lastly, one substitutes this value for y into the first equation and finds $x = \frac{200}{197}$.

7.14 Warnings

It is a common practice to solve systems of linear equations $Ax = b$ with a nonsingular matrix A via Gaussian elimination (or, equivalently, LU decomposition). However, this method requires that the matrix A satisfies certain conditions (Section 7.3).

It is easy to find nonsingular matrices for which Gaussian elimination does *not* work and LU decomposition does *not* exist. For example, let $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Then the Gaussian elimination fails instantly, because the very first pivot is zero. Furthermore, for this matrix A there is no LU decomposition $A = LU$, even if we only require that L be lower triangular (not necessarily unit lower triangular); see Exercise 7.2.

In addition, when a pivot is not exactly zero but close to zero, then things can go wrong, too: multipliers become very large, and numerical computations (in the sense specified later in Chapter 11) tend to become inaccurate. One should avoid such situations by using an appropriate *pivoting strategy*. We will describe these strategies, briefly, and illustrate them by example.

7.15 Partial pivoting

The idea is to avoid zero or small pivots by interchanging rows. At every step of Gaussian elimination one looks for the largest (in absolute value) element in the pivot column (at or below the main diagonal). Then the row containing the largest element is interchanged with the current row. Now the largest element is on the main diagonal. After that the usual elimination step is performed.

Partial pivoting ensures that all the multipliers (i.e., the elements of the matrix L) have absolute values less than or equal to one.

7.16 Example

We apply the partial pivoting strategy to Example 7.13. In the first column, the larger element is 1, hence we need to interchange the rows to put that element on the diagonal:

$$\begin{bmatrix} 1 & 3 \\ 0.01 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}.$$

Now we multiply the first row by 0.01 and subtract it from the second row. This gives

$$\begin{bmatrix} 1 & 3 \\ 0 & 1.97 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 1.96 \end{bmatrix}.$$

Then we solve the second equation for y and find $y = \frac{196}{197}$. Lastly, we substitute this value for y into the first equation and find $x = \frac{200}{197}$.

7.17 Complete pivoting

The method of *complete pivoting* involves both row and column interchanges to make use of the largest pivot available. This method provides additional insurance against buildup of computational errors.

7.18 Example

We apply the complete pivoting strategy to Example 7.13. The largest element in the whole matrix is 3, so we need to move it to the top left position by interchanging rows and columns:

$$\begin{bmatrix} 3 & 1 \\ 2 & 0.01 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}.$$

Note that interchanging columns requires permutation of the unknowns x and y . Next we multiply the first row by $2/3$ and subtract it from the second row. This gives

$$\begin{bmatrix} 3 & 1 \\ 0 & -\frac{197}{300} \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} 4 \\ -\frac{2}{3} \end{bmatrix}.$$

Then we solve the second equation for x and find $x = \frac{200}{197}$. Lastly, we substitute this value for x into the first equation and find $y = \frac{196}{197}$.

7.19 Diagonally dominant matrices

A matrix $A \in \mathbb{C}^{n \times n}$ such that

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

for all i is said to be *strictly row diagonally dominant*. If

$$|a_{jj}| > \sum_{i \neq j} |a_{ij}|$$

for all j the matrix is said to be *strictly column diagonally dominant*.

One can check that if a matrix A is strictly row (or column) diagonally dominant, then $\det A \neq 0$ (the proof can be omitted for now; this fact readily follows from Gershgorin Theorem that will be given in Chapter 15). Since all the principal minors of A are also strictly row (or column) diagonally dominant, we have $\det A_k \neq 0$ for all $1 \leq k < n$. Therefore no zero pivots will be encountered during Gaussian elimination.

With some extra effort one can show that if A is strictly column diagonally dominant, then all the multipliers (i.e., the elements of L) have absolute values ≤ 1 .

Exercise 7.1. Find a nonzero matrix $A \in \mathbb{R}^{2 \times 2}$ that admits at least two LU decomposition, i.e. $A = L_1 U_1 = L_2 U_2$, where L_1 and L_2 are two *distinct* unit lower triangular matrices and U_1 and U_2 are two *distinct* upper triangular matrices.

Exercise 7.2. Show that the matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ admits no LU decomposition, even if we only require that L be lower triangular (not necessarily unit lower triangular).

Exercise 7.3. The spectral radius of a matrix $A \in \mathbb{C}^{n \times n}$ is defined by

$$\rho_A = \max\{|\lambda| : \lambda \text{ eigenvalue of } A\}.$$

- (a) Show that $\rho_A \leq \|A\|_2$.
- (b) Give an example of a 2×2 matrix A such that $\rho_A < 1$ but $\|A\|_2 > 100$.
- (c) Show that if

$$\lim_{n \rightarrow \infty} \|A^n\|_2 = 0,$$

then $\rho_A < 1$.

Exercise 7.4 (Bonus). In the notation of the previous problem, show that if $\rho_A < 1$, then

$$\lim_{n \rightarrow \infty} \|A^n\|_2 = 0.$$

Hint: use Jordan decomposition.

Cholesky Factorization

8.1 Theorem (LDM* Decomposition)

Assume that $A \in \mathbb{C}^{n \times n}$ is non-singular and has non-singular principal minors, i.e., $\det A_k \neq 0$ for all $k = 1, \dots, n$. Then there are unique matrices L, D, M such that L, M are unit lower triangular, D is diagonal, and

$$A = LDM^* \tag{8.1}$$

$$\boxed{A} = \boxed{L} \times \boxed{D} \times \boxed{M^*}$$

$\begin{matrix} \square & \begin{matrix} 1 & 0 \\ & 1 & \\ & & \ddots \\ & & & 1 \end{matrix} & \times & \begin{matrix} & & & 0 \\ & & & & \\ & & & & \\ 0 & & & & \end{matrix} & \times & \begin{matrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{matrix} \\ A & L & & D & & M^* \end{matrix}$

Proof. According to Theorem 7.8, there is an LU decomposition $A = LU$. Let u_{11}, \dots, u_{nn} denote the diagonal components of U . Define diagonal matrix $D = \text{diag}\{u_{11}, \dots, u_{nn}\}$. Since A is non-singular, D is invertible, and $D^{-1} = \text{diag}\{u_{11}^{-1}, \dots, u_{nn}^{-1}\}$. One can easily check that $M^* = D^{-1}U$ is a unit upper triangular matrix, hence (8.1) exists.

To prove the uniqueness, suppose $A = LDM^* = L_1D_1M_1^*$ are two different LDM* decompositions. Note that all the matrices here are invertible, because A is non-singular. By the uniqueness of the LU decomposition (Theorem 7.8), we have $L = L_1$, hence, $DM^* = D_1M_1^*$ and $D_1^{-1}D = M_1^*(M^*)^{-1}$. Since both M^* and M_1^* are unit upper triangular, so is $M_1^*(M^*)^{-1}$ (by Sect. 7.6). On the other hand, $D_1^{-1}D$ is diagonal. The only matrix that is unit lower triangular and diagonal is the identity matrix. Thus $M_1^*(M^*)^{-1} = I$ and $D_1^{-1}D = I$, which implies $M^* = M_1^*$ and $D = D_1$. \square

8.2 Corollary (LDL* Decomposition)

Suppose $A \in \mathbb{C}^{n \times n}$ satisfies the assumptions of Theorem 8.1 and is Hermitian. Then there exist a unique unit lower triangular matrix L and a unique diagonal matrix D such that

$$A = LDL^*$$

Moreover, the components of D are *real* numbers.

Proof. By Theorem 8.1, $A = LDM^*$. Now $A = A^* = MD^*L^*$, and by the uniqueness of the LDM* decomposition we have $L = M$ and $D = D^*$. The latter implies that D is a real matrix. \square

8.3 Sylvester's Theorem

Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix. Then

$$A \text{ is positive definite} \iff \det A_k > 0 \quad (\forall k = 1, \dots, n)$$

Proof. Let A be positive definite. By Section 4.15, $A = B^*B$ and $\det B \neq 0$. Hence

$$\det A = \det B \times \det B^* = \det B \times \overline{\det B} = |\det B|^2 > 0.$$

One can easily check that every principal minor A_k is also a Hermitian positive definite matrix, therefore by the same argument $\det A_k > 0$.

Now let $\det A_k > 0$ for all $k = 1, \dots, n$. By Corollary 8.2, $A = LDL^*$. Denote by L_k and D_k the k -th principal minors of L and D , respectively. One can easily check that $A_k = L_k D_k L_k^*$. Since L_k is *unit* lower triangular, we have $\det L_k = 1$, thus $\det D_k = \det A_k > 0$. Denote $D = \text{diag}\{d_1, \dots, d_n\}$, hence $D_k = \text{diag}\{d_1, \dots, d_k\}$. Now

$$\det D_k = d_1 \cdots d_k > 0 \quad \forall k = 1, \dots, n$$

implies that d_1, \dots, d_n are all real and positive. Lastly,

$$\langle Ax, x \rangle = \langle LDL^*x, x \rangle = \langle DL^*x, L^*x \rangle = \langle Dy, y \rangle > 0$$

where $y = L^*x$. Note that $y \neq 0$ whenever $x \neq 0$, because L^* is invertible. \square

8.4 Corollary

Let A be a positive definite matrix. Then $a_{ii} > 0$ for all $i = 1, \dots, n$. Furthermore, let $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, and let A' be the $k \times k$ matrix formed by the intersections of the rows and columns of A with numbers i_1, \dots, i_k . Then $\det A' > 0$.

Proof. Recall that A defines an operator $\mathbb{C}^n \rightarrow \mathbb{C}^n$. We can reorder the canonical basis vectors e_1, \dots, e_n in \mathbb{C}^n and get a new ONB $\{e_{i_1}, \dots, e_{i_k}, \dots\}$. In this new basis, the operator is represented by a matrix B such that $B_k = A'$, i.e., A' is a principal minor of a positive definite matrix. Then we can apply Sylvester's Theorem 8.3. \square

The above corollary suggests a quick way of proving that a given matrix A is *not* positive definite: just spot a non-positive diagonal component or a 2×2 submatrix (with two components on the diagonal of A) with a non-positive determinant, etc.

8.5 Theorem (Cholesky Factorization)

Let $A \in \mathbb{C}^{n \times n}$ be Hermitian and positive definite. Then there exists a unique lower triangular matrix G with real positive diagonal entries such that

$$A = GG^*$$

Example (zeros in matrices are not shown):

$$\begin{bmatrix} 4 & 2 & 8 \\ 2 & 10 & 7 \\ 8 & 6 & 21 \end{bmatrix} = \begin{bmatrix} 2 & & \\ 1 & 3 & \\ 4 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 1 & 4 \\ & 3 & 2 \\ & & 1 \end{bmatrix}$$

Proof. By Corollary 8.2, the matrix A has an LDL^* decomposition $A = LDL^*$. Denote $D = \text{diag}\{d_1, \dots, d_n\}$. As it was shown in the proof of Sylvester's Theorem 8.3, all d_i 's are real and positive numbers. Thus we can form another diagonal matrix with real positive diagonal entries as follows: $D_{1/2} = \text{diag}\{\sqrt{d_1}, \dots, \sqrt{d_n}\}$. Then $D = D^{1/2}D^{1/2}$ and setting $G = LD^{1/2}$ readily gives $A = GG^*$. One can easily check that the diagonal entries of G are $\sqrt{d_1}, \dots, \sqrt{d_n}$, so they are real positive numbers, as required.

To prove the uniqueness, suppose $A = GG^* = \tilde{G}\tilde{G}^*$ are two different Cholesky factorizations. Then $\tilde{G}^{-1}G = \tilde{G}^*(G^*)^{-1}$. This is the equality of a lower triangular matrix and an upper triangular one; hence both matrices are diagonal:

$$\tilde{G}^{-1}G = \tilde{G}^*(G^*)^{-1} = D' = \text{diag}\{d'_1, \dots, d'_n\}.$$

Since A is positive definite, it must be non-singular, hence all the matrices in our formulas must be invertible. Note that $\tilde{G} = G(D')^{-1}$, which implies

$$\tilde{g}_{ii} = g_{ii}/d'_i \quad \forall i = 1, \dots, n.$$

Similarly, $\tilde{G}^* = G^*D'$, which implies

$$\tilde{g}_{ii} = g_{ii}d'_i \quad \forall i = 1, \dots, n.$$

Multiplying the above two formulas gives $\tilde{g}_{ii}^2 = g_{ii}^2$, and since these are real positive numbers, we get $\tilde{g}_{ii} = g_{ii}$. Therefore $d'_i = 1$ and $D' = I$. This implies $\tilde{G} = G$. \square

8.6 Algorithm for Cholesky factorization

Here we outline the algorithm for computing the matrix $G = (g_{ij})$ from the matrix $A = (a_{ij})$. For simplicity, we will restrict our presentation to the **real** matrices $A \in \mathbb{R}^{n \times n}$. Note that G is lower triangular, i.e., $g_{ij} = 0$ for $i < j$, hence,

$$a_{ij} = \sum_{k=1}^{\min\{i,j\}} g_{ik}g_{jk}.$$

Setting $i = j = 1$ gives $a_{11} = g_{11}^2$, hence

$$g_{11} = \sqrt{a_{11}}.$$

Next, for $2 \leq i \leq n$ we have $a_{i1} = g_{i1}g_{11}$, hence

$$g_{i1} = a_{i1}/g_{11} \quad i = 2, \dots, n.$$

This gives the first column of G . Now, inductively, assume that we already have the first $j - 1$ columns of G . Then $a_{jj} = \sum_{k=1}^j g_{jk}^2$, hence

$$g_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} g_{jk}^2}.$$

Next, for $j + 1 \leq i \leq n$ we have $a_{ij} = \sum_{k=1}^j g_{ik}g_{jk}$, hence

$$g_{ij} = \frac{1}{g_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} g_{ik}g_{jk} \right).$$

8.7 Cost of computation

The computation of g_{ij} takes $\approx 2j$ flops for each $i = j, \dots, n$, so the total is

$$\sum_{j=1}^n 2j(n-j) \approx 2n \frac{n^2}{2} - 2 \frac{n^3}{3} = \frac{n^3}{3}$$

Recall that the LU decomposition takes about $2n^3/3$ flops, so the Cholesky factorization is nearly twice as fast. It is also more stable than the LU decomposition, in the sense precisely defined below, in Chapter 13.

8.8 Criterion of positive definiteness

The above algorithm can be used to verify that a given real symmetric matrix, A , is positive definite. Whenever all the square root extractions in Section 8.6 are possible and give non zero numbers, i.e. whenever

$$a_{11} > 0 \quad \text{and} \quad a_{jj} - \sum_{k=1}^{j-1} g_{jk}^2 > 0 \quad \forall j \geq 2$$

the matrix A is positive definite. If at least one of the above quantities is *not* positive, then A is not positive definite.

Exercise 8.1. (JPE, May 1994) Let $A \in \mathbb{R}^{n \times n}$ be given, symmetric and positive definite. Define $A_0 = A$, and consider the sequence of matrices defined by

$$A_k = G_k G_k^t \quad \text{and} \quad A_{k+1} = G_k^t G_k$$

where $A_k = G_k G_k^t$ is the Cholesky factorization for A_k . Prove that the A_k all have the same eigenvalues.

QR Decomposition

9.1 Gram-Schmidt orthogonalization (revisited)

Let $\{v_1, \dots, v_n\}$ be a linearly independent set of vectors in \mathbb{C}^m ($m \geq n$). The Gram-Schmidt orthogonalization described in Section 1.25 gives an orthonormal set of vectors $\{u_1, \dots, u_n\}$ in \mathbb{C}^m , and the equations (1.14)–(1.16) can be written as

$$\begin{aligned} v_1 &= r_{11}u_1, \\ v_2 &= r_{12}u_1 + r_{22}u_2, \\ v_3 &= r_{13}u_1 + r_{23}u_2 + r_{33}u_3, \\ &\dots \\ v_n &= r_{1n}u_1 + r_{2n}u_2 + \dots + r_{n-1,n}u_{n-1} + r_{nn}u_n \end{aligned} \tag{9.1}$$

where $r_{ip} = \langle v_p, u_i \rangle$ for $i < p \leq n$ and $r_{pp} = \|w_p\|$. Note that $r_{pp} > 0$.

Let us assemble a matrix $A \in \mathbb{C}^{m \times n}$ whose columns are the vectors v_1, \dots, v_n , a matrix \hat{Q} whose columns are the vectors u_1, \dots, u_n :

$$A = \left[\begin{array}{c|c|c|c} v_1 & v_2 & \cdots & v_n \end{array} \right] \quad \text{and} \quad \hat{Q} = \left[\begin{array}{c|c|c|c} u_1 & u_2 & \cdots & u_n \end{array} \right].$$

Let $\hat{R} \in \mathbb{C}^{n \times n}$ be an upper triangular matrix defined as follows:

$$\hat{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}.$$

Then the above relations (9.1) can be written in matrix form:

$$A = \hat{Q}\hat{R} \tag{9.2}$$

One can verify the equivalence of (9.1) and (9.2) by direct inspection.

9.2 Linearly dependent case

Let us relax our assumption that v_1, \dots, v_n are linearly independent. Thus let $\{v_1, \dots, v_n\}$ be an *arbitrary* set of vectors in \mathbb{C}^m . Then the Gram-Schmidt procedure 1.25 can be adjusted as follows.

For some $p \geq 1$ we now may have $v_p = 0$ or, more generally,

$$v_p \in \text{span}\{v_1, \dots, v_{p-1}\}.$$

This implies $v_p \in \text{span}\{u_1, \dots, u_{p-1}\}$, therefore $w_p = 0$, in the notation of Section 1.25. Now the vector $u_p = w_p/\|w_p\|$ cannot be determined.

Instead, we can choose u_p as an arbitrary unit vector orthogonal to u_1, \dots, u_{p-1} (we know that such vectors exist due to Sect. 1.26). After such a “loosely constrained” selection of u_p we continue the Gram-Schmidt orthogonalization as described in Section 1.25. Every time we encounter a linear dependence $v_p \in \text{span}\{v_1, \dots, v_{p-1}\}$, we select u_p “frivolously” as specified above. In that case, the respective equation

$$v_p = r_{1p}u_1 + \dots + r_{p-1,p}u_{p-1} + r_{pp}u_p$$

for v_p , as given in (9.1), will still hold, but with $r_{pp} = 0$ instead of $r_{pp} > 0$.

9.3 Extension of \hat{Q} to Q

The vectors u_1, \dots, u_n make an orthonormal set in \mathbb{C}^m that can be extended to an ONB $\{u_1, \dots, u_m\}$, according to Sect. 1.26. Now we can form a larger, square matrix

$$Q = \left[\begin{array}{c|c|c|c|c|c|c} u_1 & u_2 & \cdots & u_n & \cdots & & \\ \hline \end{array} \right] \in \mathbb{C}^{m \times m}$$

which is *unitary*, according to Sect. 2.4. By adding $m - n$ rows consisting of zeros to the bottom of the matrix \hat{R} we get a larger matrix

$$R = \left[\begin{array}{cccc} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{array} \right] \in \mathbb{C}^{m \times n}.$$

We still call R an “upper triangular matrix”. Now the relation (9.2) can be written as

$$A = QR. \tag{9.3}$$

The equivalence of (9.2) and (9.3) can be verified by direct inspection.

We summarize the previous constructions as follows:

9.4 Theorem (QR decomposition)

For any matrix $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ there exist a unitary matrix $Q \in \mathbb{C}^{m \times m}$ and an upper triangular matrix $R \in \mathbb{C}^{m \times n}$ such that

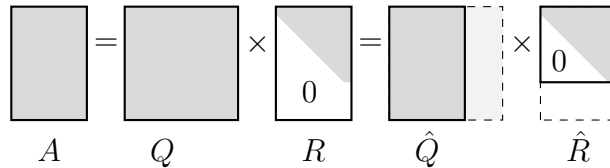
$$A = QR. \quad (9.4)$$

This is called *(full) QR decomposition* of A .

The first (left) n columns of the matrix Q form a smaller matrix $\hat{Q} \in \mathbb{C}^{m \times n}$ (whose columns are orthonormal vectors in \mathbb{C}^m) and the first (upper) n rows of the matrix R form a square upper triangular matrix \hat{R} so that

$$A = \hat{Q}\hat{R}. \quad (9.5)$$

This is called *reduced (“skinny”) QR decomposition* of A .



Proof. Given a matrix $A \in \mathbb{C}^{m \times n}$, we disassemble it into column vectors $v_1, \dots, v_n \in \mathbb{C}^m$, and then apply all the constructions described in Sections 9.1–9.3. \square

9.5 Real QR decomposition

Suppose $A \in \mathbb{R}^{m \times n}$ is a real matrix. Then all the constructions of Sections 9.1–9.3 can be done with real matrices only, and they produce real matrices Q and R . In particular, Q will be an orthogonal matrix.

9.6 Positivity of the diagonal of R

The QR decomposition can be constructed so that the diagonal components of R (\hat{R}) are real non-negative numbers:

$$r_{ii} \geq 0 \quad \forall i = 1, \dots, n.$$

If A has full rank (i.e., $\text{rank } A = n$), then the columns of A are linearly independent. In that case \hat{R} is invertible, and one can construct R (and \hat{R}) so that its diagonal components are real positive numbers:

$$r_{ii} > 0 \quad \forall i = 1, \dots, n.$$

9.7 Uniqueness of the QR decomposition

If A has full rank (i.e., $\text{rank } A = n$), then the matrix \hat{R} with real positive diagonal entries ($r_{ii} > 0$) is unique. The corresponding matrix \hat{Q} is unique, too.

Proof. Let $A = \hat{Q}\hat{R} = \hat{Q}_1\hat{R}_1$ be two different reduced QR decompositions with the specified properties. Then

$$A^*A = \hat{R}^*\hat{R}$$

because $\hat{Q}^*\hat{Q} = I$ is the identity matrix (this follows from the columns of \hat{Q} being orthonormal vectors). Similarly,

$$A^*A = \hat{R}_1^*\hat{R}_1.$$

Thus we get two Cholesky factorizations for the same matrix A^*A . But this matrix is positive definite (due to Section 4.18), hence its Cholesky factorization must be unique (cf. Section 8.5). This implies $\hat{R} = \hat{R}_1$. Lastly, $\hat{Q}_1 = \hat{Q}\hat{R}\hat{R}_1^{-1} = \hat{Q}$. \square

9.8 Cost of QR

In order to compute the reduced QR decomposition (9.5), one needs to apply Gram-Schmidt orthogonalization to the n columns of the matrix A and compute the n columns u_1, \dots, u_n of the matrix \hat{Q} . The vector u_p is found by

$$w_p = v_p - \sum_{i=1}^{p-1} \langle v_p, u_i \rangle u_i, \quad \text{and} \quad u_p = \frac{w_p}{\|w_p\|},$$

see Section 1.25. Here each scalar product $\langle v_p, u_i \rangle$ requires m multiplications and m additions, and then subtracting every term $\langle v_p, u_i \rangle u_i$ from v_p requires m multiplication and m subtractions, for each $i = 1, \dots, p$. The total is $4mp$ flops. The subsequent computation of $\|w_{p+1}\|$ and then u_{p+1} requires $3m$ flops, which is a relatively small number, and we ignore it. The total flop count is $\sum_{p=1}^n 4mp \approx 2mn^2$ flops, i.e., the QR takes

$2mn^2$ flops

9.9 Cost of SVD

There is no finite algorithms for the computation of SVD described in Chapter 5 (except for small matrices, see Remark 5.6). The reason will be discussed later in Chapter 15. In practice, the SVD is computed by special iterative algorithms. The computation of the reduced SVD requires approximately

$2mn^2 + 11n^3$ flops

9.10 Modified Gram-Schmidt orthogonalization

The algorithm from Section 1.25 is often called *classical* Gram-Schmidt orthogonalization, as opposed to the *modified* Gram-Schmidt orthogonalization we present next. Given a linearly independent set of vectors $\{v_1, \dots, v_n\}$ in \mathbb{C}^m , we denote $v_i^{(1)} = v_i$ for $i = 1, \dots, n$, then compute

$$u_1 = v_1^{(1)} / \|v_1^{(1)}\|,$$

and then modify all the remaining vectors by the rule

$$v_i^{(2)} = v_i^{(1)} - \langle v_i^{(1)}, u_1 \rangle u_1 \quad \text{for } 1 < i \leq n.$$

After that, inductively, for each $p \geq 2$ we compute

$$u_p = v_p^{(p)} / \|v_p^{(p)}\|,$$

and then modify all the remaining vectors by the rule

$$v_i^{(p+1)} = v_i^{(p)} - \langle v_i^{(p)}, u_p \rangle u_p \quad \text{for } p < i \leq n.$$

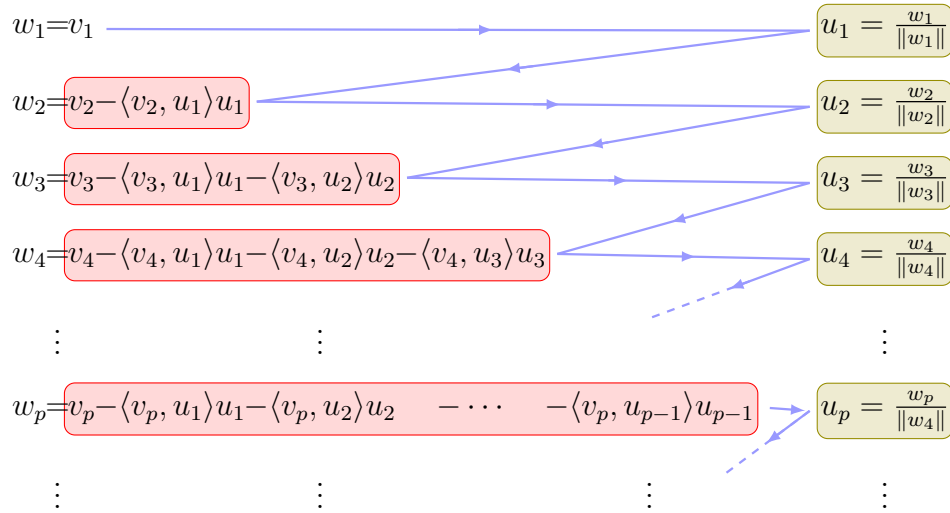
The modified and classical Gram-Schmidt methods produce the same orthonormal basis $\{u_1, \dots, u_n\}$ (i.e., these two methods are mathematically equivalent). They are based on a different logic, though.

The classical Gram-Schmidt computes u_p by making the current vector v_p orthogonal to all the previously constructed vectors u_1, \dots, u_{p-1} , without touching the remaining vectors v_{p+1}, \dots, v_n . The amount of work increases as p grows from 1 to n . This is a “lazy man schedule” - do as little as possible and leave the rest of the work “for later”.

