Åke Björck
Linköping University

Germund Dahlquist
Royal Institute of Technology

Numerical Methods in Scientific Computing

Volume II

Working copy, April 10, 2008

siam

Contents

List of Figures

List of Tables

Chapter 7

Direct Methods for Linear System

Most amateur algorithm writers, like most amateur scientists, seem to think that an algorithm is ready for publication at the point where a professional should realize that the hard and tedious work is just beginning.
—George E. Forsythe, Algorithms for Scientific Computation.
Communications of the ACM (1966).

7.1 Linear Algebra and Matrix Computations

Matrix computations are fundamental and ubiquitous in scientific computing. Although there is a link between matrix computations and linear algebra as taught in department of mathematics there also are several fundamental differences. In mathematics linear algebra is mostly about what can be deduced from the axiomatics definitions of fields and vector spaces. Many of the notions central to matrix computations, such as ill-conditioning, norms, and orthogonality, do not extend to arbitrary fields. Mathematicians tend to think of matrices as coordinate representations of linear operators with respect to some basis. However, matrices can also represent other things, such as bilinear forms, graph structures, images, DNA measurements, and so on. Many of these make little sense when viewed as an operator. In numerical linear algebra one works in the field of real or complex numbers, which allows the use of a rich set of tools like QR factorization and singular value decomposition.

The numerical solution of a system of linear equations enters at some stage in almost all applications. Even though the mathematical theory is simple and algorithms have been known for centuries, decisive progress has been made in the last few decades. It is important to note that some methods which are perfectly acceptable for theoretical use, may be useless for solving systems with several thousands or even million of unknown. Algorithms for solving linear systems are the most widely used in scientific computing and it is of great importance that they are both efficient and reliable. They also have to be adopted continuously as computer architectures change. Critical details in the algorithms can influence the efficiency and accuracy in a way the beginner can hardly expect. It is strongly advisable to use efficient and well-tested software available; see Sec. 7.5.5.

Two quite different classes of methods for solving systems of linear equations are of interest: **direct** methods and **iterative** methods. In direct methods the system is transformed by a sequence of elementary transformations into a system of simpler form, e.g., triangular or diagonal form, which can be solved in an elementary way. Disregarding rounding errors, direct methods give the exact solution after a finite number of arithmetic operations. The most important direct method is Gaussian elimination. This is the method of choice when A has full rank and no special structure. Gaussian elimination will be treated in this chapter.

A matrix A for which only a small fraction of the elements are nonzero is called **sparse**. The simplest case is when A has a banded structure, but also more general sparsity patterns can be taken advantage of; see Sec. 7.8. Indeed, without exploiting sparsity many important problems would be intractable!

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

7.1.1 Matrix Algebra

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

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

The term matrix was first coined in 1848 by J.J. Sylvester. The English mathematician Arthur Cayley (1821–1895) made the fundamental discovery that such an array of numbers can be conceived as one single algebraic quantity $A = (a_{ij})$, with which certain algebraic operations can be performed. We write $A \in \mathbf{R}^{m\times n}$, where $\mathbf{R}^{m\times n}$ denotes the set of all real $m \times n$ matrices. If $m = n$, then the matrix A is said to be square and of order n. An empty matrix is a matrix with at least one dimension equals 0, Empty matrices are convenient to use as place holders.

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

We will follow a notational convention introduced by Householder[1] and use uppercase letters (e.g., A, B) to denote matrices. The corresponding lowercase

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

letters with subscripts ij then refer to the (i, j) component of the matrix (e.g., a_{ij}, b_{ij}). Greek letters $\alpha, \beta, \ldots$ are usually used to denote scalars. Column vectors are usually denoted by lower case letters (e.g., x, y).

Basic Operations

The two fundamental operations from which everything else can be derived are addition and multiplication. The algebra of matrices satisfies the postulates of ordinary algebra with the exception of the commutative law of multiplication.

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

$$C = A + B, \qquad c_{ij} = a_{ij} + b_{ij}. \tag{7.1.1}$$

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

$$B = \alpha A, \qquad b_{ij} = \alpha a_{ij}. \tag{7.1.2}$$

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

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

that is, the ith component of y is the sum of products of the elements in the ith row of A and the column vector x. This definitions means that the linear system

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

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

The general case now follows from the requirement that if $z = By$ and $y = Ax$, then substituting we should obtain $z = BAx = Cx$, where $C = BA$. Clearly the product is only defined if and only if the number of columns in A equals the number of rows in B. The product of the matrices $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{n\times p}$ is the matrix

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

Matrix multiplication is associative and distributive,

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

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

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

Lemma 7.1.1.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The partial ordering "$\leq$" for matrices A, B and vectors x, y is to be interpreted component-wise[2]

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

It follows that if $C = AB$, then

$$|c_{ij}| \leq \sum_{k=1}^{n} |a_{ik}|\,|b_{kj}|,$$

and hence $|C| \leq |A|\,|B|$. A similar rule holds for matrix-vector multiplication.

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

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

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

Special Matrices

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

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

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

The **identity matrix** I_n of order n is the matrix $I_n = (\delta_{ij})$, where δ_{ij} is the **Kronecker symbol** defined by

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{7.1.9