The modified Gram-Schmidt computes u_p and makes all the remaining vectors v_{p+1}, \dots, v_n orthogonal to it. Once this is done, the remaining vectors will be in the orthogonal complement to the subspace $\text{span}\{u_1, \dots, u_p\}$ and there is no need to involve the previously constructed vectors anymore. The amount of work decreases as p grows from 1 to n . This is an “industrious man schedule” - do as much as possible *now* and reduce the workload.

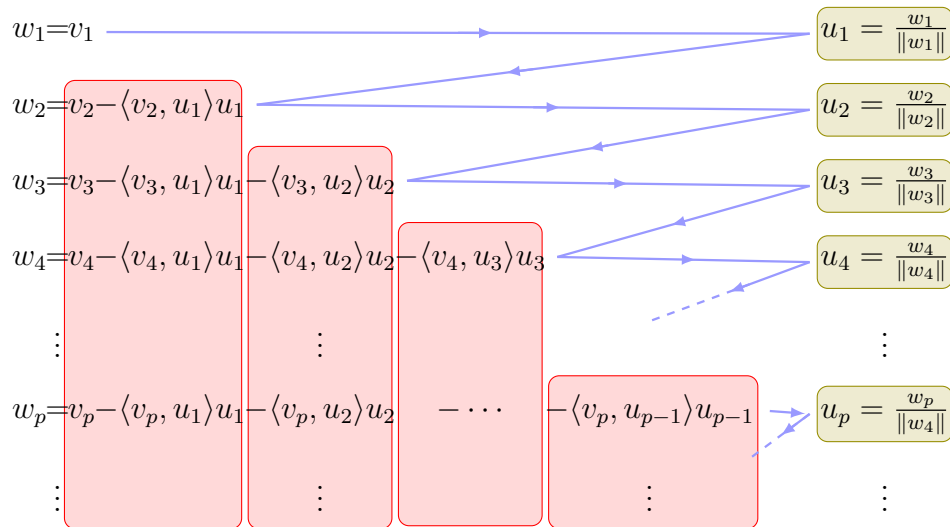
Overall, both methods require the same amount of flops. But the modified Gram-Schmidt has an important advantage that it gives more accurate numerical results in computer calculations; see programming assignment.

Classical Gram-Schmidt:



The amount of work increases at each step (the red rows grow longer)

Modified Gram-Schmidt:



The amount of work decreases at each step (the red columns get shorter)

Exercise 9.1. (JPE, September 2002) Consider three vectors

$$v_1 = \begin{pmatrix} 1 \\ \epsilon \\ 0 \\ 0 \end{pmatrix}, \quad v_2 = \begin{pmatrix} 1 \\ 0 \\ \epsilon \\ 0 \end{pmatrix}, \quad v_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \epsilon \end{pmatrix}.$$

where $\epsilon \ll 1$.

- (a) Use the classical Gram-Schmidt method to compute 3 **orthonormal** vectors q_1, q_2, q_3 , making the approximation that $1 + \epsilon^2 \approx 1$ (that is, replace any term containing ϵ^2 or smaller with zero, **but** retain terms containing ϵ). Are q_i ($i = 1, 2, 3$) pairwise orthogonal? If not, why not?
- (b) Repeat (a) using the modified Gram-Schmidt orthogonalization process. Are the q_i ($i = 1, 2, 3$) pairwise orthogonal? If not, why not?

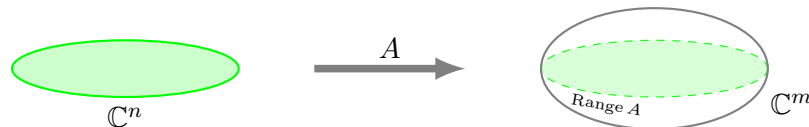
Least Squares

10.1 Definition

A system of linear equations $Ax = b$ with $A \in \mathbb{C}^{m \times n}$, $x \in \mathbb{C}^n$ and $b \in \mathbb{C}^m$, is said to be *overdetermined* if $m > n$. Since there are more equations than unknowns, the system usually has no solutions; so we will write it as $Ax \approx b$.

10.2 Conditions for existence and uniqueness of a solution

Recall that every matrix $A \in \mathbb{C}^{m \times n}$ defines a linear transformation $\mathbb{C}^n \rightarrow \mathbb{C}^m$. Since it takes a smaller space into a larger one, its range is a proper subset of \mathbb{C}^m .



It is clear that the equation $Ax = b$ has a solution if and only if $b \in \text{Range } A$. In the latter case the solution is unique if and only if our linear transformation is injective. The latter occurs if and only if $\text{Ker } A = \{0\}$ (cf. Section 0.6).

10.3 Least squares solution

Let $Ax \approx b$ be an overdetermined linear system. A vector $x \in \mathbb{C}^n$ that minimizes the function

$$E(x) = \|b - Ax\|_2$$

is called a *least squares solution* of $Ax \approx b$. The vector $r = b - Ax$ is called the *residual vector* and $\|r\|_2$ the *residual norm*.

10.4 Normal equations

Let $Ax \approx b$ be an overdetermined linear system. The linear system

$$A^*Ax = A^*b \tag{10.1}$$

is called the *system of normal equations* associated with $Ax \approx b$.

Note that $A^*A \in \mathbb{C}^{n \times n}$, hence (10.1) is an ordinary system of linear equations (not an overdetermined system).

10.5 Theorem

Let $Ax \approx b$ be an overdetermined linear system. Then

- (a) A vector x minimizes $E(x) = \|b - Ax\|_2$ if and only if it is an exact solution of the system $Ax = \hat{b}$, where \hat{b} is the orthogonal projection of b onto $\text{Range } A$.
- (b) A vector x minimizing $E(x)$ always exists. It is unique if and only if A has full rank (equivalently, $\text{Ker } A = \{0\}$).
- (c) A vector x minimizes $E(x)$ if and only if it is a solution of the system of normal equations $A^*Ax = A^*b$.

Proof. Denote $W = \text{Range } A$. We have an orthogonal decomposition $\mathbb{C}^m = W \oplus W^\perp$, in particular $b = \hat{b} + r$, where $\hat{b} \in W$ and $r \in W^\perp$ are uniquely determined by b . Now Pythagorean Theorem 1.16 gives

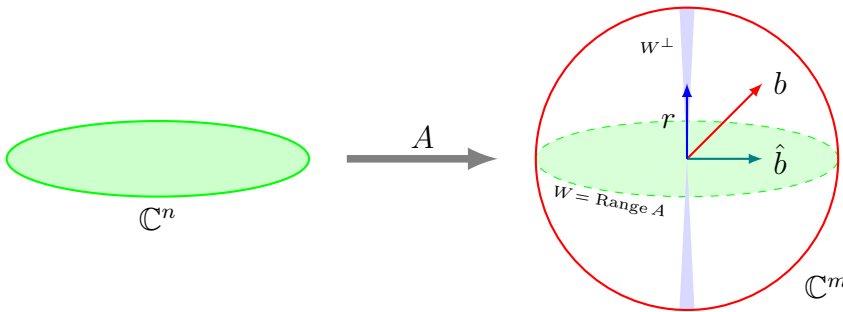
$$\begin{aligned} [E(x)]^2 &= \|b - Ax\|_2^2 = \underbrace{\|b - \hat{b}\|_2^2}_{=r \in W^\perp} + \underbrace{\|\hat{b} - Ax\|_2^2}_{\in W} \\ &= \|r\|_2^2 + \|\hat{b} - Ax\|_2^2 \geq \|r\|_2^2 \end{aligned}$$

Hence, $\min_x E(x) = \|r\|_2$ is attained whenever $Ax = \hat{b}$. Since $\hat{b} \in \text{Range } A$, there is always a vector $x \in \mathbb{C}^n$ such that $Ax = \hat{b}$. The vector x is unique whenever the map $A: \mathbb{C}^n \rightarrow \mathbb{C}^m$ is injective, i.e., whenever $\text{Ker } A = \{0\}$, i.e., whenever A has full rank. This proves (a) and (b).

To prove (c), recall that $(\text{Range } A)^\perp = \text{Ker } A^*$ (by Section 3.7), therefore $r = b - \hat{b} \in \text{Ker } A^*$. Moreover, $b - Ax \in \text{Ker } A^*$ if and only if $Ax = \hat{b}$, because \hat{b} and r are uniquely determined by b . Now

$$x \text{ minimizes } E(x) \Leftrightarrow Ax = \hat{b} \Leftrightarrow Ax - b \in \text{Ker } A^* \Leftrightarrow A^*Ax = A^*b$$

The proof is complete. □



Next we give examples that lead to overdetermined systems and least squares problems.

10.6 Linear least squares fit

Let (x_i, y_i) , $1 \leq i \leq m$, be points in the xy plane. For any straight line $y = a_0 + a_1x$ one defines the “combined distance” from that line to the given points by

$$E(a_0, a_1) = \left[\sum_{i=1}^m (a_0 + a_1x_i - y_i)^2 \right]^{1/2}$$

The line $y = a_0 + a_1x$ that minimizes the function $E(a_0, a_1)$ is called the *least squares fit* to the points (x_i, y_i) . This is a basic tool in statistics. Let

$$A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Then

$$E(a_0, a_1) = \|\mathbf{b} - A\mathbf{x}\|_2$$

Hence the least squares fit is equivalent to the least squares problem $A\mathbf{x} \approx \mathbf{b}$.

10.7 Polynomial least squares fit

Generalizing 10.6, one can fit a polynomial $y = p(x) = a_0 + a_1x + \cdots + a_nx^n$ of degree $n \geq 2$ to a set of data points (x_i, y_i) , $1 \leq i \leq m$, with $m > n + 1$. The least squares fit is based on minimizing the function

$$E(a_0, \dots, a_n) = \left[\sum_{i=1}^m \left(a_0 + a_1x_i + \cdots + a_nx_i^n - y_i \right)^2 \right]^{1/2}$$

This is equivalent to the least squares problem for an overdetermined linear system

$$a_0 + a_1x_i + \cdots + a_nx_i^n \approx y_i \quad 1 \leq i \leq m$$

in which a_0, \dots, a_n are unknowns.

10.8 Continuous least squares fit

Instead of fitting a polynomial $p(x) \in P_n(\mathbb{R})$ of degree n to a discrete data set (x_i, y_i) one can fit it to a continuous function $y = f(x)$ on $[0, 1]$. The least squares fit is based on minimization of

$$E(a_0, \dots, a_n) = \left[\int_0^1 |f(x) - p(x)|^2 dx \right]^{1/2}$$

The solution to this problem is the orthogonal projection of $f(x)$ onto $P_n(\mathbb{R})$.

To find the solution, we use the basis $\{1, x, \dots, x^n\}$ in $P_n(\mathbb{R})$. Then a_0, \dots, a_n can be found by solving the system of equations (analogous to normal equations)

$$\sum_{j=0}^n a_j \langle x^j, x^i \rangle = \langle f, x^i \rangle \quad 1 \leq i \leq n.$$

The matrix of coefficients here is

$$\langle x^j, x^i \rangle = \int_0^1 x^{i+j} dx = \frac{1}{1+i+j}$$

for $0 \leq i, j \leq n$.

Next we present three computer algorithms for solving the least square problem.

10.9 Algorithm 1, based on normal equations

This is the simplest one:

1. Form the matrix A^*A and the vector A^*b .
2. Compute the Cholesky factorization $A^*A = GG^*$.
3. Solve the lower-triangular system $Gz = A^*b$ for z by forward substitution 7.9.
4. Solve the upper-triangular system $G^*x = z$ for x by backward substitution 7.9.

The cost of this algorithm is dominated by Steps 1 and 2. Because the matrix A^*A is symmetric, its computation requires $mn(n+1) \approx mn^2$ flops. The computation of A^*b requires only $2mn$ flops, a relatively small amount which we can ignore. The Cholesky factorization takes $n^3/3$ flops (cf. Section 8.7). Hence the total cost is

$$\approx mn^2 + \frac{1}{3}n^3 \quad \text{flops}$$

Note: the above method is a *computer* algorithm. For manual work (on paper), solving the least squares problem by using normal equations means the following: you compute the matrix A^*A and the vector A^*b and then solve the system $A^*Ax = A^*b$ for x any way you want (by Gaussian elimination, for example).

10.10 Algorithm 2, based on QR decomposition

Using the reduced QR decomposition (Section 9.5) allows us to rewrite the system of normal equations as

$$\hat{R}^* \hat{Q}^* \hat{Q} \hat{R} x = \hat{R}^* \hat{Q}^* b$$

If A has full rank, the matrix \hat{R}^* is nonsingular and we cancel it out. Also, since the columns of \hat{Q} are orthonormal vectors, $\hat{Q}^* \hat{Q} = I$. Hence

$$\hat{R} x = \hat{Q}^* b$$

This suggests the following algorithm:

1. Compute the reduced QR decomposition $A = \hat{Q} \hat{R}$.
2. Compute the vector $\hat{Q}^* b$.
3. Solve the upper-triangular system $\hat{R} x = \hat{Q}^* b$ for x by backward substitution 7.9.

The cost of this algorithm is dominated by Step 1, the reduced QR factorization, which requires

$\approx 2mn^2$ flops

see Section 9.8. This is approximately twice as much as Algorithm 1 takes.

10.11 Algorithm 3, based on SVD

Using the reduced SVD (Section 5.7) allows us to rewrite the system of normal equations as

$$V \hat{D} \hat{U}^* \hat{U} \hat{D} V^* x = V \hat{D} \hat{U}^* b$$

The matrix V is unitary, and we cancel it out. If A has full rank, the matrix \hat{D} is nonsingular and we cancel it, too. Since the columns of \hat{U} are orthonormal vectors, $\hat{U}^* \hat{U} = I$. Hence

$$\hat{D} V^* x = \hat{U}^* b$$

This suggests the following algorithm:

1. Compute the reduced SVD decomposition $A = \hat{U} \hat{D} V^*$.
2. Compute the vector $\hat{U}^* b$.
3. Solve the diagonal system $\hat{D} z = \hat{U}^* b$ for z .
4. Compute $x = Vz$.

The cost of this algorithm is dominated by step 1, the reduced SVD decomposition, which requires

$\approx 2mn^2 + 11n^3$ flops

see Section 9.9. This is approximately the same amount as in Algorithm 2 for $m \gg n$, but for $n \approx m$ this algorithm is much more expensive.

Algorithm 1 is the simplest and the cheapest, but it often gives inaccurate results in numerical calculations. Algorithms 2 and 3 are more complicated and expensive (in terms of flops), but usually give more accurate numerical results, for the reasons we learn in the next chapters.

10.12 Rank deficient matrix A

If $\text{rank } A < n$, then the least squares solution is not unique: the set of solutions is $\{x \in \mathbb{C}^n : Ax = \hat{b}\}$, which is a line or a plane in \mathbb{C}^n parallel to $\text{Ker } A$. In this case the “best” solution is the one of minimal norm:

$$Ax_{\text{best}} = \hat{b} \quad \text{and} \quad \|x_{\text{best}}\|_2 \leq \|x\|_2 \quad \forall x : Ax = \hat{b}$$

Algorithms 1 and 2 fail to find any solution for a rank deficient matrix A . On the contrary, Algorithm 3 easily finds the minimal norm solution as follows.

When solving the diagonal system $\hat{D}z = \hat{U}^*b$ for z , we just set $z_i = 0$ whenever $d_{ii} = 0$ (in that case $(\hat{U}^*b)_i = 0$ automatically). This obviously gives a minimal norm vector z . Since $\|x\|_2 = \|Vz\|_2 = \|z\|_2$, we get a minimal norm vector x as well.

Exercise 10.1. (JPE, September 1997) Let

$$A = \begin{bmatrix} 3 & 3 \\ 0 & 4 \\ 4 & -1 \end{bmatrix}, \quad \text{and} \quad b = \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix}$$

Use the Gram-Schmidt process to find an orthonormal basis for the column space of A . Factor A into a product QR where $Q \in \mathbb{R}^{3 \times 2}$ has an orthonormal set of column vectors and $R \in \mathbb{R}^{2 \times 2}$ is upper triangular. Solve the least squares problem $Ax = b$. Compute the norm of the residual vector, $\|r\|$.

Exercise 10.2. (JPE, May 1998) Given the data (0,1), (3,4) and (6,5), use a QR factorization technique to find the best least squares fit by a linear function. Also, solve the problem via the system of normal equations.

Machine Arithmetic

11.1 Decimal number system

In our decimal system, natural numbers are represented by a sequence of digits. For example, $582 = 5 \cdot 10^2 + 8 \cdot 10 + 2$. Generally,

$$a_n \cdots a_1 a_0 = 10^n a_n + \cdots + 10 a_1 + a_0,$$

where $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ are digits. Fractional numbers require an additional *fractional part*:

$$f = .d_1 d_2 \dots d_t \dots = 10^{-1} d_1 + 10^{-2} d_2 + \cdots + 10^{-t} d_t + \cdots$$

which may be finite or infinite.

11.2 Floating point representation

Alternatively, any real number can be written as a product of a fractional part with a sign and a power of ten:

$$(\pm .d_1 d_2 \dots d_t \dots) \times 10^e$$

where d_i are decimal digits and $e \in \mathbb{Z}$ is an integer. For example,

$$58.2 = 0.582 \times 10^2 = 0.0582 \times 10^3, \quad \text{etc.} \quad (11.1)$$

This is called *floating point* representation of decimal numbers. The part $.d_1 d_2 \dots d_t \dots$ is called *mantissa* and e is called *exponent*. By changing the exponent e with a fixed mantissa $.d_1 d_2 \dots d_t \dots$ we can move (“float”) the decimal point, for example $0.582 \times 10^2 = 58.2$ and $0.582 \times 10^1 = 5.82$.

11.3 Normalized floating point representation

To avoid unnecessary multiple representations of the same number, such as in (11.1), we can require that $d_1 \neq 0$. We say the floating point representation is *normalized* if $d_1 \neq 0$. Then 0.582×10^2 is the only normalized representation of the number 58.2.

For every positive real $r > 0$ there is a unique integer $e \in \mathbb{Z}$ such that $f = 10^{-e}r \in [0.1, 1)$. Then $r = f \times 10^e$ is the normalized representation of r . For most of real numbers, the normalized representation is unique, however, there are exceptions, such as

$$(.9999\dots) \times 10^0 = (.1000\dots) \times 10^1,$$

both representing the number $r = 1$. In such cases one of the two representations has a finite fractional part and the other has an infinite fractional part with trailing nines.

11.4 Binary number system

In the binary number system, the base is 2 (instead of 10), and there are only two digits: 0 and 1. Any natural number N can be written, in the binary system, as a sequence of binary digits:

$$N = (a_n \cdots a_1 a_0)_2 = 2^n a_n + \cdots + 2a_1 + a_0$$

where $a_i \in \{0, 1\}$. For example, $5 = 101_2$, $11 = 1011_2$, $64 = 1000000_2$, etc. Binary system, due to its simplicity, is used by all computers. In the modern computer world, the word *bit* means *binary digit*.

Fractional numbers require an additional *fractional part*:

$$f = (.d_1 d_2 \dots d_t \dots)_2 = 2^{-1}d_1 + 2^{-2}d_2 + \cdots + 2^{-t}d_t + \cdots$$

which may be finite or infinite. For example, $0.5 = 0.1_2$, $0.625 = 0.101_2$, and

$$0.6 = (0.10011001100110011\dots)_2$$

(the blocks 00 and 11 alternate indefinitely). The floating point representation of real numbers in the binary system is given by

$$r = (\pm .d_1 d_2 \dots d_t \dots)_2 \times 2^e$$

where $.d_1 d_2 \dots d_t \dots$ is called *mantissa* and $e \in \mathbb{Z}$ is called *exponent*. Again, we say that the above representation is normalized if $d_1 \neq 0$, this ensures uniqueness for almost all real numbers. Note that $d_1 \neq 0$ implies $d_1 = 1$, i.e., every normalized binary representation begins with a one.

11.5 Other number systems

We can use a number system with any fixed base $\beta \geq 2$. By analogy with Sections 11.1 and 11.4, any natural number N can be written, in that system, as a sequence of digits:

$$N = (a_n \cdots a_1 a_0)_\beta = \beta^n a_n + \cdots + \beta a_1 + a_0$$

where $a_i \in \{0, 1, \dots, \beta - 1\}$ are digits. Fractional numbers require an additional *fractional part*:

$$f = .d_1 d_2 \dots d_t \dots = \beta^{-1} d_1 + \beta^{-2} d_2 + \cdots + \beta^{-t} d_t + \cdots$$

which may be finite or infinite. The floating point representation of real numbers in the system with base β is given by

$$r = (\pm .d_1 d_2 \dots d_t \dots)_\beta \times \beta^e$$

where $.d_1 d_2 \dots d_t \dots$ is called *mantissa* and $e \in \mathbb{Z}$ is called *exponent*. Again, we say that the above representation is normalized if $d_1 \neq 0$, this ensures uniqueness for almost all real numbers.

In the real world, computers can only handle a certain fixed number of digits in their electronic memory. For the same reason, possible values of the exponent e are always limited to a certain fixed interval. This motivates our next definition.

11.6 Machine number systems (an abstract version)

A *machine number system* is specified by four integers, (β, t, L, U) , where

$\beta \geq 2$ is the base

$t \geq 1$ is the length of the mantissa

$L \in \mathbb{Z}$ is the minimal value for the exponent e

$U \in \mathbb{Z}$ is the maximal value for the exponent e (of course, $L \leq U$).

Real numbers in a machine system are represented by

$$r = (\pm .d_1 d_2 \dots d_t)_\beta \times \beta^e, \quad L \leq e \leq U \quad (11.2)$$

The representation must be normalized, i.e., $d_1 \neq 0$.

11.7 Basic properties of machine systems

Suppose we use a machine system with parameters (β, t, L, U) . There are finitely many real numbers that can be represented by (11.2). More precisely, there are

$$2(\beta - 1)\beta^{t-1}(U - L + 1)$$

such numbers. The largest machine number is

$$M = (.(\beta - 1)\beta \dots \beta)_\beta \times \beta^U = \beta^U(1 - \beta^{-t}).$$

The smallest positive machine number is

$$m = (.10 \dots 0)_\beta \times \beta^L = \beta^{L-1}.$$

Note that zero cannot be represented in the above format, since we require $d_1 \neq 0$. For this reason, every real machine systems includes a few special numbers, like zero, that have to be represented differently. Other “special numbers” are $+\infty$ and $-\infty$.

11.8 Two standard machine systems

Most modern computers conform to the IEEE floating-point standard (ANSI/IEEE Standard 754-1985), which specifies two machine systems:

- ① *Single precision* is defined by $\beta = 2$, $t = 24$, $L = -125$ and $U = 128$.
- ② *Double precision* is defined by $\beta = 2$, $t = 53$, $L = -1021$ and $U = 1024$.

11.9 Rounding rules

A machine system with parameters (β, t, L, U) provides *exact* representation for finitely many real numbers. Other real numbers have to be *approximated* by machine numbers. Suppose $x \neq 0$ be a real number with normalized floating point representation

$$x = (\pm 0.d_1 d_2 \dots)_\beta \times \beta^e$$

where the number of digits may be finite or infinite.

If $e > U$ or $e < L$, then x cannot be properly represented in the machine system (it is either “too large” or “too small”). If $e < L$, then x is usually converted to the special number zero. If $e > U$, then x is usually converted to the special number $+\infty$ or $-\infty$.

If $e \in [L, U]$ is within the proper range, then the mantissa of x has to be reduced to t digits (if it is longer than that or infinite). There are two standard versions of such reductions:

- (a) keep the first t digits and **chop off** the rest;
- (b) **round off** to the nearest available, i.e. use the rules

$$\begin{cases} .d_1 \dots d_t & \text{if } d_{t+1} < \beta/2 \\ .d_1 \dots d_t + .0 \dots 01 & \text{if } d_{t+1} \geq \beta/2 \end{cases}$$

11.10 Relative errors

Let x be a real number and x_c its *computer representation* in a machine system with parameters (β, t, L, U) , as described above. We will always assume that x is neither too big nor too small, i.e., its exponent e is within the proper range $L \leq e \leq U$. How accurately does x_c represent x ? (How close is x_c to x ?)

The *absolute error* of the computer representation, i.e., $x_c - x$, may be quite large when the exponent e is big. It is more customary to describe the accuracy in terms of the *relative error* $(x_c - x)/x$ as follows:

$$\frac{x_c - x}{x} = \varepsilon \quad \text{or} \quad x_c = x(1 + \varepsilon).$$

It is easy to see that the maximal possible value of $|\varepsilon|$ is

$$|\varepsilon| \leq \mathbf{u} = \begin{cases} \beta^{1-t} & \text{for chopped arithmetic (a)} \\ \frac{1}{2}\beta^{1-t} & \text{for rounded arithmetic (b)} \end{cases}$$

The number \mathbf{u} is called *unit roundoff* or *machine epsilon*.

11.11 Machine epsilon

Note that the machine epsilon \mathbf{u} is *not* the smallest positive number m represented by the given machine system (cf. Sect. 11.7).

One can describe \mathbf{u} as the smallest positive number $\varepsilon > 0$ such that $(1 + \varepsilon)_c \neq 1$. In other words, \mathbf{u} is the smallest positive value that, when added to one, yields a result different from one.

In more practical terms, \mathbf{u} tells us how many accurate digits machine numbers can carry. If $\mathbf{u} \sim 10^{-p}$, then any machine number x_c carries at most p accurate decimal digits.

For example, suppose for a given machine system $\mathbf{u} \sim 10^{-7}$ and a machine number x_c representing some real number x has value 35.41879236 (when printed on paper or displayed on computer screen). Then we can say that $x \approx 35.41879$, and the digits of x beyond 9 cannot be determined. In particular, the digits 236 in the printed value of x_c are meaningless (they are “trash” to be discarded).

11.12 Machine epsilon for the two standard machine systems

Ⓘ For the IEEE floating-point *single precision* standard with chopped arithmetic $\mathbf{u} = 2^{-23} \approx 1.2 \times 10^{-7}$. In other words, approximately 7 decimal digits are accurate.

Ⓢ For the IEEE floating-point *double precision* standard with chopped arithmetic $\mathbf{u} = 2^{-52} \approx 2.2 \times 10^{-16}$. In other words, approximately 16 decimal digits are accurate.

In the next two examples, we will solve systems of linear equations by using the rules of a machine system. In other words, we will pretend that we are *computers*. This will help us understand how real computers work.

11.13 Example

Let us solve the system of equations

$$\begin{bmatrix} 0.01 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

The exact solution was found in Section 7.13:

$$x = \frac{200}{197} \approx 1.015 \quad \text{and} \quad y = \frac{196}{197} \approx 0.995.$$

Now let us solve this system by using chopped arithmetic with base $\beta = 10$ and $t = 2$ (i.e., our mantissa will be always limited to two decimal digits).

First we use Gaussian elimination (without pivoting). Multiplying the first equation by 100 and subtracting it from the second gives $-197y = -196$. Both numbers 197 and 196 are three digit long, so we must chop the third digit off. This gives $-190y = -190$, hence $y = 1$. Substituting $y = 1$ into the first equation gives $0.01x = 2 - 2 = 0$, hence $x = 0$. Thus our computed solution is

$$x_c = 0 \quad \text{and} \quad y_c = 1.$$

The relative error in x is $(x_c - x)/x = -1$, i.e., the computed x_c is 100% off mark!

Let us increase the length of mantissa to $t = 3$ and repeat the calculations. This gives

$$x_c = 2 \quad \text{and} \quad y_c = 0.994.$$

The relative error in x is now $(x_c - x)/x = 0.97$, i.e., the computed x_c is 97% off mark. Not much of improvement... We postpone the explanation until Chapter 13.

Let us now apply partial pivoting (Section 7.15). First we interchange the rows:

$$\begin{bmatrix} 1 & 3 \\ 0.01 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}.$$

Now we multiply the first equation by 0.01 and subtract it from the second to get $1.97y = 1.96$. With $t = 2$, we must chop off the third digit in both numbers: $1.9y = 1.9$, hence $y = 1$. Substituting $y = 1$ into the first equation gives $x + 3 = 4$, hence $x = 1$, i.e.,

$$x_c = 1 \quad \text{and} \quad y_c = 1.$$

This is a great improvement over the first two solutions (without pivoting). The relative error in x is now $(x_c - x)/x = -0.015$, so the computed x_c is only 1.5% off. The relative error in y is even smaller (about 0.005).

Machine arithmetic with $t = 2$:		Exact arithmetic:		Machine arithmetic with $t = 3$:
		$0.01x + 2y = 2$		
		$x + 3y = 4$		
		\downarrow		
$0.01x + 2y = 2$	Chopping off ←	$0.01x + 2y = 2$		
$-190y = -190$		$-197y = -196$		
\downarrow		\downarrow		
$0.01x + 2y = 2$		$0.01x + 2y = 2$	Chopping off →	$0.01x + 2y = 2$
$y = 1$		$y = \frac{196}{197} \approx 0.9949$		$y = 0.994$
\downarrow		\downarrow		\downarrow
$0.01x = 2 - 2 = 0$	⊕	$0.01x = 2 - \frac{392}{197} = \frac{2}{197}$	⊕	$0.01x = 2 - 1.988 = 0.02$
$y = 1$		$y = \frac{196}{197}$		$y = 0.994$
\downarrow		\downarrow		\downarrow
$x = 0$		$x = \frac{200}{197} \approx 1.0152$		$x = 2$
$y = 1$		$y = \frac{196}{197} \approx 0.9949$		$y = 0.994$

Example 11.13 without pivoting.

Machine arithmetic with $t = 2$:		Exact arithmetic:		Machine arithmetic with $t = 3$:
		$x + 3y = 4$		
		$0.01x + 2y = 2$		
		\downarrow		
$x + 3y = 4$	Chopping off ←	$x + 3y = 4$		
$1.9y = 1.9$		$1.97y = 1.96$		
\downarrow		\downarrow		
$x + 3y = 4$		$x + 3y = 4$	Chopping off →	$x + 3y = 4$
$y = 1$		$y = \frac{196}{197} \approx 0.9949$		$y = 0.994$
\downarrow		\downarrow		\downarrow
$x = 4 - 3 = 1$		$x = 4 - \frac{588}{197} = \frac{200}{197}$		$x = 4 - 2.988 = 1.02$
$y = 1$		$y = \frac{196}{197}$		$y = 0.994$

Example 11.13 with partial pivoting.

Now let us continue the partial pivoting with $t = 3$. We can keep three digits, so $1.97y = 1.96$ gives $y = 1.96/1.97 \approx 0.9949$, which we have to reduce to $y = 0.994$. Substituting this value of y into the first equation gives $x + 2.982 = 4$, which we have to reduce to $x + 2.98 = 4$, hence $x = 1.02$. So now

$$x_c = 1.02 \quad \text{and} \quad y_c = 0.994.$$

The relative error in x is $(x_c - x)/x = 0.0047$, less than 0.5%.

The table below shows the relative error of the numerical solution x_c by Gaussian elimination with pivoting and different lengths of mantissa. We see that the relative error is roughly proportional to the “typical” round-off error 10^{-t} , with a factor of about 2 to 5. We can hardly expect a better accuracy.

	relative error	typical error	factor
$t = 2$	1.5×10^{-2}	10^{-2}	1.5
$t = 3$	4.7×10^{-3}	10^{-3}	4.7
$t = 4$	2.2×10^{-4}	10^{-4}	2.2

Conclusions: Gaussian elimination without pivoting may lead to catastrophic errors, which will remain unexplained until Chapter 13. Pivoting is more reliable – it seems to provide nearly maximum possible accuracy here. But see the next example...

11.14 Example

Let us solve another system of equations:

$$\begin{bmatrix} 3 & 1 \\ 1 & 0.35 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 1.7 \end{bmatrix}$$

The exact solution here is

$$x = 1 \quad \text{and} \quad y = 2.$$

The largest coefficient, 3, is at the top left corner already, so pivoting (partial or complete) would not change anything.

Solving this system in chopped arithmetic with $\beta = 10$ and $t = 2$ gives $x_c = 0$ and $y_c = 5$, which is 150% off. Increasing the length of the mantissa to $t = 3$ gives $x_c = 0.883$ and $y_c = 2.35$, so the relative error is 17%. With $t = 4$, we obtain $x_c = 0.987$ and $y_c = 2.039$, now the relative error is 2%. The table below shows that the relative error of the numerical solutions is roughly proportional to the typical round-off error 10^{-t} , but with a big factor fluctuating around 150 or 200.

	relative error	typical error	factor
$t = 2$	1.5×10^{-0}	10^{-2}	150
$t = 3$	1.7×10^{-1}	10^{-3}	170
$t = 4$	2.0×10^{-2}	10^{-4}	200

We postpone a complete analysis of the above two examples until Chapter 13.

11.15 Computational errors

Let x and y be two real numbers represented in a machine system by x_c and y_c , respectively. An arithmetic operation $x * y$ (where $*$ stands for one of the four basic operations: $+$, $-$, \times , \div) is performed by a computer in the following way. The computer first finds $x_c * y_c$ exactly and then represents that number in its machine system. The result is $z = (x_c * y_c)_c$.

Note that, generally, z is different from $(x * y)_c$, which is the machine representation of the exact result $x * y$. Hence, z is not necessarily the best representation for $x * y$. In other words, the computer makes additional round off errors at each arithmetic operation.

Assuming that $x_c = x(1 + \varepsilon_1)$ and $y_c = y(1 + \varepsilon_2)$ we have

$$(x_c * y_c)_c = (x_c * y_c)(1 + \varepsilon_3) = [x(1 + \varepsilon_1)] * [y(1 + \varepsilon_2)](1 + \varepsilon_3)$$

where $|\varepsilon_1|, |\varepsilon_2|, |\varepsilon_3| \leq \mathbf{u}$.

11.16 Multiplication and division

For multiplication, we have

$$z = xy(1 + \varepsilon_1)(1 + \varepsilon_2)(1 + \varepsilon_3) \approx xy(1 + \varepsilon_1 + \varepsilon_2 + \varepsilon_3)$$

(here we ignore higher order terms like $\varepsilon_1\varepsilon_2$), so the relative error is (approximately) bounded by $3\mathbf{u}$. A similar estimate can be made for division:

$$z = \frac{x(1 + \varepsilon_1)(1 + \varepsilon_3)}{y(1 + \varepsilon_2)} \approx \frac{x}{y}(1 + \varepsilon_1 - \varepsilon_2 + \varepsilon_3).$$

Note: we used Taylor expansion

$$\frac{1}{1 + \varepsilon_2} = 1 - \varepsilon_2 + \varepsilon_2^2 - \varepsilon_2^3 + \dots$$

and again ignored higher order terms.

Thus again the relative error is (approximately) bounded by $3\mathbf{u}$.

Conclusion: machine multiplication and machine division magnify relative errors by a factor of three, at most.

11.17 Addition and subtraction

For addition, we have

$$z = (x + y + x\varepsilon_1 + y\varepsilon_2)(1 + \varepsilon_3) = (x + y) \left(1 + \frac{x\varepsilon_1 + y\varepsilon_2}{x + y} \right) (1 + \varepsilon_3).$$

Again ignoring higher order terms we can bound the relative error of z by

$$\frac{|x| + |y|}{|x + y|} \mathbf{u} + \mathbf{u}.$$

Thus, the operation of addition magnifies relative errors by a factor of

$$\frac{|x| + |y|}{|x + y|} + 1.$$

Similar estimates can be made for subtraction $x - y$: it magnifies relative errors by a factor

$$\frac{|x| + |y|}{|x - y|} + 1.$$

Hence the addition and subtraction magnify relative errors by a variable factor which depends on x and y . This factor may be arbitrarily large if $x + y \approx 0$ for addition or $x - y \approx 0$ for subtraction. This phenomenon is known as *catastrophic cancelation*. It occurred in our Example 11.13 when we solved it without pivoting (see the line marked with double exclamation signs on page 92).

Exercise 11.1. (JPE, September 1993). Solve the system

$$\begin{pmatrix} 0.001 & 1.00 \\ 1.00 & 2.00 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1.00 \\ 3.00 \end{pmatrix}$$

using the LU decomposition with and without partial pivoting and chopped arithmetic with base $\beta = 10$ and $t = 3$ (i.e., work with a three digit mantissa). Obtain computed solutions (x_c, y_c) in both cases. Find the exact solution, compare, make comments.

Exercise 11.2. (JPE, May 2003). Consider the system

$$\begin{pmatrix} \varepsilon & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Assume that $|\varepsilon| \ll 1$. Solve the system by using the LU decomposition with and without partial pivoting and adopting the following rounding off models (at all stages of the computation!):

$$\begin{aligned} a + b\varepsilon &= a && \text{(for } a \neq 0\text{),} \\ a + b/\varepsilon &= b/\varepsilon && \text{(for } b \neq 0\text{).} \end{aligned}$$

Find the exact solution, compare, make comments.

Condition Numbers

12.1 Introduction

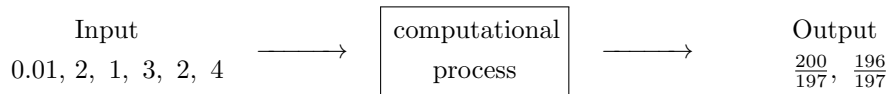
We have seen in Chapter 11 that not all real numbers can be precisely encoded in a given machine system. In most instances there is a difference between the exact value of a real number x and its computer representation x_c (this difference is called *error* of representation). We have also seen that numerical calculations with finite precision (limited length of mantissa) tend to magnify errors, sometimes dramatically. In this chapter our object of study will be the factor by which errors are magnified (that factor will be called *condition number*).

Also we have seen that in numerical analysis it is customary to deal with *relative errors*, given by $|x_c - x|/x$, rather than absolute errors, $|x_c - x|$. A relative error $\sim 10^{-k}$ means that the value of x_c , when recorded in decimal format, has k accurate digits (recall Sect. 11.11). We note that it is the order of magnitude of a relative error that matters, not its exact value. For example, relative errors 0.0001 and 0.0002 are nearly the same, in practical terms: both are of order 10^{-4} , hence both tell us that we can trust approximately four decimal digits in the recorded value of x_c .

12.2 Computational process as a function

Any computational process transforms given numbers (*input*) into computed numbers (*output*).

In Example 11.13 we were given the coefficients of a matrix (0.01, 2, 1, and 3) and the components of a right hand side vector (2 and 4). The results of our computation were two numbers: $x = \frac{200}{197}$ and $y = \frac{196}{197}$.



Given numbers can be viewed as components of a vector, and computed numbers – as components of another vector. Thus every computational process is a transformation of a vector space into another vector space. In our Example 11.13, it was $\mathbb{R}^6 \rightarrow \mathbb{R}^2$.

In other words, every computational process can be viewed as a function $f: V \rightarrow W$, where V and W are vector spaces.

12.3 Condition number of a function

Let V and W be two *normed* vector spaces, and $f: V \rightarrow W$ a function. The *condition number* κ of f at a point $x \in V$ is defined by

$$\kappa = \kappa(f, x) = \lim_{\delta \rightarrow 0} \sup_{\|\Delta x\| \leq \delta} \left(\frac{\|\Delta f\|}{\|f\|} \bigg/ \frac{\|\Delta x\|}{\|x\|} \right)$$

where $\Delta f = f(x + \Delta x) - f(x)$. This is the maximal factor by which small relative errors are magnified by f in the vicinity of the point x .

The condition number characterizes the *sensitivity* of the output, $f(x)$, to small perturbations in the input, x .

12.4 Lemma

Let $\mathbb{C}^n \rightarrow \mathbb{C}^n$ be a linear operator defined by a nonsingular matrix $A \in \mathbb{C}^{n \times n}$, and let $\|\cdot\|$ be a norm on \mathbb{C}^n . Then for every $x \in \mathbb{C}^n$

$$\kappa(A, x) \leq \|A\| \|A^{-1}\| \quad \text{and} \quad \sup_{x \neq 0} \kappa(A, x) = \|A\| \|A^{-1}\|$$

where $\|A\|$ denotes the induced matrix norm.

Proof. Let $y = Ax$ and note that $\Delta y = A(x + \Delta x) - Ax = A(\Delta x)$. Now

$$\kappa(A, x) = \sup_{\Delta x \neq 0} \frac{\|A(\Delta x)\|}{\|\Delta x\|} \frac{\|x\|}{\|Ax\|} = \|A\| \frac{\|x\|}{\|Ax\|}$$

which is defined for all $x \neq 0$. Next

$$\sup_{x \neq 0} \kappa(A, x) = \|A\| \sup_{x \neq 0} \frac{\|x\|}{\|Ax\|} = \|A\| \sup_{y \neq 0} \frac{\|A^{-1}y\|}{\|y\|} = \|A\| \|A^{-1}\|$$

(note that $x \neq 0 \iff y \neq 0$, because A is invertible). □

12.5 Condition number of a matrix

For a nonsingular matrix $A \in \mathbb{C}^{n \times n}$, the condition number with respect to a given matrix norm $\|\cdot\|$ is defined by

$$\kappa(A) = \|A\| \|A^{-1}\| \tag{12.1}$$

We denote by $\kappa_1(A)$, $\kappa_2(A)$, $\kappa_\infty(A)$ the condition numbers with respect to the 1-norm, 2-norm, and ∞ -norm, respectively.

12.6 Main theorem for the condition number of a matrix

Let $A \in \mathbb{C}^{n \times n}$ be a nonsingular matrix and

$$Ax = b \tag{1}$$

$$(A + \Delta A)(x + \Delta x) = b + \Delta b \tag{2}$$

Assume that $\|\Delta A\|$ is small so that $\|\Delta A\| \|A^{-1}\| < 1$. Then

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A) \frac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right).$$

Proof. Expanding out the second equation (2), subtracting from it the first equation (1), and premultiplying by A^{-1} gives

$$\Delta x = -A^{-1} \Delta A(x + \Delta x) + A^{-1} \Delta b.$$

Taking norms and using Section 1.9 and the triangle inequality give

$$\|\Delta x\| \leq \|A^{-1}\| \|\Delta A\| (\|x\| + \|\Delta x\|) + \|A^{-1}\| \|\Delta b\|.$$

Note that $\|b\| \leq \|A\| \|x\|$, hence the above inequality rearranges to

$$\left(1 - \|A^{-1}\| \|\Delta A\|\right) \|\Delta x\| \leq \left(\|A^{-1}\| \|\Delta A\| + \|A^{-1}\| \|A\| \frac{\|\Delta b\|}{\|b\|}\right) \|x\|.$$

Recall that $\|\Delta A\| \|A^{-1}\| < 1$, so the first factor above is positive. The theorem now follows immediately. \square

Note: The smaller the condition number $\kappa(A)$, the smaller upper bound on the relative error $\frac{\|\Delta x\|}{\|x\|}$ we get. The value of $\kappa(A)$ thus characterizes the *sensitivity* of the solution of the linear system $Ax = b$ to small perturbations of both A and b .

Interpretation. Let $Ax = b$ be a system of linear equations to be solved numerically. A computer represents A by $A_c = A + \Delta A$ and b by $b_c = b + \Delta b$. Assume that the computer finds the exact solution x_c of the perturbed system, i.e., x_c satisfies $A_c x_c = b_c$. Denote by $\Delta x = x_c - x$ the resulting error, where x denotes the exact solution of the true system $Ax = b$. Then the relative error $\|\Delta x\|/\|x\|$ can be estimated by Theorem 12.6.

12.7 Corollary

Consider the problem of solving a system of linear equations $Ax = b$ with a nonsingular matrix A . Then:

- (a) If we fix A and vary b , we get a map $f_A: b \mapsto x$. The condition number of f_A satisfies

$$\kappa(f_A, b) \leq \kappa(A) \quad \text{and} \quad \sup_{b \in \mathbb{C}^n} \kappa(A, b) = \kappa(A).$$

- (b) If we fix b and vary A , we get a map $f_b: A \mapsto x$. The condition number of f_b satisfies

$$\kappa(f_b, A) \leq \kappa(A).$$

Proof. These are particular cases of Theorem 12.6: in part (a) we set $\Delta A = 0$, and in part (b) we set $\Delta b = 0$. Then both inequalities easily follow from Theorem 12.6. To prove the equality in (a), note that $x = A^{-1}b$ and apply Lemma 12.4. Incidentally, we note that $\kappa(A^{-1}) = \kappa(A)$ for any nonsingular matrix A . \square

Remark. In part (b), we actually have equality $\kappa(f_b, A) = \kappa(A)$, but the proof is beyond the scope of our course (it can be found in the textbook, page 95).

12.8 Corollary

Assume that in Theorem 12.6 we have $\|\Delta A\| \leq \mathbf{u}\|A\|$ and $\|\Delta b\| \leq \mathbf{u}\|b\|$, i.e., the matrix A and the vector b are represented with the best possible machine accuracy. Then

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{2\mathbf{u}\kappa(A)}{1 - \mathbf{u}\kappa(A)}. \quad (12.2)$$

provided $\mathbf{u}\kappa(A) \ll 1$.

One should note that the formula (12.2) is practically meaningful only if $\mathbf{u}\kappa(A) \ll 1$, otherwise $\|\Delta x\|$ would be comparable to $\|x\|$ so the computed solution could not be trusted at all (in particular, estimating its relative error is rather pointless).

On the other hand, if $\mathbf{u}\kappa(A) \ll 1$, the formula (12.2) can be written simply (though a bit inaccurately) as

$$\frac{\|\Delta x\|}{\|x\|} \lesssim 2\mathbf{u}\kappa(A). \quad (12.3)$$

12.9 Practical interpretation

Assume that $\mathbf{u} \approx 10^{-p}$, i.e., the machine system provides p accurate digits. Now if $\kappa(A) \approx 10^q$ with $q < p$, then $\|\Delta x\|/\|x\| \leq 10^{-(p-q)}$, i.e., the numerical solution provides $p - q$ accurate digits.

We note again, as in Sect. 12.1, that only the order of magnitude of $\kappa(A)$ matters, not its exact value. For instance, there is little difference between $\kappa(A) = 100$ and $\kappa(A) = 200$, both are of order 10^2 .

Linear systems $Ax = b$ with small $\kappa(A)$ ($\sim 1, 10, 10^2$) are often called *well-conditioned*. Those with large $\kappa(A)$ ($\sim 10^3, 10^4$, etc.) are called *ill-conditioned*. Their numerical solutions are unreliable and should be avoided.

12.10 Maxmag and minmag

The *maximum* and *minimum magnification* by a matrix A are defined by

$$\text{maxmag}(A) = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

and

$$\text{minmag}(A) = \min_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \min_{\|x\|=1} \|Ax\|.$$

Note that

$$\text{maxmag}(A) = \|A\|.$$

For singular matrices, $\text{minmag}(A) = 0$. For nonsingular matrices

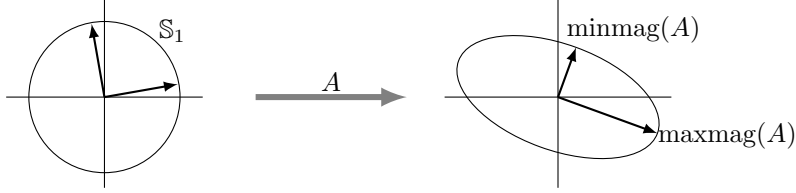
$$\text{minmag}(A) = \min_{y \neq 0} \frac{\|y\|}{\|A^{-1}y\|} = \frac{1}{\max_{y \neq 0} \frac{\|A^{-1}y\|}{\|y\|}} = \frac{1}{\|A^{-1}\|}.$$

Therefore

$$\kappa(A) = \frac{\text{maxmag}(A)}{\text{minmag}(A)} = \frac{\sup_{\|x\|=1} \|Ax\|}{\inf_{\|x\|=1} \|Ax\|} \quad (12.4)$$

The formula (12.4) makes it reasonable to say that for singular matrices $\kappa(A) = \infty$. That is, the singularity is the extreme case of ill-conditioning. Reversing the point of view, we can say that ill-conditioned matrices are “nearly singular”. This relation is made precise in Section 12.12 below.

For the 2-norm, the above relations can be described geometrically with the help of unit sphere defined by (1.1). The linear transformation $A: \mathbb{C}^n \rightarrow \mathbb{C}^n$ maps the unit sphere \mathbb{S}_1 onto an ellipsoid. The longest semiaxis of the latter is $\text{maxmag}(A)$ and its shortest semiaxis is $\text{minmag}(A)$.



Thus, $\kappa_2(A)$ shows how much the linear map $A: \mathbb{C}^n \rightarrow \mathbb{C}^n$ “distorts” the unit sphere \mathbb{S}_1 .

12.11 Properties of the condition numbers

1. If a_j denotes the j -th column of A , then $\kappa(A) \geq \|a_j\|/\|a_i\|$
2. $\kappa(A) \geq 1$ and $\kappa(I) = 1$
3. $\kappa_2(A) = 1$ if and only if A is a multiple of a unitary matrix.
4. For any unitary matrix Q ,

$$\kappa_2(QA) = \kappa_2(AQ) = \kappa_2(A)$$

5. If $D = \text{diag}\{d_1, \dots, d_n\}$ then

$$\kappa_2(D) = \kappa_1(D) = \kappa_\infty(D) = \frac{\max_{1 \leq i \leq n} |d_i|}{\min_{1 \leq i \leq n} |d_i|}$$

6. If A is Hermitian with eigenvalues $\lambda_1, \dots, \lambda_n$, then

$$\kappa_2(A) = \frac{\max_i |\lambda_i|}{\min_i |\lambda_i|}$$

7. If $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$ denote the singular values of A , then

$$\kappa_2(A) = \sigma_1/\sigma_n$$

8. We have

$$[\kappa_2(A)]^2 = \kappa_2(A^*A) = \kappa_2(AA^*) = \frac{\lambda_{\max}(A^*A)}{\lambda_{\min}(A^*A)}$$

9. We have $\kappa_2(A) = \kappa_2(A^*)$.

12.12 Closeness to a singular matrix

We have

$$\min \left\{ \frac{\|A - A_s\|_2}{\|A\|_2} : A_s \text{ is singular} \right\} = \frac{1}{\kappa_2(A)}.$$

In other words, $1/\kappa_2(A)$ is the relative distance from A to the nearest singular matrix.

Proof: This follows from Section 5.11 and Section 12.11 (part 7). \square

12.13 A posteriori error analysis using the residual

In practice the exact solution x of the system $Ax = b$ is rarely known. If it is unknown, how can we determine the relative error $\frac{\|x_c - x\|}{\|x\|}$ of a numerical solution x_c ?

One way to do that is to compute the *residual vector* $r = Ax_c - b$ and see if $r = 0$. Of course, $r = 0$ would imply that x_c is the exact solution, and we can hardly expect that. Now our intuition tells us that if r is small, the numerical solution x_c must be good. But how small should r be?

Note that $Ax_c = b + r$, so x_c is the *exact* solution of a perturbed system

$$A(\underbrace{x + \Delta x}_{x_c}) = b + \underbrace{\Delta b}_r$$

with $\Delta b = r$. Now Theorem 12.6 with $\Delta A = 0$ implies that

$$\frac{\|x_c - x\|}{\|x\|} \leq \kappa(A) \frac{\|r\|}{\|b\|}$$

In practical terms: if A is well conditioned, then the smallness of $\frac{\|r\|}{\|b\|}$ ensures the smallness of the relative error $\frac{\|x_c - x\|}{\|x\|}$. However, if A is ill-conditioned, such a conclusion cannot be made: the smallness of r does not guarantee a good accuracy of x_c .

12.14 Extension to rectangular matrices

For rectangular matrices $A \in \mathbb{C}^{m \times n}$ ($m \neq n$) the inverse A^{-1} is not defined, hence the formula (12.1) does not apply. Instead, the formula (12.4) can be used as a *definition* of the condition number:

$$\kappa(A) \stackrel{\text{def}}{=} \frac{\text{maxmag}(A)}{\text{minmag}(A)} = \frac{\sup_{\|x\|=1} \|Ax\|}{\inf_{\|x\|=1} \|Ax\|}. \quad (12.5)$$

Note that if $m < n$, then A must have a non-trivial kernel (i.e., $Ax = 0$ for some $x \neq 0$); thus the denominator in (12.5) turns zero. So the definition (12.5) only applies if $m > n$. Then one can *prove* (see Exercise 12.5) that

$$[\kappa_2(A)]^2 = \kappa_2(A^*A) = \frac{\lambda_{\max}(A^*A)}{\lambda_{\min}(A^*A)}.$$

Exercise 12.1. (JPE, September 1997). Show that, given a matrix $A \in \mathbb{R}^{n \times n}$, one can choose vectors b and Δb so that if

$$Ax = b$$

$$A(x + \Delta x) = b + \Delta b$$

then

$$\frac{\|\Delta x\|_2}{\|x\|_2} = \kappa_2(A) \frac{\|\Delta b\|_2}{\|b\|_2}$$

Explain the significance of this result for the ‘optimal’ role of condition numbers in the sensitivity analysis of linear systems.

(Hint: use SVD to show that it is enough to consider the case where A is a diagonal matrix.)

Exercise 12.2. (JPE, combined May 1997 and May 2008)

(a) Compute the condition numbers κ_1 , κ_2 and κ_∞ for the matrix

$$A = \begin{pmatrix} 1 & 2 \\ 1.01 & 2 \end{pmatrix}$$

(b) Show that for every non-singular 2×2 matrix A we have $\kappa_1(A) = \kappa_\infty(A)$.

Exercise 12.3. (JPE, September 2002). Consider a linear system $Ax = b$. Let x^* be the exact solution, and let x_c be some computed approximate solution. Let $e = x^* - x_c$ be the error and $r = b - Ax_c$ the residual for x_c . Show that

$$\frac{1}{\kappa(A)} \frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|x^*\|} \leq \kappa(A) \frac{\|r\|}{\|b\|}$$

Interpret the above inequality for $\kappa(A)$ close to 1 and for $\kappa(A)$ large.

Exercise 12.4. Prove properties 6 and 7 of condition numbers listed in Section 12.11.

Exercise 12.5. Suppose the condition number of a rectangular matrix $A \in \mathbb{C}^{m \times n}$ with $m > n$ is defined by (12.5). Prove that

$$[\kappa_2(A)]^2 = \kappa_2(A^*A) = \frac{\lambda_{\max}(A^*A)}{\lambda_{\min}(A^*A)}$$

Numerical Stability

13.1 Introduction

When a system of linear equations $Ax = b$ is solved numerically, the computer only knows machine representations A_c and b_c of A and b . Then, at best, it can find $x_c = A_c^{-1}b_c$, instead of $x = A^{-1}b$. According to 12.3, the resulting relative error will be

$$E_1 = \frac{\|x_c - x\|}{\|x\|} \lesssim 2\mathbf{u}\kappa(A) \quad (13.1)$$

This may be bad enough already when the matrix A is ill-conditioned. However, in reality things appear to be even worse, since the computer does *not* evaluate $x_c = A_c^{-1}b_c$ precisely (apart from trivial cases). The computer executes a certain sequence of arithmetic operations (a program) designed to solve the system $Ax = b$. As the program runs, more and more round-off errors are made at each step, and the errors compound toward the end.

As a result, the computer returns a vector \hat{x}_c *different from* $x_c = A_c^{-1}b_c$, i.e., the actual output \hat{x}_c is *not* the solution of the perturbed system $A_c x_c = b_c$. Since errors are made in the process of computation, we expect to arrive at $\hat{x}_c \neq x_c$, therefore $A_c \hat{x}_c \neq b_c$. The actual output \hat{x}_c depends on how many computational errors are made and how big they are. Those errors depend not only on the machine system but even more on the algorithm that is used to solve the system $Ax = b$. See the diagram below.

Of course, we do not expect the final relative error

$$E_2 = \frac{\|\hat{x}_c - x\|}{\|x\|} \quad (13.2)$$

be smaller than E_1 , but we hope that it will not be much larger either. In other words, a good algorithm should not magnify the errors caused already by conditioning. If this is the case, the algorithm is said to be *stable*.

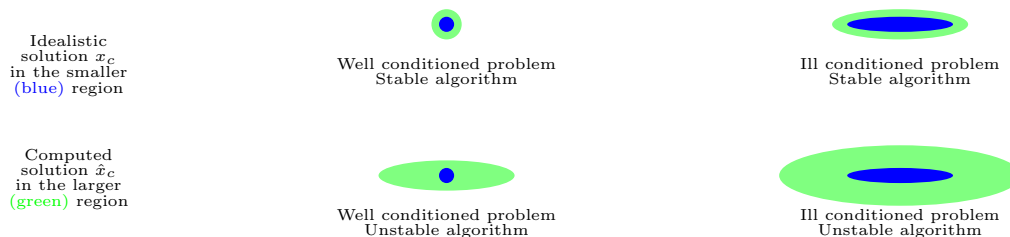
13.2 Stable algorithms (definition)

An algorithm for solving a system of linear equations $Ax = b$ is said to be *stable* (or *numerically stable*) if

$$\frac{\|\hat{x}_c - x\|}{\|x\|} \leq C \mathbf{u} \kappa(A) \quad (13.3)$$

where $C > 0$ is a constant. More precisely, C must be independent of A, b and the machine system, but it may depend on the size of the matrix, n .

We see that for stable algorithms, the actual relative error E_2 is of the same order of magnitude as the ‘idealistic’ relative error E_1 , thus a stable algorithm does not magnify the errors caused already by conditioning.



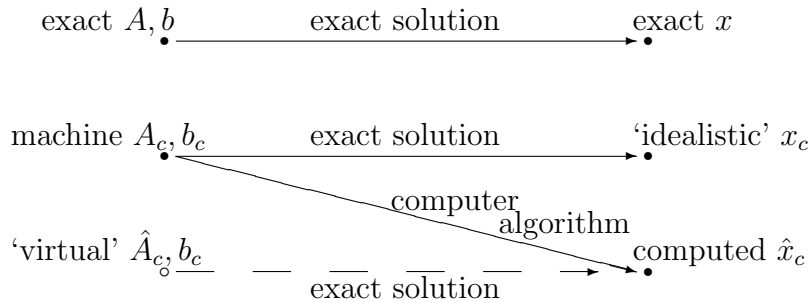
13.3 Backward error analysis

We have seen in Section 12.13 that the computed solution x_c to a system $Ax = b$ can be represented as the *exact* solution of a perturbed system $A(x + \Delta x) = b + \Delta b$. This is the idea of backward error analysis: for each computed solution \hat{x}_c find a perturbed matrix, $\hat{A} = A + \Delta A$, such that

$$(A + \Delta A)\hat{x}_c = b_c.$$

After that Theorem 12.6 can be used to estimate the final relative error $E_2 = \frac{\|\hat{x}_c - x\|}{\|x\|}$ in terms of the relative error $\frac{\|\Delta A\|}{\|A\|}$ of the perturbation of A .

We call $A + \Delta A$ *virtual* matrix, since it is neither given nor computed numerically, its existence is the result of a logical analysis. Moreover, it is far from being unique. One wants to find a virtual matrix $A + \Delta A$ as close to A as possible, i.e., to make $\|\Delta A\|$ as small as possible.



13.4 Backward stable algorithms (definition)

An algorithm for solving a system of linear equations $Ax = b$ is said to be *backward stable* if there exists a virtual matrix $A + \Delta A$ such that

$$\frac{\|\Delta A\|}{\|A\|} \leq C\mathbf{u}$$

where $C > 0$ is a constant. More precisely, C must be independent of A, b and the machine system, but it may depend on the size of the matrix, n .

13.5 Theorem

Every backward stable algorithm is stable.

Proof. By Theorem 12.6,

$$\frac{\|\hat{x}_c - x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A) \frac{\|\Delta A\|}{\|A\|}} \frac{\|\Delta A\|}{\|A\|} \lesssim C\mathbf{u}\kappa(A). \quad \square$$

The proofs of stability (or instability) of algorithms in linear algebra are quite involved. We only present relevant facts here, without proofs.

13.6 Theorem (without proof)

If one uses LU decomposition $A = LU$ for solving a system $Ax = b$, then there is a virtual matrix $A + \Delta A$ such that

$$\|\Delta A\| \leq C\|L\| \|U\| \mathbf{u},$$

where $C > 0$ is a constant independent of A and the machine system (but it may depend on the size of the matrix, n). Thus, the LU algorithm is unstable, its accuracy deteriorates when $\|L\| \|U\|$ is large.

Theorem 13.6 may *explain* why numerical solutions of particular systems of equations happen to be inaccurate.

13.7 Example

In Example 11.13, the LU decomposition (without pivoting) is

$$\begin{bmatrix} 0.01 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 100 & 1 \end{bmatrix} \begin{bmatrix} 0.01 & 2 \\ 0 & -197 \end{bmatrix}$$

hence

$$\|L\|_{\infty} \|U\|_{\infty} = 101 \cdot 197 \sim 10^4.$$

This explains the huge errors of the corresponding numerical solutions that we observed.

13.8 Further facts (without proofs)

- (a) Applying partial pivoting ensures that the entries of L are uniformly bounded: $|L_{ij}| \leq 1$. Also, it is observed in practice that in *most cases* $\|U\| \leq C\|A\|$, hence the partial pivoting algorithm is *usually* stable. There are, however, some unusual cases where partial pivoting becomes unstable. Therefore its accuracy cannot be guaranteed.
- (b) The LU decomposition with complete pivoting is always stable, in this case one can *prove* that $\|U\| \leq C\|A\|$.
- (c) The Cholesky factorization $A = GG^*$ of a positive definite matrix A is a particular form of the LU decomposition, so the above analysis applies. In this case, we know that

$$a_{ii} = \sum_{j=1}^i g_{ij}^2$$

see Section 8.6. Thus, one can easily prove that $\|G\| \leq C\|A\|^{1/2}$, hence the Cholesky factorization is always numerically stable.

These facts may *explain* the accuracy of numerical solutions of particular systems of equations, as illustrated by the following examples.

13.9 Example

In Example 11.13, the matrix

$$\begin{bmatrix} 0.01 & 2 \\ 1 & 3 \end{bmatrix}$$

has singular values $\sigma_1 = 3.7037$ and $\sigma_2 = 0.5319$, hence its condition number is $\kappa_2(A) = \sigma_1/\sigma_2 = 6.96$. This explains a moderate factor (≤ 5) by which relative errors of the numerical solutions are related to the minimal error 10^{-t} when we used a stable algorithm with pivoting in Example 11.13.

13.10 Example

In Example 11.14, the matrix

$$\begin{bmatrix} 3 & 1 \\ 1 & 0.35 \end{bmatrix}$$

has singular values $\sigma_1 = 3.33$ and $\sigma_2 = 0.0150$, hence its condition number is $\kappa_2(A) = \sigma_1/\sigma_2 = 222$. This explains a large factor (up to 200) by which relative errors of the numerical solutions are related to the minimal error 10^{-t} in Example 11.14, even though we used a stable algorithm (the LU decomposition with complete pivoting).

Remember that a stable algorithm *should not increase* errors already caused by conditioning, but it cannot cancel them out.

Exercise 13.1. (JPE, September 2004) Compute the LU decomposition $A = LU$ for the matrix

$$A = \begin{bmatrix} 0.01 & 2 \\ 1 & 3 \end{bmatrix}$$

Compute $\|L\|_\infty\|U\|_\infty$. What does this imply about the numerical stability of solving a system of linear equations $Ax = y$ by LU decomposition without pivoting?

Numerically Stable Least Squares

In Chapter 10 we discussed overdetermined systems of equations

$$Ax \approx b$$

where $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ and presented three algorithms for computing the least squares solution x : (i) based on normal equations, (ii) based on QR decomposition, and (iii) based on SVD.

14.1 Normal equations revisited

Algorithm 10.9 via normal equations has many advantages. It is the simplest and most elegant algorithm for the least squares problem. It should be used whenever the computations are precise, in particular if one solves the problem manually (on paper).

For machine computations, algorithm 10.9 seems to be good, too: it is at least twice as cheap (in terms of flops) as the other two algorithms, so it is the fastest one. It has been used in computer programming almost exclusively before 1970s, when computers were slow and the top speed was high priority.

However, machine computations involve round-off errors. And it has been gradually noticed that Algorithm 10.9 based on normal equations tends to magnify errors and produce inaccurate results to the extent that was eventually deemed unacceptable. As of now, normal equations are almost entirely abandoned and used only to demonstrate how bad things can be.

The loss of accuracy can be best understood in terms of the condition number. If A is the given matrix, then we have seen (Section 12.14) that

$$\kappa_2(A^*A) = [\kappa_2(A)]^2.$$

If the matrix A is ill-conditioned (i.e., its condition number is large), then the matrix A^*A will have a much larger condition number.

More precisely, suppose that our machine arithmetic has precision $\mathbf{u} = 10^{-p}$ (defined in Section 11.11) and let $\kappa_2(A) \sim 10^q$ for some $q > 0$. Then a

numerical solution of the least squares problem should produce $p - q$ accurate decimal digits. However, solving the problem by normal equations would involve the matrix A^*A whose condition number is

$$\kappa_2(A^*A) = [\kappa_2(A)]^2 \sim 10^{2q}.$$

Thus the resulting numerical solution would have only $p - 2q$ accurate decimal digits: we lose an extra q accurate digits.

Such a loss of accuracy may be disastrous. Suppose we use single precision ($p = 7$) and our matrix A is just mildly ill-conditioned, with $\kappa_2(A) \sim 10^3$ (recall Section 12.9). Then solving the problem via normal equations we would give us just $7 - 2 \cdot 3 = 1$ (one!) accurate digit, instead of $7 - 3 = 4$ that can be theoretically expected for the given matrix.

Another extreme example: suppose

$$A = \begin{bmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{bmatrix}$$

where $\varepsilon \approx \frac{1}{2} \sqrt{\mathbf{u}}$. This matrix has condition number

$$\kappa_2(A) = \frac{\sqrt{2 + \varepsilon^2}}{\varepsilon} \approx \frac{2}{\varepsilon},$$

so we can expect the numerical solution to have relative error $\sim \mathbf{u}/\varepsilon \sim \sqrt{\mathbf{u}}$, which is not too bad (in single precision we expect 3-4 accurate digits, and in double precision – eight accurate digits).

However, normal equations require the construction of the matrix

$$A^*A = \begin{bmatrix} 1 + \varepsilon^2 & 1 \\ 1 & 1 + \varepsilon^2 \end{bmatrix}$$

Since $\varepsilon^2 < \mathbf{u}$, the matrix A^*A will be stored in the computer memory as $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ (the additional term ε^2 would disappear due to round-off; recall Section 12.9). The matrix $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ is singular, so the algorithm via normal equations would just fail (the computer program would crush).

Still, it is possible to find a reasonably good numerical solution in the last example, with relative error $\sim \sqrt{\mathbf{u}}$, if one uses more elaborate algorithms.

14.2 QR-based algorithm revisited

Let us examine the accuracy of the QR-based Algorithm 10.10.

Due to Section 12.11 (part 4), $\kappa_2(A) = \kappa_2(QA)$ for any unitary matrix $Q \in \mathbb{C}^{m \times m}$, hence

$$\kappa_2(R) = \kappa_2(Q^*A) = \kappa_2(A).$$

Therefore, this method is safe, regarding the conditioning of the problem, no additional loss of accuracy occurs due to conditioning.

The other aspect of numerical algorithms is stability, described in Chapter 13. The actual error may greatly depend on a particular sequence of computations producing the matrices Q and R .

So far we have mentioned two specific procedures for constructing Q and R : the classical Gram-Schmidt (Sect. 1.25) and the modified Gram-Schmidt (Sect. 9.10). It turns out that the classical Gram-Schmidt is *unstable*, hence it leads to unpredictable round-off errors in numerical computations. The modified Gram-Schmidt is *stable* and generally produces accurate results (see programming assignment).

However, there are better algorithms for computing the QR decomposition: those are based on reflection and rotation matrices. We will learn them in this chapter.

14.3 SVD-based algorithm revisited

Lastly we briefly examine the accuracy of the SVD-based Algorithm 10.11.

Let $A = UDV^*$ be an SVD of A . Due to Section 12.11 (part 4), $\kappa_2(A) = \kappa_2(UA) = \kappa_2(AV)$ for any unitary matrices $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ hence

$$\kappa_2(D) = \kappa_2(U^*AV) = \kappa_2(A).$$

Therefore, this method is just as safe, regarding the conditioning of the problem, as the QR-based Algorithm 10.10.

However, Algorithm 10.11 requires the computation of an SVD for the matrix A , for which no simple algorithm exists (the computation of SVD is beyond the scope of this course). Standard software packages (like MATLAB) use only stable and highly accurate algorithms for the SVD computation. If one uses a standard software package, the SVD-based method 10.11 is as good as the QR-based method 10.10 (see programming assignment).

Just a minor note: the SVD-based method 10.11 is more reliable when the matrix A is almost singular or exactly singular; cf. Section 10.12.

14.4 Hyperplanes

Let V be a finite dimensional vector space. A subspace $W \subset V$ is called a *hyperplane* if $\dim W = \dim V - 1$. If V is an inner product space and $W \subset V$ a hyperplane, then

$$\dim W^\perp = 1.$$

If $x \in W^\perp$ is a nonzero vector, then $W^\perp = \text{span}\{x\}$. In that case

$$W = (W^\perp)^\perp = (\text{span}\{x\})^\perp.$$

Hence any hyperplane W is completely determined by a non-zero vector x orthogonal to it.

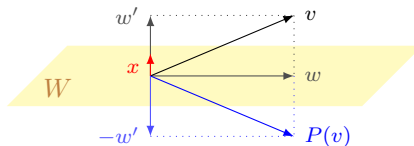
14.5 Reflections

Let $W \subset V$ be a hyperplane in an inner product space V . Recall that $V = W \oplus W^\perp$ (Sect. 1.29). Thus for any vector $v \in V$ we have a unique decomposition $v = w + w'$, where $w \in W$ and $w' \in W^\perp$.

The linear operator P on V defined by

$$Pv = P(w + w') = w - w'$$

is called a *reflection* (or *reflector*) across the hyperplane W .



Note: $P(w) = w$ for all $w \in W$, so P is an identity on W . Also, $P(w') = -w'$ for all $w' \in W^\perp$, so P negates vectors orthogonal to W .

Since W is completely determined by a nonzero vector x orthogonal to W , the reflector can be fully described in terms of x .

14.6 Householder reflection matrices

Let $x \neq 0$ be a nonzero vector in \mathbb{C}^n . The $n \times n$ matrix

$$P = I - 2 \frac{xx^*}{x^*x} = I - 2 \frac{xx^*}{\|x\|^2}$$

is called the *Householder reflection matrix* (or *Householder reflector*) corresponding to x .

Obviously, P is unchanged if x is replaced by cx for any $c \neq 0$.

14.7 Basic properties of Householder reflectors

Let P be the reflector matrix corresponding to a vector $x \neq 0$. Then

- (a) $Px = -x$.
- (b) $Px = x$ whenever $\langle x, x \rangle = 0$.
- (c) P is Hermitian (in the real case it is symmetric).
- (d) P is unitary (in the real case it is orthogonal).
- (e) P is involution, i.e. $P^2 = I$.

Proof. Direct calculation. □

14.8 Theorem

Let $y \in \mathbb{C}^n$. Choose a scalar $\sigma \in \mathbb{C}$ so that

$$|\sigma| = \|y\| \quad \text{and} \quad \sigma \cdot \langle e_1, y \rangle \in \mathbb{R}. \quad (14.1)$$

Suppose that $x = y + \sigma e_1 \neq 0$. Let $P = I - 2xx^*/\|x\|^2$ be the corresponding Householder reflector. Then $Px = -\sigma e_1$.

Proof. By direct calculation, we have

$$\langle y - \sigma e_1, y + \sigma e_1 \rangle = \|y\|^2 - \sigma \langle e_1, y \rangle + \bar{\sigma} \langle y, e_1 \rangle - |\sigma|^2 = 0$$

where we used both identities of (14.1). Now

$$\begin{aligned} 14.7(a) \text{ implies:} & \quad P(y + \sigma e_1) = -y - \sigma e_1 \\ 14.7(b) \text{ implies:} & \quad P(y - \sigma e_1) = y - \sigma e_1 \end{aligned}$$

Adding these two equations proves the theorem. □

14.9 Remarks

- (a) To choose σ in Theorem 14.8, write a polar representation for $\langle e_1, y \rangle = re^{i\theta}$ and then set $\sigma = \pm \|y\| e^{-i\theta}$.
- (b) In the real case, we have $\langle e_1, y \rangle \in \mathbb{R}$, and one can just set $\sigma = \pm \|y\|$. For machine calculation with round-off errors, the better choice is

$$\sigma = \begin{cases} +\|y\| & \text{if } \langle e_1, y \rangle \geq 0 \\ -\|y\| & \text{if } \langle e_1, y \rangle < 0 \end{cases}.$$

Then the computation of the vector $x = y + \sigma e_1$ will be numerically stable, there will be no danger of catastrophic cancellation.

- (c) It is geometrically obvious that for any two vectors $x, y \in \mathbb{R}^n$ with equal 2-norms, $\|x\|_2 = \|y\|_2$, there is a reflector P that takes x to y . In the complex space \mathbb{C}^n , this is *not true*: for generic vectors $x, y \in \mathbb{C}^n$ with equal 2-norms there is no reflector that takes x to y . But according to Theorem 14.8, one can always find a reflector that takes x to cy with some scalar $c \in \mathbb{C}$.

14.10 Corollary

For any vector $y \in \mathbb{C}^n$ there is a scalar σ (which was defined in 14.8 and specified in 14.9) and a matrix P , which is either a reflector or the identity ($P = I$), such that $Py = -\sigma e_1$.

Proof. Apply Theorem 14.8 in the case $y + \sigma e_1 \neq 0$ and set $P = I$ otherwise. \square

14.11 QR Decomposition via Householder reflectors

For any $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ there is a QR decomposition with a unitary matrix $Q \in \mathbb{C}^{m \times m}$ that is a product of Householder reflectors. The number of those reflectors is $\leq n$.

Proof. We use induction on n . Let $n = 1$, so that A is just one column m -vector. By Corollary 14.10 there is a matrix P (a reflector or identity) such that $PA = -\sigma e_1$ with a scalar $\sigma \in \mathbb{C}$. Hence, $A = PR$ where $R = -\sigma e_1$ is an upper triangular matrix $m \times 1$. This provides the basis for our induction.

Now let $n \geq 1$. Let a_1 denote the first column of A . Again, by Corollary 14.10 there is a matrix P (a reflector or identity) such that $Pa_1 = -\sigma e_1$. Hence,

$$PA = \begin{bmatrix} -\sigma & w^* \\ 0 & B \end{bmatrix}$$

where $w \in \mathbb{C}^{n-1}$ and $B \in \mathbb{C}^{(m-1) \times (n-1)}$. By the inductive assumption, there is a unitary matrix $Q_1 \in \mathbb{C}^{(m-1) \times (m-1)}$ and an upper triangular matrix $R_1 \in \mathbb{C}^{(m-1) \times (n-1)}$ such that $B = Q_1 R_1$. Now we expand Q_1 to a unitary $m \times m$ matrix as follows:

$$Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & Q_1 \end{bmatrix}$$

By Section 2.11, the matrix Q_2 is unitary whenever Q_1 is. Furthermore, if Q_1 is a product of $\leq n-1$ reflectors, then the same is true for Q_2 . Now one can easily check that $PA = Q_2 R$ where

$$R = \begin{bmatrix} -\sigma & w^* \\ 0 & R_1 \end{bmatrix}$$

is an upper triangular matrix. Hence, $A = QR$ with $Q = PQ_2$. \square

14.12 Givens rotation matrices

Let $1 \leq p < q \leq m$ and $\theta \in [0, 2\pi)$. The matrix $G = G_{p,q,\theta}$ defined by

$$g_{pp} = \cos \theta, \quad g_{pq} = \sin \theta, \quad g_{qp} = -\sin \theta, \quad g_{qq} = \cos \theta,$$

and $g_{ij} = \delta_{ij}$ otherwise is called a *Givens rotation matrix* (or *Givens rotator*):

$$G = \begin{bmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & c & & & s & & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & -s & & & 1 & & & \\ & & & & & & & c & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow p \\ \\ \\ \\ \leftarrow q \\ \\ \\ \\ \\ \\ \end{matrix}$$

$$\begin{matrix} \uparrow & \uparrow \\ p & q \end{matrix}$$

(here we use convenient shorthand notation $c = \cos \theta$ and $s = \sin \theta$).

It is easy to check that G is an orthogonal matrix (recall Sect. 2.11).

14.13 Geometric description of Givens rotators

Note that the vector $y = Gx$ has components

$$\begin{aligned} y_p &= x_p \cos \theta + x_q \sin \theta \\ y_q &= -x_p \sin \theta + x_q \cos \theta \\ y_i &= x_i \qquad \qquad \qquad \forall i \notin \{p, q\} \end{aligned}$$

Hence the matrix G defines a clockwise rotation through the angle θ of the $x_p x_q$ coordinate plane in \mathbb{R}^m with all the other coordinates left unchanged.

14.14 QR decomposition via Givens rotators

For any $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ there is a QR decomposition with an orthogonal matrix $Q \in \mathbb{R}^{m \times m}$ that is a product of Givens rotators.

Proof. Let a_p be the leftmost column of A that contains a nonzero entry below the main diagonal, i.e.,

$$p = \min\{j \geq 1 : a_{ij} \neq 0 \text{ for some } i > j\}$$

(if such p does not exist, the matrix A is already upper triangular).

Let $a_{qp} \neq 0$ be one of the nonzero entries in the column a_p below the main diagonal, i.e., satisfying $q > p$. Consider the matrix $A' = GA$ where $G = G_{p,q,\theta}$ is a Givens rotator. Note that p and q are already selected, but we are still free to choose θ .

The structure of the matrices $A = (a_{ij})$ and $A' = (a'_{ij})$ can be revealed by examining the following diagram:

$$\begin{bmatrix}
 1 & & & & & & & & & & \\
 & \ddots & & & & & & & & & \\
 & & 1 & & & & & & & & \\
 & & & c & & & & & s & & \\
 & & & & 1 & & & & & & \\
 & & & & & \ddots & & & & & \\
 & & & & & & 1 & & & & \\
 & & & -s & & & & c & & & \\
 & & & & & & & & 1 & & \\
 & & & & & & & & & \ddots & \\
 & & & & & & & & & & 1
 \end{bmatrix}
 \begin{bmatrix}
 * & \cdots & & & & & & & & & \\
 0 & \ddots & & & & & & & & & \\
 0 & 0 & * & & * & & & & & & \\
 0 & \cdots & 0 & a_{pp} & & & & & & & \\
 0 & \cdots & 0 & * & & & & & & & \\
 \vdots & \ddots & \vdots & \vdots & & & & & & & \\
 0 & \cdots & 0 & * & & & & & & & \\
 0 & \cdots & 0 & a_{qp} & & & & & & & \\
 0 & \cdots & 0 & * & & & & & & & \\
 \vdots & \ddots & \vdots & \vdots & & & & & & & \\
 0 & \cdots & 0 & * & \cdots & & & & & &
 \end{bmatrix}$$

$$= \begin{bmatrix}
 * & \cdots & & & & & & & & & \\
 0 & \ddots & & & & & & & & & \\
 0 & 0 & * & & * & & & & & & \\
 0 & \cdots & 0 & a'_{pp} & & & & & & & \\
 0 & \cdots & 0 & * & & & & & & & \\
 \vdots & \ddots & \vdots & \vdots & & & & & & & \\
 0 & \cdots & 0 & * & & & & & & & \\
 0 & \cdots & 0 & a'_{qp} & & & & & & & \\
 0 & \cdots & 0 & * & & & & & & & \\
 \vdots & \ddots & \vdots & \vdots & & & & & & & \\
 0 & \cdots & 0 & * & \cdots & & & & & &
 \end{bmatrix}$$

The following facts can be verified by direct inspection:

- (a) The first $p - 1$ columns of A' are the same as those of A , hence they are still zero below the main diagonal.

- (b) In the p -th column, only the components a'_{pp} and a'_{qp} will be different from the corresponding components of A (hence, if A had some zeros in the p -th column below the main diagonal, those will propagate to A').
- (c) The subdiagonal component a'_{qp} of A' is given by

$$a'_{qp} = -a_{pp} \sin \theta + a_{qp} \cos \theta.$$

Our goal is to make $a'_{qp} = 0$. We can achieve this by using our freedom to choose θ . The following choice will do it:

$$\cos \theta = \frac{a_{pp}}{\sqrt{a_{pp}^2 + a_{qp}^2}} \quad \text{and} \quad \sin \theta = \frac{a_{qp}}{\sqrt{a_{pp}^2 + a_{qp}^2}}. \quad (14.2)$$

Note that one never actually evaluates the angle θ , since $G_{p,q,\theta}$ only contains $\cos \theta$ and $\sin \theta$, and these are given by the above formulas.

In this way we eliminate one nonzero element a_{qp} below the main diagonal. Working from left to right, one can convert A into an upper triangular matrix $\tilde{G}A = R$ where

$$\tilde{G} = G_{p_k, q_k, \theta_k} \cdots G_{p_1, q_1, \theta_1}$$

will be a product of Givens rotators. Each nonzero element of A below the main diagonal requires one multiplication by a rotator. Then we get $A = QR$ with an orthogonal matrix

$$Q = \tilde{G}^T = G_{p_1, q_1, \theta_1}^T \cdots G_{p_k, q_k, \theta_k}^T.$$

It remains to note that $G_{p,q,\theta}^T = G_{p,q,-\theta}$, so it is a Givens rotator, too. \square

14.15 Cost of QR via Givens rotators

The evaluation of $\cos \theta$ and $\sin \theta$ by (14.2) takes 6 flops (the square root extraction is counted here as one flop). The subsequent multiplication of A by $G_{p,q,\theta}$ takes $6n$ flops. Thus, the elimination of one nonzero subdiagonal component of A takes $6n + 6$ flops. If A originally has k nonzero subdiagonal entries, the QR decomposition via Givens rotators will take $6k(n + 1) \sim 6kn$ flops.

When the lower triangle of A is mostly filled with nonzero numbers, then k is close to its maximal value, $mn - n^2/2$. In that case the total cost will be very high, it will greatly exceed the cost of QR via Householder reflectors. Hence Givens rotators are very inefficient for generic matrices.

However, Givens rotators work well if the matrix A is *sparse*, i.e., contains just a few nonzero components below the main diagonal. Then Givens rotators can give the quickest result. We will see such instances later.

Exercise 14.1. Let $x, y \in \mathbb{C}^n$ be such that $x \neq y$ and $\|x\|_2 = \|y\|_2 \neq 0$. Show that there is a reflector matrix P such that $Px = y$ **if and only if** $\langle x, y \rangle \in \mathbb{R}$. For an extra credit: show that if the above reflector exists, then it is unique.

Exercise 14.2. (simplified of JPE, May 2011) Prove that any Givens rotator matrix in \mathbb{R}^2 is a product of two Householder reflector matrices. Can a Householder reflector matrix be a product of Givens rotator matrices?

Exercise 14.3 (Bonus). (JPE May, 2010) Let

$$A = \begin{bmatrix} 3 & -3 \\ 0 & 4 \\ 4 & 1 \end{bmatrix}$$

- (a) Find the QR factorization of A by Householder reflectors.
- (b) Use the results in (a) to find the least squares solution of $Ax = b$, where

$$b = [16 \ 11 \ 17]^T$$

(Note: there is a typo in the original JPE exam, it is corrected here.)

Computation of Eigenvalues: Theory

15.1 Preface

Eigenvalues of a matrix $A \in \mathbb{C}^{n \times n}$ are the roots of its characteristic polynomial

$$C_A(x) = \det(xI - A), \quad \text{where } \deg C_A(x) = n.$$

It is a consequence of the famous Galois group theory (Abel's theorem) that there is no closed formula for the roots of a generic polynomial of degree > 4 . Hence, there are no finite algorithms for computation of the roots of polynomials (or eigenvalues, for that matter).

Thus, all the methods for computing eigenvalues of matrices of size $n \geq 5$ are necessarily iterative, they provide successive approximations to the eigenvalues, but never exact results. Furthermore, even though for $n = 3$ and $n = 4$ exact formulas exist, they are rather impractical and often numerically unstable, so even in these cases iterative methods are used instead.

For this reason, matrix decompositions that reveal eigenvalues (Schur and SVD) cannot be implemented by finite algorithms. On the other hand, decompositions that do *not* reveal eigenvalues (QR, LU, Cholesky) *can* be implemented by finite algorithms (and we have seen some of those):

Decompositions revealing eigenvalues	Decompositions not revealing eigenvalues
SVD LU	Schur Cholesky QR
No finite algorithms exist	Finite algorithms exist

In the last chapters of the course we will learn iterative algorithms for computing eigenvalues and eigenvectors. We note that the most important

matrix decomposition, SVD, requires the eigenvalues of a Hermitian positive semidefinite matrix A^*A . Hence it is particularly important to develop algorithms for computing eigenvalues of Hermitian matrices.

If an eigenvalue λ of a matrix A is known, an eigenvector x corresponding to λ can be found by solving the linear system $(A - \lambda I)x = 0$ (say, by LU decomposition). Conversely, if an eigenvector x is known, the corresponding eigenvalue λ can be immediately found by simple formula $\lambda = (Ax)_i/x_i$, whenever $x_i \neq 0$. Thus eigenvalues help to compute eigenvectors and vice versa. For this reason eigenvalues and eigenvectors are often computed ‘in parallel’. In this chapter, we develop a theoretical basis for computation of eigenvalues and eigenvectors, while in the next two chapters we will turn to practical algorithms.

15.2 Rayleigh quotient

Let $A \in \mathbb{C}^{n \times n}$. We call

$$r(x) = \frac{x^*Ax}{x^*x} = \frac{\langle Ax, x \rangle}{\langle x, x \rangle}, \quad x \neq 0$$

the *Rayleigh quotient* of x .

15.3 Restriction to the unit sphere

The Rayleigh quotient is defined for all vectors $x \neq 0$, so it is a function on $\mathbb{C}^n \setminus \{0\}$, with values in \mathbb{C} . However, for any nonzero scalar $c \neq 0$

$$r(cx) = \frac{\langle cAx, cx \rangle}{\langle cx, cx \rangle} = \frac{|c|^2 \langle Ax, x \rangle}{|c|^2 \langle x, x \rangle} = \frac{\langle Ax, x \rangle}{\langle x, x \rangle} = r(x),$$

hence r is constant on the line $\text{span}\{x\}$. Since any nonzero vector $x \neq 0$ is a scalar multiple of a unit vector $u = x/\|x\|$, the function $r(x)$ is completely defined by its values on the unit sphere (1.1)

$$\mathbb{S}_1 = \{u \in \mathbb{C}^n : \|u\| = 1\}.$$

On the unit sphere \mathbb{S}_1 the Rayleigh quotient can be computed by a simpler formula:

$$r(u) = u^*Au = \langle Au, u \rangle.$$

Note that $r(u)$ is a quadratic function of the coordinates of $u \in \mathbb{S}_1$.

15.4 Properties of Rayleigh quotient

(a) If A is Hermitian, then $r(x) \in \mathbb{R}$ for any nonzero vector $x \in \mathbb{C}^n$.

[This follows from Section 4.12(a)]

(b) If x is an eigenvector with eigenvalue λ , then $Ax = \lambda x$ and so

$$r(x) = \lambda$$

(c) If x is an arbitrary unit vector, then $r(x)x = (x^*Ax)x$ is the orthogonal projection of the vector Ax onto the line spanned by x . Hence

$$\|Ax - r(x)x\|_2 = \min_{\mu \in \mathbb{C}} \|Ax - \mu x\|_2$$

Note: if one regards x as an ‘approximate’ eigenvector, then the Rayleigh quotient $r(x)$ is the best choice that one can make for the associated ‘approximate’ eigenvalue in the sense that the value $\mu = r(x)$ comes closest (in the 2-norm) to achieving the desired relation $Ax - \mu x = 0$.

Let $A \in \mathbb{C}^{n \times n}$ and x a unit eigenvector of A corresponding to eigenvalue λ . Let $y \approx x$ be another unit vector close to x and $r = y^*Ay$. It follows from the above that $r \approx \lambda$, but how close is r to λ ?

15.5 Theorem

Let $A \in \mathbb{C}^{n \times n}$ and x a unit eigenvector of A corresponding to eigenvalue λ . Let y be another unit vector and $r = y^*Ay$. Then

$$|\lambda - r| \leq 2 \|A\|_2 \|x - y\|_2. \quad (15.1)$$

Moreover, if A is a Hermitian matrix, then there is a constant $C = C(A) > 0$ such that

$$|\lambda - r| \leq C \|x - y\|_2^2. \quad (15.2)$$

Proof. To prove (15.1), we write

$$\lambda - r = x^*A(x - y) + (x - y)^*Ay.$$

Now the triangle inequality and Cauchy-Schwarz inequality readily give (15.1).

Now, assume that A is Hermitian. Then there is an ONB of eigenvectors, and we can assume that x is one of them. Denote that ONB by $\{x, x_2, \dots, x_n\}$ and the corresponding eigenvalues by $\lambda, \lambda_2, \dots, \lambda_n$. Let $y = cx + c_2x_2 + \dots + c_nx_n$. Then

$$\|y - x\|^2 = |c - 1|^2 + \sum_{i=2}^n |c_i|^2 \geq \sum_{i=2}^n |c_i|^2$$

On the other hand, $\|y\| = 1$, i.e., $c^2 + c_2^2 + \dots + c_n^2 = 1$, so

$$\lambda = \lambda|c|^2 + \sum_{i=2}^n \lambda|c_i|^2$$

Now, $Ay = c\lambda x + \sum_{i=2}^n c_i\lambda_i x_i$, therefore

$$r = \langle Ay, y \rangle = \lambda|c|^2 + \sum_{i=2}^n \lambda_i|c_i|^2.$$

Subtracting the last two formulas gives

$$\lambda - r = \sum_{i=2}^n (\lambda - \lambda_i)|c_i|^2.$$

The result now follows with

$$C = \max_{2 \leq i \leq n} |\lambda - \lambda_i|.$$

The theorem is proved. \square

Note: one may feel uncomfortable to deal with an unspecified constant C in (15.2). In fact, it easily follows from Sect. 4.20 that $C \leq 2\|A\|_2$.

15.6 Lemma

Let L and G be subspaces of \mathbb{C}^n and $\dim G > \dim L$. Then there is a nonzero vector $y \in G$ orthogonal to L , i.e., $y \in L^\perp$.

Proof. By way of contradiction, suppose $G \cap L^\perp = \{0\}$. Then $G \oplus L^\perp$ would be a subspace of \mathbb{C}^n , and its dimension would be

$$\dim G + \dim L^\perp = \dim G + n - \dim L > n,$$

which is impossible. \square

Next let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix with eigenvalues $\lambda_1, \dots, \lambda_n$. Since they are all real, we can number them in an increasing order, so that

$$\lambda_1 \leq \dots \leq \lambda_n.$$

Also recall that $x^*Ax \in \mathbb{R}$ for any vector $x \in \mathbb{C}$, by Sect. 15.4 (a).

15.7 Courant-Fisher Minimax Theorem

Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$. Then for every $i = 1, \dots, n$

$$\lambda_i = \min_{L: \dim L=i} \max_{x \in L \setminus \{0\}} \frac{x^* Ax}{x^* x}$$

where L stands for a vector subspace of \mathbb{C}^n .

Proof. Let $\{u_1, \dots, u_n\}$ be an ONB of eigenvectors of A corresponding to the eigenvalues $\lambda_1, \dots, \lambda_n$. If $\dim L = i$, then by Lemma 15.6 there is a nonzero vector $y \in L \setminus \{0\}$ orthogonal to the space $\text{span}\{u_1, \dots, u_{i-1}\}$. Hence, the first $i - 1$ coordinates of y are zero, i.e. $y = \sum_{j=i}^n c_j u_j$. Thus

$$\frac{y^* Ay}{y^* y} = \frac{\sum_{j=i}^n |c_j|^2 \lambda_j}{\sum_{j=i}^n |c_j|^2} \geq \lambda_i$$

because $\lambda_j \geq \lambda_i$ for all $j = i, \dots, n$. Therefore,

$$\max_{x \in L \setminus \{0\}} \frac{x^* Ax}{x^* x} \geq \frac{y^* Ay}{y^* y} \geq \lambda_i.$$

On the other hand, let $L_i = \text{span}\{u_1, \dots, u_i\}$. Obviously, $\dim L_i = i$ and for every nonzero vector $x \in L_i$ we have $x = \sum_{j=1}^i c_j u_j$, so

$$\frac{x^* Ax}{x^* x} = \frac{\sum_{j=1}^i |c_j|^2 \lambda_j}{\sum_{j=1}^i |c_j|^2} \leq \lambda_i$$

because $\lambda_j \leq \lambda_i$ for all $j = 1, \dots, i$. The theorem is proved. \square

15.8 Corollary

A close examination of the proof of the above theorem gives us

$$\lambda_i = \max_{x \in L_i \setminus \{0\}} \frac{x^* Ax}{x^* x}, \quad (15.3)$$

where $L_i = \text{span}\{u_1, \dots, u_i\}$.

15.9 Theorem

Let A and ΔA be two Hermitian matrices. Let $\alpha_1 \leq \dots \leq \alpha_n$ be the eigenvalues of A . Let δ_{\min} and δ_{\max} the smallest and the largest eigenvalues of ΔA . Denote the eigenvalues of the matrix $B = A + \Delta A$ by $\beta_1 \leq \dots \leq \beta_n$. Then for each $i = 1, \dots, n$

$$\alpha_i + \delta_{\min} \leq \beta_i \leq \alpha_i + \delta_{\max}. \quad (15.4)$$

Proof. Let $\{u_1, \dots, u_n\}$ be an ONB of eigenvectors of A corresponding to its eigenvalues $\alpha_1, \dots, \alpha_n$. For each $i = 1, \dots, n$, let $L_i = \text{span}\{u_1, \dots, u_i\}$. Then $\dim L_i = i$, hence by Theorem 15.7 and Corollary 15.8

$$\begin{aligned} \beta_i &\leq \max_{x \in L_i \setminus \{0\}} \frac{x^* B x}{x^* x} \\ &= \max_{x \in L_i \setminus \{0\}} \left[\frac{x^* A x}{x^* x} + \frac{x^* \Delta A x}{x^* x} \right] \\ &\leq \max_{x \in L_i \setminus \{0\}} \frac{x^* A x}{x^* x} + \max_{x \in L_i \setminus \{0\}} \frac{x^* \Delta A x}{x^* x} \\ &= \alpha_i + \max_{x \in L_i \setminus \{0\}} \frac{x^* \Delta A x}{x^* x} \\ &\leq \alpha_i + \max_{x \in \mathbb{C}^n \setminus \{0\}} \frac{x^* \Delta A x}{x^* x} \\ &= \alpha_i + \delta_{\max} \end{aligned}$$

where at the last step we used Section 4.12 (b). Thus we proved the right inequality in (15.4). The left one can be proved by a similar calculation, but we can achieve the result quicker by a trick. We can rewrite that inequality as

$$\alpha_i \leq \beta_i - \delta_{\min} \quad (15.5)$$

and note that $-\delta_{\min}$ is the *largest* eigenvalue of the matrix $-\Delta A$. Then we apply the (already proven) right inequality in (15.4) to the matrices B , $-\Delta A$ and $A = B + (-\Delta A)$ and obtain (15.5). The theorem is completely proved. \square

15.10 Corollary

Since $\|\Delta A\|_2 = \max\{|\delta_{\min}|, |\delta_{\max}|\}$ (due to Section 4.20), we have

$$|\alpha_i - \beta_i| \leq \|\Delta A\|_2 \quad \forall i = 1, \dots, n$$

15.11 Approximation analysis using the residual

Suppose one finds an approximate eigenvalue μ and an approximate unit eigenvector x of a matrix A . How can one estimate the closeness of μ to the actual (but unknown) eigenvalue λ of A ?

One way to do this is to compute the residual $r = Ax - \mu x$. If $r = 0$, then μ is an eigenvalue of A . If that r is small, then one may hope that μ is close to an eigenvalue of A . But how close is it?

To place this question in the context of the previous theorems, we define the matrix $\Delta A = -rx^*$. Then

$$(A + \Delta A)x = Ax - rx^*x = \mu x,$$

so (μ, x) are an exact (!) eigenpair of the perturbed matrix $A + \Delta A$. Also note that $\|\Delta A\|_2 = \|r\|_2$, so the norm $\|\Delta A\|_2$ is readily computable.

If we could apply Corollary 15.10, we would be able to conclude that

$$|\mu - \lambda| \leq \|\Delta A\|_2 = \|r\|_2. \quad (15.6)$$

But! In Corollary 15.10 the matrices A and ΔA must be Hermitian. And our matrices (especially $\Delta A = -rx^*$) may not be Hermitian. Thus we need to extend the estimate in Corollary 15.10 to more general matrices.

15.12 Bauer-Fike theorem

Let $A \in \mathbb{C}^{n \times n}$ be a diagonalizable matrix, so that

$$X^{-1}AX = D = \text{diag}\{\lambda_1, \dots, \lambda_n\}$$

If μ is an eigenvalue of a perturbed matrix $A + \Delta A$, then

$$\min_{1 \leq i \leq n} |\mu - \lambda_i| \leq \kappa_p(X) \|\Delta A\|_p$$

where $\|\cdot\|_p$ stands for any p -norm ($1 \leq p \leq \infty$).

Proof. The matrix $A + \Delta A - \mu I$ is singular. Hence so is the matrix

$$X^{-1}(A + \Delta A - \mu I)X = D + X^{-1}\Delta AX - \mu I.$$

If $D - \mu I$ is singular, then μ is an eigenvalue of A and the claim is trivial. Otherwise

$$(D - \mu I)^{-1}[D + X^{-1}\Delta AX - \mu I] = I + (D - \mu I)^{-1}(X^{-1}\Delta AX)$$

is a singular matrix. The Neumann lemma (Exercise 1.2) implies

$$1 \leq \|(D - \mu I)^{-1}(X^{-1}\Delta AX)\|_p \leq \|(D - \mu I)^{-1}\|_p \|X^{-1}\|_p \|\Delta A\|_p \|X\|_p$$

Lastly, observe that $(D - \mu I)^{-1}$ is diagonal, so

$$\|(D - \mu I)^{-1}\|_p = \max_{1 \leq i \leq n} \frac{1}{|\lambda_i - \mu|} = \frac{1}{\min_{1 \leq i \leq n} |\lambda_i - \mu|}$$

The theorem now follows. □

15.13 Corollary

If A is a normal matrix, then in the above theorem

$$\min_{1 \leq i \leq n} |\mu - \lambda_i| \leq \|\Delta A\|_2$$

because X is a unitary matrix and so $\kappa_2(X) = 1$. Thus, (15.6) is actually true for any normal matrix, in particular for any Hermitian matrix A .

Theorem 15.12 answers the question raised in Section 15.11, it gives an estimate on $|\mu - \lambda|$ in terms of $\|\Delta A\|_p$ and $\kappa_p(X)$. However, this answer may not be good enough – it gives *one* estimate for *all* eigenvalues. In practical applications, some eigenvalues can be estimated better than others. It is important then to develop finer estimates for individual eigenvalues.

15.14 Left eigenvectors (definition)

Let $A \in \mathbb{C}^{n \times n}$. A nonzero vector $x \in \mathbb{C}^n$ is called a *left eigenvector* of A corresponding to an eigenvalue λ if

$$x^* A = \lambda x^*$$

Note: this is equivalent to

$$A^* x = \bar{\lambda} x,$$

i.e., x is an ordinary (“right”) eigenvector of A^* corresponding to the eigenvalue $\bar{\lambda}$:

$$(\lambda, x) \text{ is a left eigenpair for } A \iff (\bar{\lambda}, x) \text{ is a right eigenpair for } A^*. \quad (15.7)$$

But why don’t we introduce a notion of *left eigenvalue*?

15.15 Lemma

A matrix A has a left eigenvector $x \neq 0$ corresponding to eigenvalue λ if and only if λ is a root of the characteristic polynomial of A :

$$(\lambda, x) \text{ is a left eigenpair for } A \iff (\lambda, y) \text{ is a right eigenpair for } A \text{ for some } y \neq 0$$

Proof. $A^* x = \bar{\lambda} x$ can be written as $(A^* - \bar{\lambda} I)x = 0$, and since $x \neq 0$, we conclude that $\det(A^* - \bar{\lambda} I) = 0$. Due to (0.3), we have $\det(A - \lambda I) = 0$, i.e., $C_A(\lambda) = 0$. \square

Lemma 15.15 explains why we do not introduce a notion of *left eigenvalue*: the set of eigenvalues for left eigenvectors is just the same as the set of eigenvalues for ordinary (right) eigenvectors.

Our next question is: for a given eigenvalue λ of A , how different the corresponding left and right eigenvectors are? More precisely, how different the corresponding left and right *eigenspaces* are?

15.16 Lemma

For any eigenvalue λ of A , the dimension of the right eigenspace equals the dimension of the left eigenspace (i.e., the *geometric multiplicity* of λ is the same, in the left and right senses).

Proof. Recall that the ordinary (right) eigenspace is $\text{Ker}(A - \lambda I)$. By (15.7), the left eigenspace is $\text{Ker}(A^* - \bar{\lambda}I)$. Now we have, due to (0.1):

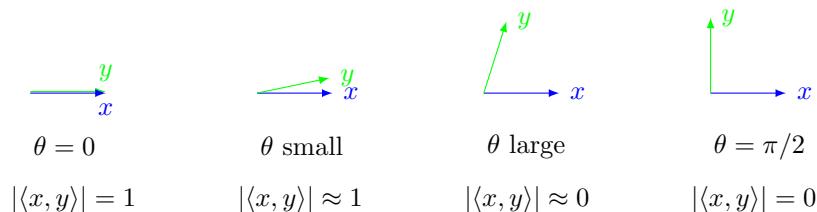
$$\dim \text{Ker}(A - \lambda I) = n - \text{rank}(A - \lambda I) = n - \text{rank}(A^* - \bar{\lambda}I) = \dim \text{Ker}(A^* - \bar{\lambda}I).$$

We used the fact that $\text{rank } B = \text{rank } B^*$ for any matrix B . □

Lemma 15.16 shows that for each eigenvalue λ of A the corresponding left and right eigenspaces have the same dimension. But do those spaces have to coincide? If not, do they have to be close to each other?

When both eigenspaces are one-dimensional and real, their closeness can be measured by the angle between them. If x is a right unit eigenvector and y is a left unit eigenvector, then the angle θ between the corresponding eigenspaces $\text{span}(x)$ and $\text{span}(y)$ satisfies $\cos \theta = \langle x, y \rangle$; cf. (1.22). Thus the “indicator of closeness” of those eigenspaces will be the inner product $\langle x, y \rangle = y^*x$.

Recall that by Cauchy-Schwarz inequality (Section 1.12) for any unit vectors x and y we have $|\langle x, y \rangle| \leq 1$, so the range of values of $|\langle x, y \rangle|$ is the interval $[0, 1]$. The angle between the left and right eigenspaces $\text{span}(x)$ and $\text{span}(y)$ is small if $|\langle x, y \rangle|$ is close to 1, and large if $|\langle x, y \rangle|$ is close to 0. These spaces coincide if $|\langle x, y \rangle| = 1$ and are orthogonal to each other if $|\langle x, y \rangle| = 0$.



15.17 Lemma

Let λ be an eigenvalue with a right eigenvector x , and $\mu \neq \lambda$ be *another* eigenvalue with a left eigenvector y . Then $\langle x, y \rangle = 0$, i.e. $x \perp y$.

(In other words, left and right eigenspaces corresponding to *different* eigenvalues are orthogonal to one another.)

Proof. First, we have $\langle Ax, y \rangle = \lambda \langle x, y \rangle$. On the other hand, according to (15.7), we have

$$\langle Ax, y \rangle = \langle x, A^*y \rangle = \langle x, \bar{\mu}y \rangle = \mu \langle x, y \rangle.$$

Hence, $\lambda \langle x, y \rangle = \mu \langle x, y \rangle$. Since $\lambda \neq \mu$, we must have $\langle x, y \rangle = 0$. □

15.18 Lemma

Let λ be a *simple* eigenvalue (this means its algebraic multiplicity is one) with right and left eigenvectors x and y , respectively. Then $\langle x, y \rangle \neq 0$.

(In other words, left and right eigenspaces corresponding to the *same* simple eigenvalue λ cannot be orthogonal to each other.)

Proof. We can always choose *unit* eigenvectors, in particular let $\|x\| = 1$. By the Schur decomposition (Section 6.1), there is a unitary matrix Q such that

$$Q^* A Q = \begin{bmatrix} \lambda & h^* \\ 0 & B \end{bmatrix}$$

with some $h \in \mathbb{C}^{n-1}$ and $B \in \mathbb{C}^{(n-1) \times (n-1)}$. According to the proof in Section 6.1, we can construct the matrix Q so that its first column is the vector x , i.e., that $Q e_1 = x$. Since λ is a simple eigenvalue of A , it *cannot* be an eigenvalue of B . Thus the matrix $\lambda I - B$ is invertible. Hence $\bar{\lambda} I - B^*$ is invertible, too. Denote $z = (\bar{\lambda} I - B^*)^{-1} h$. Then

$$\bar{\lambda} z - B^* z = h \implies h^* + z^* B = \lambda z^*.$$

Now one can readily verify that

$$[1 \ z^*] Q^* A Q = \lambda [1 \ z^*] \implies [1 \ z^*] Q^* A = \lambda [1 \ z^*] Q^*.$$

Denote $w^* = [1 \ z^*] Q^*$. The above equation now takes a short form

$$w^* A = \lambda w^*.$$

Hence w is a left eigenvector of A . By the simplicity of λ , the vector w is a nonzero multiple of y . However, observe that

$$w^* x = [1 \ z^*] Q^* Q e_1 = [1 \ z^*] e_1 = 1 (\neq 0),$$

which proves the lemma. □

15.19 Theorem

Let $A \in \mathbb{C}^{n \times n}$ have a simple eigenvalue λ with right and left unit eigenvectors x and y , respectively. Let $E \in \mathbb{C}^{n \times n}$ be another matrix such that $\|E\|_2 = 1$. For small ε , denote by $\lambda(\varepsilon)$, $x(\varepsilon)$, and $y(\varepsilon)$ the eigenvalue, the right and left unit eigenvectors of the matrix $A + \varepsilon E$ that correspond to λ , x and y when $\varepsilon = 0$. Then

$$|\lambda'(0)| \leq \frac{1}{|y^* x|}$$

Proof. We note that the existence and differentiability of $\lambda(\varepsilon)$, $x(\varepsilon)$, and $y(\varepsilon)$ follow from the inverse function theorem; we leave out the details. Now we write the equation

$$(A + \varepsilon E)x(\varepsilon) = \lambda(\varepsilon)x(\varepsilon)$$

and differentiate it with respect to ε :

$$Ax'(\varepsilon) + Ex(\varepsilon) + \varepsilon Ex'(\varepsilon) = \lambda'(\varepsilon)x + \lambda x'(\varepsilon)$$

Setting $\varepsilon = 0$ gives

$$Ax'(0) + Ex = \lambda'(0)x + \lambda x'(0).$$

Now we premultiply this equation by the vector y^* :

$$y^*Ax'(0) + y^*Ex = \lambda'(0)y^*x + \lambda y^*x'(0).$$

Since $y^*A = \lambda y^*$, the two terms with $x'(0)$ cancel each other out, and we arrive at

$$y^*Ex = \lambda'(0)y^*x.$$

Now we use the Cauchy-Schwarz inequality and other standard inequalities:

$$|\lambda'(0)| |y^*x| = |y^*Ex| = |\langle Ex, y \rangle| \leq \|Ex\|_2 \|y\|_2 \leq \|E\|_2 \|x\|_2 \|y\|_2 = 1.$$

This proves the theorem. Note that $y^*x \neq 0$ by Lemma 15.18. □

We regard the matrix $A + \varepsilon E$ as the perturbation of A “in the direction” of E . If the matrix E is known, one can find $\lambda'(0)$ exactly:

$$\lambda'(0) = \frac{y^*Ex}{y^*x}.$$

Knowing $\lambda'(0)$ would be useful, because by Taylor expansion

$$\lambda(\varepsilon) = \lambda + \frac{y^*Ex}{y^*x}\varepsilon + O(\varepsilon^2)$$

we could have a fairly good approximation to $\lambda(\varepsilon)$ for small ε . In practice, however, the perturbation matrix E is totally unknown, we only know the order of magnitude of perturbation, i.e., the value of ε . Hence we cannot compute an approximation to $\lambda(\varepsilon)$ by Taylor expansion, we can only use the upper bound of Theorem 15.19 to estimate how far $\lambda(\varepsilon)$ can be from λ :

$$|\lambda(\varepsilon) - \lambda(0)| \leq \frac{\varepsilon}{|y^*x|} + O(\varepsilon^2).$$

This shows that the *sensitivity* of the eigenvalue λ to small perturbations of the matrix A is determined by the value of $\frac{1}{|y^*x|}$.

15.20 Condition number of a simple eigenvalue

Let λ be a simple eigenvalue (this means that its algebraic multiplicity is equal to one) of a matrix $A \in \mathbb{C}^{n \times n}$ and x, y the corresponding right and left unit eigenvectors. Then

$$K(\lambda) = \frac{1}{|y^*x|}$$

is called the *condition number* of the eigenvalue λ .

15.21 Remarks

- (a) The condition number $K(\lambda)$ is well defined for every simple eigenvalue. Indeed, by Lemma 15.18 we have $|y^*x| \neq 0$. Also, the value of $|y^*x|$ does not depend on the particular choice of x and y , because $|(ay)^*(bx)| = |ab| |y^*x| = |y^*x|$ for any scalars a and b such that $|a| = |b| = 1$.
- (b) If λ is a multiple eigenvalue, the condition number $K(\lambda)$ is *not* defined, and for a good reason. If the geometric multiplicity of λ is ≥ 2 , then for *any* right eigenvector x there is a left eigenvector y such that $y^*x = 0$ (Exercise 15.4), which would result in $K(\lambda) = \infty$.
- (c) Even if the geometric multiplicity of λ equals one, we may have a pathological situation where $y^*x = 0$. Example: $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ with a multiple eigenvalue $\lambda = 0$; here the right eigenvector is $x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and the left eigenvector is $y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

15.22 Properties of condition numbers for simple eigenvalues

- (a) $K(\lambda) \geq 1$.
- (b) If the matrix A is normal, then $K(\lambda) = 1$ for all its simple eigenvalues.
- (c) Conversely, if a matrix A has all simple eigenvalues with $K(\lambda) = 1$ for each of them, then A is normal.

Proof. By Cauchy-Schwarz inequality, $|y^*x| \leq \|x\|_2 \|y\|_2 = 1$. This implies (a). For the proofs of (b) and (c), see Exercises 15.2 and 15.3.

15.23 Relation to Schur decomposition

We see that matrices with the most well-conditioned eigenvalues (such that $K(\lambda) = 1$ for all the eigenvalues) are normal matrices. On the other hand, normal matrices are characterized by the fact that their Schur decomposition $Q^*AQ = T$ results in an upper-triangular matrix T that is actually diagonal (i.e., its off-diagonal components are zeros). One can expect that if the Schur matrix T is *nearly diagonal* (i.e., its off-diagonal components are small), then the eigenvalues of A are well-conditioned. On the contrary, if some off-diagonal components of T are large, then at least some eigenvalues are ill-conditioned.

15.24 Multiple and ill-conditioned eigenvalues

Remarks 15.21 (b) and (c) show that if we attempt to define the condition number for a multiple eigenvalue λ , we are bound to get $K(\lambda) = \infty$. Not surprisingly, if a simple eigenvalue λ is ill-conditioned (i.e., $K(\lambda) \approx \infty$), then it is ‘nearly multiple’. The following theorem makes this statement precise.

15.25 Theorem (without proof)

Let λ be a simple eigenvalue. Then there is a matrix E such that

$$\frac{\|E\|_2}{\|A\|_2} \leq \frac{1}{\sqrt{K(\lambda)^2 - 1}}$$

and λ is a *multiple* eigenvalue of $A + E$.

(This implies that a simple but ill-conditioned eigenvalue λ becomes multiple under a small perturbation of the matrix A .)

We conclude this chapter by two important theorems (first and second Gershgorin theorems).

Recall that the eigenvalues of a diagonal matrix $D = \text{diag}\{d_1, \dots, d_n\}$ are its diagonal components d_1, \dots, d_n . Naturally, under small perturbations of D its eigenvalues should remain close to d_1, \dots, d_n . More precisely, if E is a small matrix, then the eigenvalues of $A = D + E$ should be close to d_1, \dots, d_n .

But how close? This is the subject of the next two theorems.

15.26 First Gershgorin theorem

Let $D = \text{diag}\{d_1, \dots, d_n\}$ be a diagonal matrix and $E = (e_{ij})$ any matrix. Then every eigenvalue of $A = D + E$ lies in at least one of the closed disks

$$\mathcal{D}_i = \left\{ z \in \mathbb{C} : |z - d_i| \leq \sum_{j=1}^n |e_{ij}| \right\}$$

The closed disks \mathcal{D}_i are called *Gershgorin disks*.

Proof. Let λ be an eigenvalue of A and $x = [x_1, \dots, x_n]^T$ the corresponding eigenvector. Let x_r be the largest (in absolute value) component of x , i.e.,

$$|x_r| = \max\{|x_1|, \dots, |x_n|\}.$$

Since $x \neq 0$, we have $x_r \neq 0$.

Next we use the equation $Ax = \lambda x$. The r -th components of these two vectors are

$$(Ax)_r = d_r x_r + \sum_{j=1}^n e_{rj} x_j \quad \text{and} \quad (\lambda x)_r = \lambda x_r$$

thus

$$d_r x_r + \sum_{j=1}^n e_{rj} x_j = \lambda x_r.$$

Dividing by x_r gives

$$d_r + \sum_{j=1}^n e_{rj} u_j = \lambda$$

where $u_j = x_j/x_r$. Note that $|u_j| \leq 1$ for every $j = 1, \dots, n$. Therefore

$$|\lambda - d_r| = \left| \sum_{j=1}^n e_{rj} u_j \right| \leq \sum_{j=1}^n |e_{rj}| |u_j| \leq \sum_{j=1}^n |e_{rj}|$$

The theorem is proved. □

Theorem 15.26 says that every eigenvalue λ of A belongs to *one of* the Gershgorin disks. A common misconception is that every Gershgorin disk must contain an eigenvalue of A . This is *not* always true. (Example: $A = \begin{bmatrix} 0 & 4 \\ 1 & 0 \end{bmatrix}$.)

If the Gershgorin disks are *disjoint*, i.e., $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ for $i \neq j$, then indeed each \mathcal{D}_i contains exactly one eigenvalue of A . However if the disks overlap, the eigenvalues may “travel” from one disk to another, so that some disks contain two or more eigenvalues while others contain none.



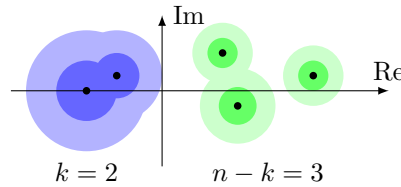
The second Gershgorin theorem describes the distribution of the eigenvalues of A between the disks more precisely.

15.27 Second Gershgorin theorem

Suppose the union of some k Gershgorin disks $\mathcal{D}_{i_1}, \dots, \mathcal{D}_{i_k}$ is disjoint from the union of the other $n - k$ Gershgorin disks, i.e.,

$$\left(\bigcup_{i \in \{i_1, \dots, i_k\}} \mathcal{D}_i\right) \cap \left(\bigcup_{i \notin \{i_1, \dots, i_k\}} \mathcal{D}_i\right) = \emptyset.$$

Then there are exactly k eigenvalues of A (counting multiplicity) in the former union and $n - k$ in the latter.



Proof. For brevity, denote for every $i = 1, \dots, n$

$$h_i = \sum_{j=1}^n |e_{ij}|$$

and consider the matrix $A(s) = D + sE$ for $0 \leq s \leq 1$. We will use (without proof) the following standard fact in complex analysis: the roots of a complex polynomial change continuously with its coefficients. This fact implies that the eigenvalues of the matrix $A(s)$ depend continuously on s .

The Gershgorin disks $\mathcal{D}_i(s)$ for the matrix $A(s)$ have centers d_1, \dots, d_n (the same for all $0 \leq s \leq 1$) and radii sh_1, \dots, sh_n that grow as s increases from 0 to 1. When $s = 0$, each Gershgorin disk $\mathcal{D}_i(0)$ is just a point, d_i , which is an eigenvalue of the matrix $A(0) = D$. Thus the statement of the theorem is true for the matrix $A(0)$. For $0 < s < 1$, each Gershgorin disk $\mathcal{D}_i(s)$ of the matrix $A(s)$ lies inside the corresponding disk $\mathcal{D}_i = \mathcal{D}_i(1)$ of the matrix $A = A(1)$. Therefore

$$\left(\bigcup_{i \in \{i_1, \dots, i_k\}} \mathcal{D}_i(s)\right) \cap \left(\bigcup_{i \notin \{i_1, \dots, i_k\}} \mathcal{D}_i(s)\right) = \emptyset.$$

for all $0 \leq s \leq 1$. Due to the continuity, the eigenvalues of $A(s)$ cannot “jump” from one union to the other and vice versa. Therefore the statement of the theorem remains valid for the matrix $A(s)$ for all $0 \leq s \leq 1$. Lastly recall that $A(1) = A$. \square

Exercise 15.1. (JPE May, 1994). Let $X^{-1}AX = D$, where D is a diagonal matrix.

- (i) Show that the columns of X are right eigenvectors and the conjugate rows of X^{-1} are left eigenvectors of A .
- (ii) Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of A . Show that there are right eigenvectors x_1, \dots, x_n and left eigenvectors y_1, \dots, y_n such that

$$A = \sum_{i=1}^n \lambda_i x_i y_i^*$$

Exercise 15.2. Let $A \in \mathbb{C}^{n \times n}$. Show that

- (i) λ is an eigenvalue of A iff $\bar{\lambda}$ is an eigenvalue of A^* .
- (ii) if A is normal, then for each eigenvalue the left and right eigenspaces coincide;
- (iii) if A is normal, then for any simple eigenvalue λ of A we have $K(\lambda) = 1$.

Exercise 15.3. Let $A \in \mathbb{C}^{n \times n}$ and $B = Q^*AQ$, where Q is a unitary matrix. Show that if the left and right eigenspaces of A are equal, then B enjoys the same property. After that show that A is normal. Finally, prove that if A has all simple eigenvalues with $K(\lambda) = 1$, then A is normal.

Exercise 15.4. Suppose λ is an eigenvalue of geometric multiplicity ≥ 2 for a matrix A . Show that for each right eigenvector x there is a left eigenvector y such that $y^*x = 0$.

Exercise 15.5. Use the Gershgorin theorems to show that a symmetric, strictly row diagonally dominant real matrix with positive diagonal elements is positive definite.

Exercise 15.6 (Bonus). Let $A \in \mathbb{C}^{n \times n}$ be Hermitian with eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$. Let $\mu_1 \leq \dots \leq \mu_{n-1}$ be all the eigenvalues of the $(n-1)$ -st principal minor A_{n-1} of A . Use the Minimax theorem to prove the *interlacing property*

$$\lambda_1 \leq \mu_1 \leq \lambda_2 \leq \dots \leq \lambda_{n-1} \leq \mu_{n-1} \leq \lambda_n$$

Computation of Eigenvalues: Power Method

We now turn to practical methods for computing eigenvalues of matrices. This chapter presents simple and straightforward *power method*, and the next (and last) chapter – a more complicated one, called *QR algorithm*.

16.1 Preface

Here is our basic task: given a matrix $A \in \mathbb{C}^{n \times n}$, compute its eigenvalues (and eigenvectors). According to Chapter 15, there is no finite algorithm (no closed formula) for solving this problem. All solutions are necessarily iterative: we will just construct a sequence of numbers $\lambda^{(k)}$, where $k = 1, 2, \dots$, that would converge to an eigenvalue λ of A , i.e., $\lim_{k \rightarrow \infty} \lambda^{(k)} = \lambda$.

Also, our methods will not apply to *all* matrices, they will only work under certain conditions. Fortunately, our methods apply to *most* matrices. More precisely, our methods apply to *typical, generic* matrices; they only fail in exceptional cases (where some “bad accident” or some coincidence occurs). We will have to exclude such cases from our discussion.

First, we will always assume that the matrix A is *diagonalizable*⁴. This assumption implies that there is a basis of eigenvectors. We will denote by $\lambda_1, \dots, \lambda_n$ the eigenvalues of A and the corresponding eigenvectors by x_1, \dots, x_n , which are chosen so that $\{x_1, \dots, x_n\}$ makes a basis in \mathbb{C}^n .

We always number the eigenvalues so that their absolute values decrease:

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|.$$

In other words, λ_1 is the *largest* eigenvalue and λ_n is the *smallest* one.

Second, we assume that $|\lambda_1| > |\lambda_2|$. This excludes a possible coincidence of $|\lambda_1|$ and $|\lambda_2|$ (this coincidence is another exceptional case that we rule out of the discussion). We call λ_1 the *dominant eigenvalue*. Our assumption implies that λ_1 is a simple eigenvalue and its eigenspace $\text{span}\{x_1\}$ is one-dimensional.

⁴Typical matrices are diagonalizable. In particular, matrices whose eigenvalues are *distinct* are always diagonalizable. So we will only have to exclude *some* matrices whose eigenvalues coincide (and coincidence can be regarded as an exceptional case).

16.2 Power method: a general scheme

Let $q \in \mathbb{C}^n$ be any vector. Since there is basis $\{x_1, \dots, x_n\}$ of eigenvectors of A , we can write

$$q = c_1x_1 + \dots + c_nx_n.$$

Note that

$$\begin{aligned} Aq &= c_1Ax_1 + \dots + c_nAx_n \\ &= c_1\lambda_1x_1 + \dots + c_n\lambda_nx_n. \end{aligned}$$

By induction, for any $k \geq 1$ we have

$$A^kq = c_1\lambda_1^kx_1 + \dots + c_n\lambda_n^kx_n.$$

Let us consider vector

$$q^{(k)} = A^kq/\lambda_1^k \tag{16.1}$$

i.e.,

$$q^{(k)} = c_1x_1 + \underbrace{c_2(\lambda_2/\lambda_1)^kx_2 + \dots + c_n(\lambda_n/\lambda_1)^kx_n}_{\Delta_k}. \tag{16.2}$$

The first term c_1x_1 is fixed (it does not change with k), and the other terms converge to zero, because $|\lambda_i/\lambda_1| < 1$ for all $i = 2, \dots, n$. Therefore

$$\lim_{k \rightarrow \infty} q^{(k)} = c_1x_1. \tag{16.3}$$

Moreover, the “remainder” $\Delta_k = q^{(k)} - c_1x_1$ satisfies

$$\|\Delta_k\| = \|q^{(k)} - c_1x_1\| \leq \text{const} \cdot r^k \tag{16.4}$$

where

$$r = |\lambda_2/\lambda_1| < 1.$$

The number r characterizes the speed of convergence $\|\Delta_k\| \rightarrow 0$. More precisely,

$$\|\Delta^{(k)}\|/\|\Delta^{(k-1)}\| \rightarrow r \quad \text{as } k \rightarrow \infty$$

(assuming $c_2 \neq 0$); thus each “error” $\|\Delta^{(k)}\|$ is approximately r times the previous “error” $\|\Delta^{(k-1)}\|$, i.e., the “errors” decrease as a geometric progression with ratio r . For this reason r is called the *convergence ratio* or the *contraction number*. The smaller r , the faster the convergence in (16.3).

16.3 Linear, quadratic and cubic convergence

Let $a_k \in \mathbb{C}$ be a sequence of complex numbers such that $a_k \rightarrow a$ as $k \rightarrow \infty$. We say that the convergence $a_k \rightarrow a$ is *linear* if

$$|a_{k+1} - a| \approx r|a_k - a|$$

for some $0 < r < 1$ and all sufficiently large k . This means that the distance from the current value a_k to the limit (“target”) value a decreases (shrinks) by a factor of r , i.e., as a geometric progression. The convergence in (16.3) is linear, with $r = |\lambda_2/\lambda_1|$. If

$$|a_{k+1} - a| \approx C|a_k - a|^p$$

with some constant $C > 0$ and power $p > 1$, then the convergence is said to be *superlinear*; it is faster than linear. For $p = 2$ the convergence is said to be *quadratic*, and for $p = 3$ it is *cubic*.

16.4 Remarks on convergence

If the convergence is linear, then by induction we have

$$|a_k - a| \approx |a_0 - a| r^k.$$

We see that the “remainder” $|a_k - a|$ decreases as r^k , i.e., exponentially fast, which is very fast by calculus standards. But in numerical calculations standards are different, and such a convergence is *slow*; see below.

Suppose, for example, $r = 0.5$. Then

$$\frac{|a_k - a|}{|a_0 - a|} \approx \frac{1}{2^k}.$$

If a_k denotes the computed approximation to an unknown number a , then its relative error is $\sim 1/2^k$. This means that the first k binary digits in a and a_k coincide, i.e., we can trust k binary digits in a_k . Therefore each iteration adds one accurate binary digits (one bit) to the result. To compute a accurately in single precision, one would need about 22–23 iterations, and in double precision – about 52–53 iterations. This is too much for practical applications. And we picked a fairly small ratio $r = 0.5$. Now imagine how long the process might take if $r = 0.9$ or $r = 0.99$.

Now suppose the convergence is quadratic. Then, if the error at the current iteration is $\frac{1}{2^m}$, it will be $\sim \frac{1}{2^{2m}}$ at the next iteration. Hence if the current approximation carries m accurate binary digits, the next one will carry $2m$. Thus each iteration doubles (!) the number of accurate digits. Starting with just one accurate binary digit, one gets 2, 4, 8, 16, 32 digits at the following iterations. So one needs 4–5 iterations in single precision and only one more iteration in double precision. This is fast enough for practical applications.

Similarly, the cubic convergence means that each iteration triples (!!!) the number of accurate digits. See Example 16.21 for an illustration.

16.5 Scaling problem in the power method

Back to the power method: note that the limit vector c_1x_1 in (16.3) is an eigenvector of A corresponding to λ_1 (unless $c_1 = 0$). Thus we found a sequence of vectors converging to an eigenvector of A .

However, the vector $q^{(k)} = A^k q / \lambda_1^k$ cannot be computed in practice, because we do not know λ_1 in advance (in fact, computing λ_1 is our ultimate goal). On the other hand, we cannot just drop the factor $1/\lambda_1^k$ and use the sequence of vectors $A^k q$. Indeed, $\|A^k q\| \rightarrow \infty$ if $|\lambda_1| > 1$ and $\|A^k q\| \rightarrow 0$ if $|\lambda_1| < 1$, thus our vectors would either grow too large or get too small. This would cause a breakdown (overflow or underflow) in computations. We must somehow normalize, or scale, the vector $A^k q$. To this end we will define a new sequence of vectors recursively: $q_0 = q$ and

$$q_k = Aq_{k-1}/\sigma_k \quad (16.5)$$

for $k \geq 1$, where σ_k is a properly chosen scaling factor; see below. We will also define an approximation $\lambda_1^{(k)}$ to the eigenvalue λ_1 by using the vector q_k .

16.6 First choice for the scaling factor

There are two standard choices for σ_k . One is

$$\sigma_k = \|Aq_{k-1}\|_2. \quad (16.6)$$

This ensures $\|q_k\|_2 = 1$, i.e., all our q_k 's will be unit vectors. Thus they will not grow too large or too small.

Now one can approximate the eigenvalue λ_1 by the Rayleigh quotient

$$\lambda_1^{(k)} = q_k^* A q_k. \quad (16.7)$$

We will show that $\lambda_1^{(k)}$ indeed converges to λ_1 , as $k \rightarrow \infty$, and the vectors q_k get closer and closer to the eigenspace $\text{span}\{x_1\}$ corresponding to λ_1 . However, the sequence of vectors q_k need not converge to any specific limit vector, as the following example shows.

16.7 Example

Let $A = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$. Let us choose $q_0 = [1, 1]^T$. It is easy to see that $q_k = [(-1)^k, 0]^T$ and $\lambda_1^{(k)} = -1$ for all $k \geq 1$. Hence $\lambda_1^{(k)}$ coincides with the dominant eigenvalue $\lambda_1 = -1$ of the matrix A , and q_k is a corresponding eigenvector – a perfect result. However, q_k does not converge to a limit, it keeps oscillating (flipping back and forth between two opposite vectors $[1, 0]^T$ and $[-1, 0]^T$).

16.8 Example

Here is a more interesting example. Let

$$A = \begin{bmatrix} -3 & 1 \\ 1 & 1 \end{bmatrix}.$$

This matrix has eigenvalues⁵

$$\lambda_1 = -1 - \sqrt{5} \approx -3.2361$$

$$\lambda_2 = -1 + \sqrt{5} \approx +1.2361$$

Note that λ_1 is the dominant eigenvalue because $|\lambda_1| > |\lambda_2|$. The corresponding eigenvector is $x_1 = [0.9732, -0.2298]^T$.

Let us pick $q_0 = [1, 0]^T$ and use (16.6) for the scaling factor and (16.12) for the approximation to λ_1 . Then $Aq_0 = [-3, 1]^T$, so $q_1 = [-0.9487, 0.3162]^T$ and $\lambda_1^{(1)} = -3.2$.

At the next iteration, $Aq_1 = [3.1623, -0.6325]^T$, so $q_2 = [0.9806, -0.1961]^T$ and $\lambda_1^{(2)} = -3.2308$. The table below summarizes the first five iterations.

Here $\lambda_1^{(k)}$ converges to the dominant eigenvalue λ_1 and q_k gets closer and closer to the eigenspace $\text{span}\{x_1\}$. But q_k does not have a limit – it keeps oscillating between two limit vectors: x_1 and $-x_1$.

	q_{k-1}	Aq_{k-1}	σ_k	$q_k = Aq_{k-1}/\sigma_k$	$\lambda_1^{(k)}$
$k = 1$	[+1.0000, +0.0000]	[-3.0000, +1.0000]	3.1623	[-0.9487, +0.3162]	-3.2000
$k = 2$	[-0.9487, +0.3162]	[+3.1623, -0.6325]	3.2249	[+0.9806, -0.1961]	-3.2308
$k = 3$	[+0.9806, -0.1961]	[-3.1379, +0.7845]	3.2344	[-0.9701, -0.2425]	-3.2353
$k = 4$	[-0.9701, -0.2425]	[+3.1530, -0.7276]	3.2358	[+0.9744, -0.2249]	-3.2360
$k = 5$	[+0.9744, -0.2249]	[-3.1480, +0.7495]	3.2360	[-0.9728, +0.2316]	-3.2361
	↓			↓	↓
Limit:	no limit (oscillates)			no limit (oscillates)	-3.2361

⁵We will provide approximate numerical values for all our constants with four digits after the decimal point. The calculations were performed by MATLAB®.

To estimate how close the unit vector q_k is to the one-dimensional eigenspace $\text{span}\{x_1\}$, denote by p_k the orthogonal projection of q_k on $\text{span}\{x_1\}$ and by $d_k = q_k - p_k$ the orthogonal component. Then $\|d_k\|$ measures the distance from q_k to $\text{span}\{x_1\}$.

16.9 Theorem (convergence of the power method)

Assume the following:

- (i) The matrix A is diagonalizable, as described in Section 16.1.
- (ii) λ_1 is the dominant eigenvalue, as described in Section 16.1.
- (iii) $q_0 = c_1x_1 + \cdots + c_nx_n$ is chosen so that $c_1 \neq 0$.

Then the distance from q_k to the eigenspace $\text{span}\{x_1\}$ converges to zero and $\lambda_1^{(k)}$ converges to λ_1 . Furthermore,

$$\|d_k\| \leq \text{const} \cdot r^k \quad (16.8)$$

and

$$|\lambda_1^{(k)} - \lambda_1| \leq \text{const} \cdot r^k \quad (16.9)$$

where $r = |\lambda_2/\lambda_1| < 1$.

Proof. By induction, we have

$$q_k = \frac{Aq_{k-1}}{\sigma_k} = \frac{A^2q_{k-2}}{\sigma_{k-1}\sigma_k} = \cdots = \frac{A^kq_0}{\sigma_1 \cdots \sigma_k}$$

According to (16.1)–(16.2) we have

$$q_k = \frac{\lambda_1^k}{\sigma_1 \cdots \sigma_k} q^{(k)} = \frac{\lambda_1^k}{\sigma_1 \cdots \sigma_k} (c_1x_1 + \Delta_k)$$

i.e., q_k is proportional to $c_1x_1 + \Delta_k$. Note that c_1x_1 is a fixed non-zero vector, $\|\Delta_k\| \rightarrow 0$ by (16.4). Also, $\|q_k\|_2 = 1$ by construction, and $\|c_1x_1 + \Delta_k\| \approx \|c_1x_1\| > 0$, so for all large enough k we have $\frac{1}{2}\|c_1x_1\| < \|c_1x_1 + \Delta_k\| < 2\|c_1x_1\|$. Therefore the scalar $\frac{\lambda_1^k}{\sigma_1 \cdots \sigma_k}$ cannot get too large or too small, i.e.,

$$c_1 < \left| \frac{\lambda_1^k}{\sigma_1 \cdots \sigma_k} \right| < c_2$$

for some positive constants $0 < c_1 < c_2 < \infty$. Now the claim (16.8) can be easily derived from (16.4) (we leave this as an exercise). The last claim (16.9) follows from (16.8) and (15.1). \square

16.10 Second choice for the scaling factor

The other popular choice for σ_k is the largest (in absolute value) component of the vector Aq_{k-1} , i.e.,

$$\sigma_k = (Aq_{k-1})_m \quad \text{so that} \quad |(Aq_{k-1})_m| = \max_i |(Aq_{k-1})_i|. \quad (16.10)$$

According to (16.5), the largest (in absolute value) component of q_k will be now equal to one, i.e.,

$$(q_k)_m = 1 \quad \text{and} \quad \max_i |(q_k)_i| \leq 1 \quad (16.11)$$

This implies $1 \leq \|q_k\|_2 \leq \sqrt{n}$, so again our vectors q_k will not grow too large or get too small. As in Section 16.6, one can approximate the eigenvalue λ_1 by the Rayleigh quotient

$$\lambda_1^{(k)} = \frac{q_k^* A q_k}{q_k^* q_k}. \quad (16.12)$$

Theorem 16.9 applies to this new procedure word for word. Its proof only needs one minor change: instead of $\|q_k\|_2 = 1$ we now have $1 \leq \|q_k\|_2 \leq \sqrt{n}$, but this change does not affect the validity of the argument.

Since for large k the vector q_k will be almost an eigenvector of A , we will get $Aq_k \approx \lambda_1 q_k$. Therefore the largest component of Aq_k will be $\approx \lambda_1$. On the other hand, the largest component of Aq_k will be our σ_{k+1} , according to (16.10). Thus the scaling factor itself can be used as an approximation to λ_1 , i.e., we can set

$$\tilde{\lambda}_1^{(k)} = \sigma_k. \quad (16.13)$$

This approximation to λ_1 works well in typical cases. More precisely, it works if the eigenvector x_1 has *one* component with the largest absolute value, i.e.,

$$\exists m \in \{1, \dots, n\}: \max_{i \neq m} |(x_1)_i| < |(x_1)_m|. \quad (16.14)$$

Then if k is large enough, q_k will be almost proportional to x_1 , hence the largest component of q_k will be $(q_k)_m$, where m is the same as in (16.14), i.e.,

$$\max_{i \neq m} |(q_k)_i| < |(q_k)_m|.$$

Thus the index m in (16.10)–(16.11) will be the same for all large k , and therefore (16.13) will give a good approximation to $\lambda_1^{(k)}$; see the next example.

16.11 Example

We continue Example 16.8. We work with the same matrix $A = \begin{bmatrix} -3 & 1 \\ 1 & 1 \end{bmatrix}$ and pick the same initial vector $q_0 = [1, 0]^T$. But now we use (16.10) for the scaling factor and (16.13) for the approximation to λ_1 .

The table below summarizes the first nine iterations. Here $\tilde{\lambda}_1^{(k)}$ converges to the dominant eigenvalue

$$\lambda_1 = -1 - \sqrt{5} \approx -3.2361$$

and q_k converges to the dominant eigenvector

$$[1, 2 - \sqrt{5}]^T \approx [1.0000, -0.2361]^T.$$

	q_{k-1}	Aq_{k-1}	$\sigma_k = \tilde{\lambda}_1^{(k)}$	$q_k = Aq_{k-1}/\sigma_k$
$k = 1$	[+1.0000, +0.0000]	[-3.0000, +1.0000]	-3.0000	[+1.0000, -0.3333]
$k = 2$	[+1.0000, -0.3333]	[-3.3333, +0.6667]	-3.3333	[+1.0000, -0.2000]
$k = 3$	[+1.0000, -0.2000]	[-3.2000, +0.8000]	-3.2000	[+1.0000, -0.2500]
$k = 4$	[+1.0000, -0.2500]	[-3.2500, +0.7500]	-3.2500	[+1.0000, -0.2308]
$k = 5$	[+1.0000, -0.2308]	[-3.2308, +0.7692]	-3.2308	[+1.0000, -0.2381]
$k = 6$	[+1.0000, -0.2381]	[-3.2381, +0.7619]	-3.2381	[+1.0000, -0.2353]
$k = 7$	[+1.0000, -0.2353]	[-3.2353, +0.7647]	-3.2353	[+1.0000, -0.2364]
$k = 8$	[+1.0000, -0.2364]	[-3.2364, +0.7636]	-3.2364	[+1.0000, -0.2360]
$k = 9$	[+1.0000, -0.2360]	[-3.2360, +0.7640]	-3.2360	[+1.0000, -0.2361]
	↓		↓	↓
Limit:	[+1.0000, -0.2361]		-3.2361	[+1.0000, -0.2361]

The choice of the scaling factor (16.10) and the approximation (16.13) have two attractive advantages:

- (a) they are easier to compute than (16.6) and (16.12)
- (b) the vector q_k *does* converge to a limit, as $k \rightarrow \infty$ (unlike our Example 16.7).

Back to Example 16.7: if we use the scaling factor (16.10) and the approximation (16.13), instead of (16.6) and (16.12), then we will get the vector $q_k = [1, 0]$ for all $k \geq 1$, so there will be no oscillations.

The scaling factor (16.10) and the approximation (16.13) work well in typical cases, where the dominant eigenvector x_1 has *one* component with the largest absolute value. However, if the eigenvector x_1 has *more than one* largest component, things can go wrong; see the next example. (Incidentally, our example shows that the JPE problem #8 in Fall 2012 is stated incorrectly.)

16.12 Example

Let $A = \begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix}$. This is a symmetric matrix with eigenvalues $\lambda_1 = 3$ and $\lambda_2 = -1$; the corresponding eigenvectors are $x_1 = [1, -1]^T$ and $x_2 = [1, 1]^T$. Note that the two components of x_1 have equal absolute values. Let us choose $q_0 = [1, 0]^T$. Then $Aq_0 = [1, -2]^T$, and our rules (16.10) and (16.13) give

$$\tilde{\lambda}_1^{(1)} = \sigma_1 = -2$$

At the next iteration we get

$$q_1 = Aq_0/\sigma_1 = [-0.5, 1]^T$$

and $Aq_1 = [-2.5, 2]$, hence

$$\tilde{\lambda}_1^{(2)} = \sigma_2 = -2.5$$

which even worse than $\tilde{\lambda}_1^{(1)}$ (it is farther away from $\lambda_1 = 3$). At the following iteration $q_2 = [1, -0.8]^T$ and $Aq_2 = [2.6, -2.8]^T$, so that

$$\tilde{\lambda}_1^{(3)} = \sigma_3 = -2.8$$

which is still worse than $\tilde{\lambda}_1^{(2)}$. In this example $\tilde{\lambda}_1^{(k)}$ converges to the wrong limit:

$$\tilde{\lambda}_1^{(k)} \rightarrow -3 \text{ as } k \rightarrow \infty, \quad \text{while } \lambda_1 = 3.$$

The vectors q_k get closer and closer to the eigenspace $\text{span}\{x_1\}$ corresponding to λ_1 , but they do *not* converge to any specific limit, they oscillate between two limit points: $[1, -1]^T$ and $[-1, 1]^T$.

	q_{k-1}	Aq_{k-1}	$\sigma_k = \tilde{\lambda}_1^{(k)}$	$q_k = Aq_{k-1}/\sigma_k$
$k = 1$	[+1.0000, +0.0000]	[+1.0000, -2.0000]	-2.0000	[-0.5000, +1.0000]
$k = 2$	[-0.5000, +1.0000]	[-2.5000, +2.0000]	-2.5000	[+1.0000, -0.8000]
$k = 3$	[+1.0000, -0.8000]	[+2.6000, -2.8000]	-2.8000	[-0.9286, +1.0000]
$k = 4$	[-0.9286, +1.0000]	[-2.9286, +2.8571]	-2.9286	[+1.0000, -0.9756]
$k = 5$	[+1.0000, -0.9756]	[+2.9512, -2.9756]	-2.9756	[-0.9918, +1.0000]
	↓		↓	↓
Limit:	no limit (oscillates)		-3.0000 (wrong limit)	no limit (oscillates)

We should admit that the matrix A in the above example is somewhat exceptional: in practice it rarely happens that the dominant eigenvector x_1 has two or more equally large components.

16.13 Initial vector

The initial vector q_0 only has to fulfill the requirement $c_1 \neq 0$. Unacceptable vectors (those with $c_1 = 0$) form a hyperplane in \mathbb{C}^n defined by

$$H_{\text{bad}} = \text{span} \{x_2, \dots, x_n\}$$

A hyperplane is a “tiny” part of the space \mathbb{C}^n , so if we pick a vector q_0 “randomly”, it will most likely be *not* in that bad hyperplane.

Even if we have a bad luck of choosing a vector $q_0 \in H_{\text{bad}}$ accidentally, something else may come to the rescue. All the vectors q_k , $k \geq 1$, must lie in H_{bad} , theoretically, but due to round-off errors the numerically computed vectors q_k will very likely drift away from that hyperplane, and then they would approach the dominant eigenspace $\text{span}\{x_1\}$, as desired.

If that does not seem to be enough, we can apply the following strategy that guarantees an absolute safety. We should choose not just one, but n different initial vectors that make a basis in \mathbb{C}^n (for instance, we can choose e_1, \dots, e_n). Then we carry out the power method starting with each of these initial vectors, i.e., carry out the power method n times, each time with a different initial vector, and run it until convergence. Since our initial vectors make a basis, one of them has to be away from the hyperplane H_{bad} , so we are bound to converge to the dominant eigenvalue λ_1 at least once.

16.14 Aiming at the smallest eigenvalue

We spent so much effort on developing methods for computing the *dominant* eigenvalue λ_1 . But what about the other eigenvalues, $\lambda_2, \dots, \lambda_n$? Our next goal is to compute the *smallest* eigenvalue λ_n . Again, we need certain assumptions on the matrix A .

First, we keep our main assumption made in Section 16.1: the matrix A is diagonalizable. We denote its eigenvalues by $\lambda_1, \dots, \lambda_n$ and order them so that

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|.$$

We still denote by x_1, \dots, x_n corresponding eigenvectors that make a basis in \mathbb{C}^n .

In addition, we now make two new assumptions:

- (i) The matrix A is invertible (this implies $\lambda_n \neq 0$).
- (ii) The matrix A has *one* smallest eigenvalue, i.e., $|\lambda_n| < |\lambda_{n-1}|$.

Both new assumptions rule out certain exceptional matrices, so all typical matrices A satisfy our assumptions. We also note that if assumption (i) fails, then $\lambda_n = 0$, and we do not need an elaborate algorithm to compute it.

16.15 Inverse power method

The inverse matrix A^{-1} has eigenvalues $\lambda_1^{-1}, \dots, \lambda_n^{-1}$ and the same corresponding eigenvectors x_1, \dots, x_n . Note that $|\lambda_1^{-1}| \leq \dots \leq |\lambda_n^{-1}|$. Our assumption (ii) implies $|\lambda_n^{-1}| > |\lambda_{n-1}^{-1}|$, hence the matrix A has *one* dominant eigenvalues, λ_n^{-1} , with eigenvector x_n .

Thus the matrix A^{-1} satisfies our assumptions made in Section 16.1, so that one can apply the power method to A^{-1} . This gives, of course, its dominant eigenvalue, which is λ_n^{-1} , and the corresponding eigenspace $\text{span}\{x_n\}$. The reciprocal of the former will be the smallest eigenvalue λ_n of A . This trick is called *inverse power method*, because it consists of the power method applied to the inverse matrix A^{-1} .

According to Theorem 16.9, the inverse power method converges linearly, with the ratio

$$r = |\lambda_{n-1}^{-1}/\lambda_n^{-1}| = |\lambda_n/\lambda_{n-1}| < 1. \quad (16.15)$$

16.16 Practical implementation

The inverse power method goes through the same steps as the power method does, except A is replaced with A^{-1} . In particular, the recurrent formula (16.5) now takes form

$$q_k = A^{-1}q_{k-1}/\sigma_k. \quad (16.16)$$

In manual calculations (on paper), one just computes A^{-1} and uses it.

However, in numerical calculations done by computer, the construction of the inverse matrix A^{-1} is too expensive and should be avoided. Fortunately, it can be avoided in the present situation. All we need is to compute vectors $A^{-1}q_0, A^{-1}q_1, \dots$, according to (16.16). In order to compute the product $A^{-1}b$ for any given vector $b \in \mathbb{C}^n$, one can just solve the system of linear equations $Ax = b$, and its solution will be the desired vector $x = A^{-1}b$. The system $Ax = b$ can be solved by the LU decomposition of the matrix A , as described in Chapter 7. This trick is routinely used in the computer implementation of the inverse power method.

Now we know how to compute the largest and the smallest eigenvalues of a given matrix A . Now we turn to the task of computing any eigenvalue.

16.17 Aiming at any eigenvalue

Recall that for any eigenvalue λ of A with eigenvector x we have $Ax = \lambda x$. Thus for any scalar $\rho \in \mathbb{C}$ we have

$$(A - \rho I)x = (\lambda - \rho)x$$

Therefore $\lambda - \rho$ is an eigenvalue of the matrix $A - \rho I$, with the same eigenvector x .

Recall that, in our notation, the given matrix A has eigenvalues $\lambda_1, \dots, \lambda_n$ with eigenvectors x_1, \dots, x_n . Now the matrix $A - \rho I$ has eigenvalues

$$\lambda_1 - \rho, \lambda_2 - \rho, \dots, \lambda_n - \rho$$

with the same eigenvectors x_1, x_2, \dots, x_n . We see that the whole spectrum (the set of all eigenvalues) of A is *shifted* by subtracting a constant scalar ρ .

If we can compute an eigenvalue λ' of the new matrix $A' = A - \rho I$, then $\lambda' + \rho$ would be an eigenvalue of A . The corresponding eigenvector x' of $A' = A - \rho I$ would be also an eigenvector of A .

Now suppose we want to compute an eigenvalue λ_i of the given matrix A , which is neither the largest nor the smallest one. Then the power method or the inverse power method would not be able to find λ_i . But it is possible that for some scalar $\rho \in \mathbb{C}$ and for the new matrix $A' = A - \rho I$, the corresponding eigenvalue $\lambda' = \lambda_i - \rho$ will be either the largest or the smallest one, i.e.,

$$|\lambda_i - \rho| = \max_{1 \leq j \leq n} |\lambda_j - \rho| \quad (16.17)$$

or

$$|\lambda_i - \rho| = \min_{1 \leq j \leq n} |\lambda_j - \rho|. \quad (16.18)$$

Then we can apply the power method, or the inverse power method, to A' , compute λ' , and lastly determine $\lambda_i = \lambda' + \rho$.

It remains to find a scalar ρ so that either (16.17) or (16.18) holds. A moment's thought reveals that it may be possible to ensure (16.17), but there are examples where it is not possible. On the other hand, it is *always possible* to ensure (16.18) – one just needs to choose a number ρ which is *closer* to the desired eigenvalue λ_i than to any other eigenvalue λ_j .

In other words, we need just a rough approximation to the desired eigenvalue λ_i , just a number ρ that is closer to λ_i than to any other eigenvalue λ_j . In most practical applications, this is not a difficult task.

16.18 Power method with shift

Assume that ρ is a rough approximation to an eigenvalue λ_i of A , i.e., ρ is closer to λ_i than to any other eigenvalue λ_j of A :

$$|\lambda_i - \rho| < |\lambda_j - \rho| \quad \forall j \neq i.$$

Then the matrix $A' = A - \rho I$ will have the smallest eigenvalue $\lambda' = \lambda_i - \rho$ with the eigenvector $x' = x_i$.

We can apply the inverse power method to $A' = A - \rho I$. It will give us the smallest eigenvalue λ' and a corresponding eigenvector x' of the matrix A' . Then we can determine $\lambda_i = \lambda' + \rho$ and $x_i = x'$, i.e., the desired eigenvalue and eigenvector of A . This trick is called the *power method with shift*, because it is based on shifting the eigenvalues of A by a conveniently chosen scalar $\rho \in \mathbb{C}$. That scalar is called the *shift*.

According to Theorem 16.9 and (16.15), the power method with shift ρ converges linearly, with the ratio

$$r = \frac{|\lambda_i - \rho|}{\min_{j \neq i} |\lambda_j - \rho|} < 1. \quad (16.19)$$

16.19 Note on the speed of convergence

The speed of convergence of the power method depends on how big or small the ratio r is; see Section 16.4. If r is small (close to zero), the convergence is actually pretty fast⁶. So it is in our best interests to select the scalar ρ so that r will be small. Fortunately, this is often possible.

The formula (16.19) suggests that if we keep changing ρ so that $\rho \rightarrow \lambda_i$, then the numerator and denominator of (16.19) have the following limits:

$$\left. \begin{array}{l} |\lambda_i - \rho| \rightarrow 0 \\ \min_{j \neq i} |\lambda_j - \rho| \rightarrow \min_{j \neq i} |\lambda_j - \lambda_i| > 0 \end{array} \right\} \implies r \rightarrow 0.$$

Thus, the better ρ approximates λ_i , the smaller r is, and the faster the convergence will be. We will explore this idea in the next section.

⁶For example, if $r = 0.1$, then each iteration adds one accurate *decimal* digit; hence, in single precision, 6–7 iterations might give us maximum accuracy. If $r = 0.01$, then each iteration adds *two* accurate decimal digits, i.e., 6–7 iterations might suffice in double precision.

16.20 Power method with Rayleigh quotient shift

As it turns out, we do not have to work hard to provide a good approximation ρ to λ_i that would accelerate the convergence – the power method itself can do this for us.

Recall that at each iteration we find a vector q_k that gets closer and closer to the eigenspace $\text{span}\{x_i\}$ corresponding to the eigenvalue λ_i of the matrix A . Thus the corresponding Rayleigh quotient $\frac{q_k^* A q_k}{q_k^* q_k}$ would get closer and closer to λ_i . Why not to use *this* Rayleigh quotient as the best available approximation to λ_i ? In other words, at every iteration we can reset (update) the shift ρ by using the Rayleigh quotient of the vector q_k obtained at the previous iteration.

This suggests the following scheme. First, one chooses an initial vector q_0 and an initial shift ρ_0 (which does not have to be a very good approximation to λ_i). Then for each $k \geq 1$ one computes

$$q_k = \frac{(A - \rho_{k-1}I)^{-1}q_{k-1}}{\sigma_k} \quad (16.20)$$

where σ_k is a convenient scaling factor, for example, $\sigma_k = \|(A - \rho_{k-1}I)^{-1}q_{k-1}\|$, and one updates the shift by

$$\rho_k = \frac{q_k^* A q_k}{q_k^* q_k}$$

This brilliant scheme is known as *power method with Rayleigh quotient shift*.

Its convergence is, generally, quadratic (better than linear). If the matrix A is Hermitian, the convergence is even faster – it is cubic!!!

16.21 Example

Consider the symmetric matrix

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 4 \end{bmatrix}$$

and let $q_0 = (1, 1, 1)^T$ be the initial vector and $\rho_0 = 5$ the initial shift. When power method with Rayleigh quotient shift is applied to A , the following values ρ_k are computed at the first two iterations:

$$\rho_1 = 5.2131, \quad \rho_2 = 5.214319743184$$

The actual value is $\lambda = 5.214319743377$. After only two iterations, the method produces 10 accurate digits. The next iteration would bring about 30 accurate digits – more than enough for double precision arithmetic.

16.22 Power method: pros and cons

- (a) The power method is classic. Its logic is simple, and the method generally works well. The convergence is quadratic, and for Hermitian matrices the convergence is cubic.
- (b) The power method is still very slow for two reasons. The main one is that each iteration is rather expensive. Indeed, according to (16.20) we need to compute $(A - \rho_{k-1}I)^{-1}q_{k-1}$, i.e., solve a system of linear equations $(A - \rho_{k-1}I)x = q_{k-1}$. This can be done by LU decomposition of the matrix $A - \rho_{k-1}I$, which keeps changing at every iteration (because of ρ_{k-1}). Hence the LU decomposition must be performed anew at every iteration, and each LU decomposition takes $\approx 2n^3/3$ flops (Section 7.10). Thus the cost of computing one eigenvalue is $\approx 2pn^3/3$, where p is the number of iterations (typically, p is about 5 or 10).

The other reason is that each eigenvalue must be computed independently – we need to choose an initial vector q_0 and an initial shift ρ_0 , and then run the inverse power method until its iterations converge. Thus, if we want to compute all n eigenvalues of A , the total computational cost will be $2pn^4/3$ flops. This is quite a big number.

- (c) Lastly, if the matrix A is real (and this is the most common case in practice), then in order to compute its non-real complex eigenvalues $\lambda_i \notin \mathbb{R}$ one has to use complex shifts $\rho_k \notin \mathbb{R}$. Thus one has to deal with complex matrices $(A - \rho_{k-1}I)$, which is inconvenient and expensive.

It would be nice to operate with real matrices only, for as long as possible, and obtain pairs of complex conjugate eigenvalues only at the very last step. The QR algorithm, to be discussed in the next chapter, provides such a luxury.

Exercise 16.1. (JPE, May 2003) Let A be a symmetric matrix with eigenvalues such that $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_{n-1}| > |\lambda_n|$. Suppose $z \in \mathbb{R}^n$ with $z^T x_1 \neq 0$, where $Ax_1 = \lambda_1 x_1$. Prove that, for some constant C ,

$$\lim_{k \rightarrow \infty} \frac{A^k z}{\lambda_1^k} = Cx_1$$

and use this result to devise a reliable algorithm for computing λ_1 and x_1 . Explain how the calculation should be modified to obtain (a) λ_n and (b) the eigenvalue closest to 2.

Exercise 16.2. (JPE, September 1996) The matrix

$$A = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix}$$

has eigenpairs

$$(\lambda, x) = \left(2, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right), \left(-1, \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \right), \left(3, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right),$$

Suppose the power method is applied with starting vector

$$z_0 = [1, 1, -1]^t / \sqrt{3}$$

- (a) Determine whether or not the iteration will converge to an eigenpair of A , and if so, which one. Assume exact arithmetic.
- (b) Repeat (a), except now use the inverse iteration with the same starting vector z_0 and the Rayleigh quotient of z_0 as approximation for the eigenvalue.
- (c) Now answer both (a) and (b) again, except this time use standard fixed precision floating point arithmetic, i.e. computer arithmetic.

Computation of Eigenvalues: QR Algorithm

In this last chapter of the course we present the QR algorithm for computing eigenvalues of any matrix $A \in \mathbb{C}^{n \times n}$. The QR algorithm (not to be confused with QR decomposition!) dates back to the early 1960s, and in the recent decades it became the most widely used method for calculating eigenvalues.

17.1 “Pure” QR algorithm

Let $A \in \mathbb{C}^{n \times n}$. The algorithm starts with $A_0 = A$ and generates a sequence of matrices A_k defined as follows:

$$\begin{aligned} A_{k-1} &= Q_k R_k && \text{(QR decomposition)} \\ A_k &= R_k Q_k && \text{(reverse multiplication)} \end{aligned}$$

That is, a QR decomposition of the current matrix A_{k-1} is computed first, and then its factors are multiplied together in the reverse order to produce the next matrix A_k . One iteration of the QR algorithm is called *QR step*. It consists of two substeps: (i) QR decomposition and (ii) reverse multiplication.

17.2 Two basic facts

(i) All matrices A_k in the QR algorithm are unitary equivalent.

Proof. Indeed, $A_k = R_k Q_k = Q_k^{-1} A_{k-1} Q_k$. □

As a result, all matrices A_k have the same eigenvalues. They also have the same 2-condition number, ensuring that the QR algorithm is numerically stable.

(ii) If A is a Hermitian matrix, then all A_k 's are Hermitian matrices as well.

Proof. This follows from (i) and Section 3.10. □

The fact (b) ensures that the important Hermitian property is not lost during the QR procedure.

Note that $\|R_k\|_F = \|A_k\|_F = \|A\|_F$ for all $k \geq 1$ (Exercise 2.2). Thus all our matrices A_k and R_k have the same *mass* (or *energy*); see Section 5.9. At the first part of each QR step (the QR decomposition of the matrix A_{k-1}), the entire mass is pulled up to the upper triangle. At the second step (the reverse multiplication of R_k and Q_k), the mass tends to stay on top of A_k , even though some of it “leaks back” to the bottom. This is illustrated by the following example, where for simplicity both R_k and Q_k are filled with identical numbers:

$$\begin{array}{ccc} \begin{bmatrix} 2 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix} & \times & \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\ R_k & \times & Q_k = A_k \end{array}$$

As k grows, successive repetitions of the QR step strengthen the effect of pulling the mass upward. These considerations make the following theorem plausible (though its formal proof is beyond the scope of the course):

17.3 Theorem (convergence of the QR algorithm)

Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of A satisfying

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n| > 0$$

Under one technical condition⁷, the matrix $A_k = (a_{ij}^{(k)})$ is guaranteed to converge to an upper triangular form, so that

- (a) $a_{ij}^{(k)} \rightarrow 0$ as $k \rightarrow \infty$ for all $i > j$.
- (b) $a_{ii}^{(k)} \rightarrow \lambda_i$ as $k \rightarrow \infty$ for all i .

The convergence is linear, with the ratio

$$r = \max_k |\lambda_{k+1}/\lambda_k| < 1.$$

Note that $a_{ij}^{(k)}$ for $i < j$ need not converge to a limit, as $k \rightarrow \infty$ (see example below). This is why we say that A_k converges to “an upper triangular form”, but not necessarily to any particular upper triangular matrix.

⁷The technical condition is this: the matrix Y whose i -th row is a left eigenvector of A corresponding to λ_i , for all $1 \leq i \leq n$, must have an LU decomposition (i.e., all its principal minors must be nonsingular; see Chapter 7).

17.4 Example

Here we apply the QR algorithm to the matrix $A = \begin{bmatrix} 3 & 2 \\ 1 & 1 \end{bmatrix}$. The calculations were performed with MATLAB®, and the QR decomposition was done by the standard MATLAB function `qr`.

	A_{k-1}	Q_k	R_k	$A_k = R_k Q_k$
$k = 1$	$\begin{bmatrix} +3.0000 & +2.0000 \\ +1.0000 & +1.0000 \end{bmatrix}$	$\begin{bmatrix} -0.9487 & -0.3162 \\ -0.3162 & +0.9487 \end{bmatrix}$	$\begin{bmatrix} -3.1623 & -2.2136 \\ 0 & +0.3162 \end{bmatrix}$	$\begin{bmatrix} +3.7000 & -1.1000 \\ -0.1000 & +0.3000 \end{bmatrix}$
$k = 2$	$\begin{bmatrix} +3.7000 & -1.1000 \\ -0.1000 & +0.3000 \end{bmatrix}$	$\begin{bmatrix} -0.9996 & +0.0270 \\ +0.0270 & +0.9996 \end{bmatrix}$	$\begin{bmatrix} -3.7014 & +1.1077 \\ 0 & +0.2702 \end{bmatrix}$	$\begin{bmatrix} +3.7299 & +1.0073 \\ +0.0073 & +0.2701 \end{bmatrix}$
$k = 3$	$\begin{bmatrix} +3.7299 & +1.0073 \\ +0.0073 & +0.2701 \end{bmatrix}$	$\begin{bmatrix} -1.0000 & -0.0020 \\ -0.0020 & +1.0000 \end{bmatrix}$	$\begin{bmatrix} -3.7299 & -1.0078 \\ 0 & +0.2681 \end{bmatrix}$	$\begin{bmatrix} +3.7319 & -1.0005 \\ -0.0005 & +0.2681 \end{bmatrix}$
$k = 4$	$\begin{bmatrix} +3.7319 & -1.0005 \\ -0.0005 & +0.2681 \end{bmatrix}$	$\begin{bmatrix} -1.0000 & +0.0001 \\ +0.0001 & +1.0000 \end{bmatrix}$	$\begin{bmatrix} -3.7319 & +1.0006 \\ 0 & +0.2680 \end{bmatrix}$	$\begin{bmatrix} +3.7320 & +1.0000 \\ +0.0000 & +0.2680 \end{bmatrix}$
$k = 5$	$\begin{bmatrix} +3.7320 & +1.0000 \\ +0.0000 & +0.2680 \end{bmatrix}$	$\begin{bmatrix} -1.0000 & -0.0000 \\ -0.0000 & +1.0000 \end{bmatrix}$	$\begin{bmatrix} -3.7320 & -1.0000 \\ 0 & +0.2679 \end{bmatrix}$	$\begin{bmatrix} +3.7321 & -1.0000 \\ -0.0000 & +0.2679 \end{bmatrix}$

The matrix A_k converges to an upper triangular form: its diagonal components converge to the eigenvalues of A :

$$\lambda_1 = 2 + \sqrt{3} \approx 3.7321, \quad \lambda_2 = 2 - \sqrt{3} \approx 0.2679,$$

its component below the diagonal converges to zero, and its component above the diagonal oscillates between two limit points: $+1$ and -1 .

17.5 Advantages of the QR algorithm

- The QR algorithm produces *all* eigenvalues of A in “one go”, by a single sequence of iterations. By contrast, the power method (Chapter 16) requires a separate sequence of iterations for each eigenvalue.
- The QR algorithm does *not* require an initialization – there is no need for an arbitrary choice of an initial vector q_0 or an initial shift ρ_0 that the power method relies upon.
- An important case is that of a Hermitian matrix A . We know that all the A_k 's are Hermitian matrices, too. Since the lower triangle of A_k converges to zero, so does its mirror image – the upper triangle. Therefore A_k converges to a diagonal matrix, a very nice result.

Despite all the obvious advantages, the “pure” QR algorithm is still quite expensive. Mainly, each iteration is very costly: the QR decomposition takes $2n^3$ flops (Sect. 9.8), and the multiplication of two $n \times n$ matrices R_k and Q_k generally takes $2n^3$ flops, as one can verify directly. We can take advantage of the triangular structure of R_k and avoid multiplications by zeros, then the cost of multiplication $R_k \times Q_k$ will be half of $2n^3$ flops, i.e., n^3 flops. This makes it $3n^3$ flops total, for each QR step. Thus if we make p QR steps, the total cost will be $3pn^3$.

This seems to be an improvement over $2pn^4/3$ flops needed by the power method (Section 16.22), but in practice this cost is still too high. We also need to remember that the convergence is slow (linear), thus the pure QR algorithm may require many more iterations than the power method does (see Section 16.3).

Fortunately, the computational cost can be dramatically reduced with the help of Hessenberg matrices, see below. The use of Hessenberg matrices will also allow us to develop tricks that speed up the convergence, from linear to quadratic.

17.6 Hessenberg matrix

$A \in \mathbb{C}^{n \times n}$ is called an (*upper*) *Hessenberg matrix* if $a_{ij} = 0$ for all $i > j + 1$, i.e., whenever A has the form

$$A = \begin{bmatrix} \times & \times & \cdots & \cdots & \times \\ \times & \times & \ddots & \ddots & \vdots \\ 0 & \times & \times & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \times & \times \end{bmatrix}$$

Thus, A may have non-zero components in its upper triangle and on its subdiagonal. It is just “one step away” from being an upper triangular.

Recall that an upper triangular matrix has its eigenvalues on its diagonal, i.e., its eigenvalues are “exposed”. The eigenvalues of a Hessenberg matrix are *not* exposed, they are “hidden”, but if any subdiagonal component is zero, i.e., $a_{i+1,i} = 0$ for some i , then the eigenvalue problem can be simplified:

17.7 Lucky reduction of the eigenvalue problem

Suppose A is Hessenberg and $a_{i+1,i} = 0$ for some $1 \leq i \leq n - 1$. Then A has a block-diagonal structure

$$A = \begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix}$$

where $A_1 \in \mathbb{C}^{i \times i}$ and $A_2 \in \mathbb{C}^{(n-i) \times (n-i)}$. Now the eigenvalues of A consist of those of A_1 and A_2 . Thus it is enough to find the eigenvalues of the two smaller matrices A_1 and A_2 , while the matrix B can be completely ignored.

17.8 Theorem (Hessenberg decomposition)

Every matrix $A \in \mathbb{C}^{n \times n}$ is unitary equivalent to a Hessenberg matrix, i.e.

$$A = Q^* H Q$$

where H is a Hessenberg matrix and Q is a unitary matrix. There is an exact finite algorithm (*Arnoldi algorithm*) for computing H and Q .

Proof. The matrix equation $A = Q^* H Q$ can be rewritten as $AQ^* = Q^* H$. Denote by q_i , $1 \leq i \leq n$, the columns of the unitary matrix Q^* and by h_{ij} the entries of H . Equating the columns of the matrices AQ^* and $Q^* H$ (and remembering that $h_{ij} = 0$ for $i > j + 1$) we obtain the following system of equations

$$\begin{aligned} Aq_1 &= h_{11}q_1 + h_{21}q_2 \\ Aq_2 &= h_{12}q_1 + h_{22}q_2 + h_{32}q_3 \\ &\dots \\ Aq_i &= h_{1i}q_1 + \dots + h_{ii}q_i + h_{i+1,i}q_{i+1} \\ &\dots \\ Aq_n &= h_{1n}q_1 + \dots + h_{n-1,n}q_{n-1} + h_{n,n}q_n \end{aligned}$$

Note that the last equation is slightly different from the others, since it terminates on a diagonal entry of H .

Now the Arnoldi algorithm goes along the lines similar to the classical Gram-Schmidt orthogonalization. We pick an arbitrary unit vector q_1 , compute $v_1 = Aq_1$ and

$$w_2 = v_1 - \langle v_1, q_1 \rangle q_1$$

It is easy to check that w_2 will be orthogonal to q_1 . Then we set $h_{11} = \langle v_1, q_1 \rangle$ and $h_{21} = \|w_2\|$ and define $q_2 = w_2/\|w_2\|$. This enforces the first equation in the above system.

Then, for every $i = 2, \dots, n - 1$ we make four steps:

Step 1: compute $v_i = Aq_i$.

Step 2: for all $j = 1, \dots, i$ compute $h_{ji} = \langle v_i, q_j \rangle$.

Step 3: compute $w_i = v_i - \sum_{j=1}^i h_{ji} q_j$.

(note: the vector w_i is guaranteed to be orthogonal to q_1, \dots, q_i).

Step 4: compute $h_{i+1,i} = \|w_i\|$ and $q_{i+1} = w_i/h_{i+1,i}$.

Finally, for $i = n$ we execute steps 1 and 2 only.

An exception may occur at step 4 if it happens that $h_{i+1,i} = 0$. In that case we simply pick an arbitrary unit vector q_{i+1} orthogonal to q_1, \dots, q_i and continue the procedure “as if nothing happened”. \square

The exceptional case $h_{i+1,i} = 0$ in Step 4 is sometimes referred to as the *breakdown* of the Arnoldi algorithm. This term is quite misleading, however, since the method does not really break down. In fact, the resulting Hessenberg matrix H will then have a simpler structure leading to a lucky reduction of the eigenvalue problem (Section 17.7).

The matrices H and Q in Theorem 17.8 are far from being unique. The very first step of the Arnoldi algorithm is based on an arbitrary choice of a unit vector q_1 (the first column of Q^*). Actually, by choosing a simple vector q_1 (such as $q_1 = e_1$), we can reduce the amount of computational work, see next.

17.9 Cost of Arnoldi algorithm

The cost of Step 1 is $2n^2$ flops, the cost of Step 2 is $2ni$ flops, the same for Step 3, and lastly Step 4 takes $3n$ flops. The total cost is then

$$\sum_{i=1}^n (2n^2 + 4ni + 3n) \sim 4n^3$$

(by choosing $q_1 = e_1$ one can save some work and compute H and Q in $\frac{10}{3}n^3$ flops; see the textbook for more details). This cost is comparable to the cost of one QR step for a generic (non-Hessenberg) matrix.

The Hessenberg decomposition can be regarded as a *pre-processing* of the matrix A ; it only needs to be done once, and then the QR algorithm can be applied to the resulting Hessenberg matrix. It is important, obviously, that the Hessenberg property is not lost during the QR iterations.

17.10 Preservation of Hessenberg structure

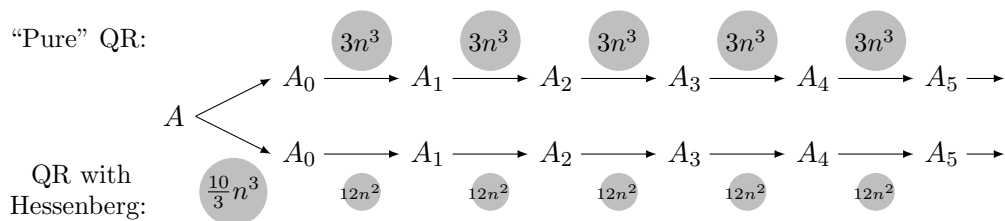
If an invertible matrix A_0 is Hessenberg, then all the matrices A_k generated by the QR algorithm will be Hessenberg matrices as well.

Proof. By induction, let A_{k-1} be Hessenberg. Then $A_{k-1} = Q_k R_k$ and so $Q_k = A_{k-1} R_k^{-1}$. This is a product of a Hessenberg matrix A_{k-1} and an upper triangular matrix R_k^{-1} . One can verify by direct inspection that such a product is always a Hessenberg matrix. Now $A_k = R_k Q_k$ is also a product of an upper triangular matrix R_k and a Hessenberg matrix Q_k , so it is a Hessenberg matrix, too. \square

For a noninvertible Hessenberg matrix A_0 , the above property remains valid, but the proof is more complicated and we leave it out. Basically, one can show that Q_k constructed by Gram-Schmidt orthogonalization (Chapter 9) is also Hessenberg, thus A_k will again be a Hessenberg matrix.

17.11 Cost of a QR step for Hessenberg matrices

For Hessenberg matrices, the QR algorithm can be implemented with a substantial reduction of computational cost. First, the QR decomposition can be done via Givens rotators, it will take $6n^2$ flops, according to Section 14.15. Second, Q_k is a product of $n - 1$ rotators, and the multiplication of R_k by Q_k can be organized so that it will take $\leq 6n^2$ flops (we omit details). The total for one QR step is then $\leq 12n^2$ flops, a dramatic drop from $3n^3$ flops required for a non-Hessenberg matrix A (as we said after Section 17.5).



17.12 The case of Hermitian matrices

An important case is that of Hermitian matrices. In that case A_0 will be both Hermitian (Section SAequiv) and Hessenberg. Hence A_0 will be a *tridiagonal matrix* ($h_{ij} = 0$ for all $|i - j| > 1$). Its construction by the Arnoldi algorithm takes only $\frac{4}{3}n^3$ flops (see the textbook for details).

The use of Hessenberg matrices greatly reduces the cost of each QR step. Another standing issue is a slow (linear) convergence (Theorem 17.3), with ratio

$$r = \max_k |\lambda_{k+1}/\lambda_k| < 1. \quad (17.1)$$

For example, if A has eigenvalues 1, 9, and 10, then the ratio is $r = 0.9$, and the process would be crawling. Fortunately, we have improvement in this respect, too:

17.13 Theorem (without proof)

Assume that A_0 , and hence A_k for all $k \geq 1$, are Hessenberg matrices. Then the convergence $a_{i,i-1}^{(k)} \rightarrow 0$ as $k \rightarrow \infty$ in Theorem 17.3 is linear with ratio $r_i = |\lambda_i/\lambda_{i-1}|$. In addition, the convergence $a_{nn}^{(k)} \rightarrow \lambda_n$ is linear with ratio $r_n = |\lambda_n/\lambda_{n-1}|$.

Now each subdiagonal entry $a_{i,i-1}^{(k)}$ has *its own* ratio of convergence r_i , so some may converge faster than others. The convergence is no longer governed by one single ratio (17.1). For example, if A has eigenvalues 1, 9, and 10, then the last (bottom) subdiagonal entry converges with ratio $r = 1/9$, which is very fast. This new flexibility allows us to develop smart tricks accelerating the convergence of some selected entries first, and then all the remaining entries.

17.14 QR algorithm with shift

One can modify the matrix A to decrease the ratio $r_n = |\lambda_n/\lambda_{n-1}|$ and thus make the convergence

$$a_{n,n-1}^{(k)} \rightarrow 0 \quad \text{and} \quad a_{nn}^{(k)} \rightarrow \lambda_n$$

of the two bottom entries faster.

This can be done with the help of *shifting*, as in Section 16.18. Recall that for any $\rho \in \mathbb{C}$ the matrix $A - \rho I$ has eigenvalues $\lambda_1 - \rho, \dots, \lambda_n - \rho$. The new matrix $A - \rho I$ is still Hessenberg (or Hermitian) if A itself was Hessenberg (resp., Hermitian), so we lose nothing by replacing A with $A - \rho I$.

Now suppose ρ is a good approximation to the smallest eigenvalue λ_n . If we apply the QR steps to the matrix $A' = A - \rho I$, then the convergence of the two bottom entries of the new matrix will be linear with ratio

$$r'_n = |\lambda_n - \rho|/|\lambda_{n-1} - \rho|.$$

Now we can control the speed of convergence by choosing/adjusting the shift ρ . Obviously, if $\rho \rightarrow \lambda_n$, then $r'_n \rightarrow 0$. Thus the better ρ approximates λ_n the faster the convergence of the two bottom entries will be.

17.15 QR algorithm with Rayleigh quotient shift

The approximation ρ can be updated at every iteration, as in Section 16.20, by using the Rayleigh quotient

$$\rho = \rho_k = u_k^* A_k u_k$$

where u_k is an approximate unit eigenvector of the matrix A_k corresponding to its smallest eigenvalue λ_n . In practice, a simple and convenient choice for u_k is $u_k = e_n$, which gives $\rho_k = a_{nn}^{(k)}$. Then the *QR algorithm with shift* goes as follows:

$$\begin{aligned} A_{k-1} - \rho_{k-1}I &= Q_k R_k && \text{(QR decomposition of } A_{k-1} - \rho_{k-1}I) \\ R_k Q_k + \rho_{k-1}I &= A_k && \text{(computation of the next matrix } A_k) \\ \rho_k &= a_{nn}^{(k)} && \text{(setting } \rho_k \text{ to the trailing entry of } A_k) \end{aligned}$$

This is called the *Rayleigh quotient shift*. The convergence of the two bottom entries of A_k is now quadratic, basically for the same reasons as it was in Section 16.20.

The use of Rayleigh quotient shift allows us very quickly – say, in about five iterations – to “kill” the bottom subdiagonal entry $a_{n,n-1}^{(k)}$ and find the smallest eigenvalue λ_n of the matrix A . But the other subdiagonal entries, $a_{i+1,i}^{(k)}$, $1 \leq i \leq n-2$, will move to zero slowly – linearly, with variable ratios. For example if A has eigenvalues 1, 9, and 10, then the smallest one will be found very fast, but the other two may take 50–100 (or more) iterations, because the convergence will proceed with a pretty bad ratio $r = 0.9$.

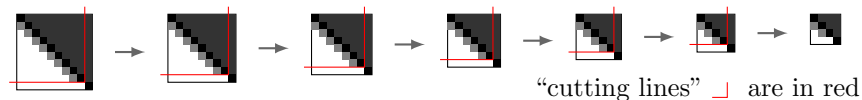
So we apply one more trick – the “lucky reduction” described in Section 17.7 – to “split off” the smallest eigenvalue λ_n and reduce the size of the matrix A :

17.16 QR algorithm with deflation

When $a_{n,n-1}^{(k)}$ becomes practically zero and $a_{nn}^{(k)}$ becomes practically equal to λ_n , the matrix A_k will have the following form:

$$A_k = \begin{bmatrix} \hat{A}_k & b_k \\ 0 & \lambda_n \end{bmatrix}$$

where \hat{A}_k is an $(n-1) \times (n-1)$ Hessenberg matrix. Its eigenvalues are those of A , minus λ_n , i.e., they are $\lambda_1, \dots, \lambda_{n-1}$. Now we can apply further steps of the QR algorithm (with shift) to the matrix \hat{A}_k , instead of A_k . This quickly produces its smallest eigenvalue, λ_{n-1} , which can be split off in the same manner. Then we reduce our matrix further, to the size $(n-2) \times (n-2)$, and so on. This procedure is called the *deflation* of the matrix A :



In practice, each eigenvalue of A requires just 3-5 iterations (QR steps), on the average. For Hermitian matrices, it is even faster – just 2-3 iterations per eigenvalue. Remember also that each QR step for Hessenberg matrices is very cheap (just $12n^2$ flops). This is how the QR algorithm with Hessenberg matrices, Rayleigh quotient shift, and deflation achieves a top speed.

The above procedure works well on generic matrices, but it fails occasionally in certain exceptional cases. One such case is a matrix A with two equally large eigenvalues, i.e., $|\lambda_i| = |\lambda_j|$ for some $i \neq j$. Such a matrix does not meet the requirements of Theorem 17.3, so the convergence is not guaranteed. In that case the QR algorithm may “get stuck”, as illustrated by the following example.

17.17 Example

Let $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Note that A is already a Hessenberg matrix, thus the Hessenberg decomposition is not needed here. Now the pure QR algorithm gives

$$A_0 = Q_1 R_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

hence

$$A_1 = R_1 Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = A$$

and we are “back to square one”. The process goes nowhere. The Rayleigh quotient shift $\rho = a_{22}$ proposed in Section 17.15 has no effect either, since $a_{22} = 0$. The reason of this failure is that the eigenvalues of A , which are $+1$ and -1 , have equal absolute values – this is a symmetry which confuses the QR algorithm, it “cannot decide” which eigenvalue to approach. To break this symmetry, one needs to choose the shift ρ differently.

An alternative shift ρ can be used whenever the procedure “gets stuck”:

17.18 Wilkinson shift

At step k of the QR algorithm with shift, let B_k denote the trailing 2×2 block at the bottom right corner of A_k :

$$B_k = \begin{bmatrix} a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{n,n}^{(k)} \end{bmatrix} \quad (17.2)$$

Now set ρ_k to the eigenvalue of B_k that is closer to $a_{n,n}^{(k)}$ (in case of a tie, either one can be taken). This is called *Wilkinson shift*.

The eigenvalues of a 2×2 matrix can be easily (and precisely) computed by the quadratic formula, whether they are real or complex. If they are real, then the Wilkinson shift will help break the symmetry like in Example 17.17.

17.19 Example 17.17 continued

The Wilkinson shift here is either $\rho = 1$ or $\rho = -1$. Let us choose $\rho = -1$. Then

$$A_0 - \rho I = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = Q_1 R_1 = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} \sqrt{2} & \sqrt{2} \\ 0 & 0 \end{bmatrix}$$

and then

$$A_1 = R_1 Q_1 + \rho I = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

The matrix A_1 is diagonal and its diagonal entries are the eigenvalues of A .

So the QR algorithm with the Wilkinson shift converges in just one step.

Lastly, we discuss an important case of real matrices with complex eigenvalues. It was already mentioned in Section 16.22(c).

If the matrix A is real (which is the most common case in practice), then its complex eigenvalues come in conjugate pairs $a \pm ib$. They have equal absolute values $|a + bi| = |a - bi|$, immediately violating the main assumption of Theorem 17.3.

Furthermore, the pure QR algorithm is bound to fail for the following simple reason. If A is real, then all Q_k , R_k and A_k will be real matrices as well, thus we cannot even expect the real diagonal entries of A_k to converge to the complex eigenvalues of A . In this case A_k may not converge to anything at all.

The QR algorithm with shift may be able to find complex eigenvalues of a real matrix A , but it needs to use complex shifts $\rho \notin \mathbb{R}$ and complex matrices $A - \rho I$, which is inconvenient and expensive. Fortunately, the QR algorithm can be organized so that it operates with real matrices only, for as long as possible, and obtains pairs of complex conjugate eigenvalues only at the very last step:

17.20 Wilkinson shifts for complex eigenvalues of real matrices

Suppose at the step k of the QR algorithm, the matrix A_k is still real, but the eigenvalues of its 2×2 trailing block B_k in (17.2) are non-real complex numbers. Let us denote them by ρ_k and $\bar{\rho}_k$ (remember that they are complex conjugate to each other). Then the Wilkinson shift ρ_k will be a complex number, and the matrix $A_k - \rho_k I$ will be complex, too. The QR step will then produce a complex matrix A_{k+1} , which we do not really want.

In this case one can use the following trick to avoid further working with complex matrices: at the very next QR step we set $\rho_{k+1} = \bar{\rho}_k$, i.e., we choose the shift ρ_{k+1} to be the *other* complex eigenvalue of B_k . Then, remarkably, the resulting matrix A_{k+2} will be real again, for the reasons explained below.

Furthermore, the values ρ_k and $\bar{\rho}_k$ will approximate two complex eigenvalues of the matrix A . Therefore, the QR algorithm with Wilkinson shift is able to compute conjugate pairs of complex eigenvalues of a real matrix A , and stay with real matrices for as long as possible.

Actually, there is no need to compute the complex matrix A_{k+1} mentioned above. One can just combine the two QR steps together and construct A_{k+2} directly from A_k (bypassing A_{k+1}). This can be carried out entirely in real arithmetic. The resulting all-real procedure is called the *double-step QR algorithm with Wilkinson shift*.

In the following, all complex scalars and matrices are marked red. This includes matrix A_2 , which will remain hypothetically complex until we prove that it is real.

17.21 Analysis of the double-step

Let $A_0 \in \mathbb{R}^{n \times n}$ be a real matrix. Let ρ and $\bar{\rho}$ be two complex non-real conjugate numbers that are *not* eigenvalues of A_0 . Let us apply two QR steps with the shifts ρ and $\bar{\rho}$:

$$A_0 - \rho I = Q_1 R_1 \quad R_1 Q_1 + \rho I = A_1 \quad (17.3)$$

$$A_1 - \bar{\rho} I = Q_2 R_2 \quad R_2 Q_2 + \bar{\rho} I = A_2 \quad (17.4)$$

Since the matrices $A_0 - \rho I$ and $A_1 - \bar{\rho} I$ are nonsingular, the above QR decompositions may be constructed so that R_1 and R_2 have positive real entries, cf. Section 9.7. Then, remarkably, A_2 will be a real matrix.

Proof. It follows from (17.3) that $R_1 = Q_1^{-1} A_0 - \rho Q_1^{-1}$, hence

$$A_1 = (Q_1^{-1} A_0 - \rho Q_1^{-1}) Q_1 + \rho I = Q_1^{-1} A_0 Q_1$$

Similarly, it follows from (17.4) that $R_2 = Q_2^{-1} A_1 - \bar{\rho} Q_2^{-1}$, hence

$$A_2 = (Q_2^{-1} A_1 - \bar{\rho} Q_2^{-1}) Q_2 + \bar{\rho} I = Q_2^{-1} A_1 Q_2$$

Next we will compute the product $(A_0 - \bar{\rho} I)(A_0 - \rho I)$ in two ways. First,

$$\begin{aligned} (A_0 - \bar{\rho} I)(A_0 - \rho I) &= A_0^2 - (\rho + \bar{\rho})A_0 + \rho\bar{\rho}I \\ &= A_0^2 - 2(\operatorname{Re} \rho)A_0 + |\rho|^2 I. \end{aligned}$$

Since $(\operatorname{Re} \rho)$ and $|\rho|$ are real numbers, the matrix $A = (A_0 - \bar{\rho} I)(A_0 - \rho I)$ is entirely real.

On the other hand, we have

$$\begin{aligned} A &= (A_0 - \bar{\rho} I)(A_0 - \rho I) \\ &= (A_0 - \bar{\rho} I)Q_1 R_1 \\ &= Q_1 Q_1^{-1} (A_0 - \bar{\rho} I) Q_1 R_1 \\ &= Q_1 (Q_1^{-1} A_0 Q_1 - \bar{\rho} I) R_1 \\ &= Q_1 (A_1 - \bar{\rho} I) R_1 \\ &= Q_1 Q_2 R_2 R_1. \end{aligned}$$

This is actually a QR decomposition of the matrix A , with orthogonal matrix $Q = Q_1 Q_2$ and upper triangular matrix $R = R_2 R_1$. Since the matrices R_1 and R_2 have positive diagonal entries, so does its product R . Therefore, according to Section 9.7, the above QR decomposition is unique.

But every real matrix A has a real QR decomposition $A = QR$ with positive diagonal entries of R (by Sections 9.5 and 9.6), hence by its uniqueness both matrices $Q = Q_1 Q_2$ and $R = R_2 R_1$ must be real. Lastly,

$$A_2 = Q_2^{-1} A_1 Q_2 = (Q_1 Q_2)^{-1} A_0 (Q_1 Q_2) = Q^{-1} A_0 Q.$$

As a product of three real matrices, A_2 must be a real matrix as well. \square

Exercise 17.1. (JPE, September 2009) Let $A \in \mathbb{C}^{n \times n}$ be nonsingular. Let $A = Q_1 R_1$ be a QR decomposition of A , and for $k \geq 1$ define inductively $AQ_k = Q_{k+1} R_{k+1}$, a QR decomposition of AQ_k .

- (a) Prove that there exists an upper triangular matrix U_k such that $Q_k = A^k U_k$ and a lower triangular matrix L_k such that $Q_k = (A^*)^{-k} L_k$.
- (b) Suppose $\lim_{k \rightarrow \infty} R_k = R_\infty$ and $\lim_{k \rightarrow \infty} Q_k = Q_\infty$ exist. Determine the eigenvalues of A in terms of R_∞ .

Exercise 17.2. (JPE, May 2006) Let $A \in \mathbb{C}^{n \times n}$ be tri-diagonal and Hermitian, with all its super-diagonal entries nonzero. Prove that the eigenvalues of A are distinct.

(Hint: show that for any scalar λ , the matrix $A - \lambda I$ has rank at least $n - 1$.)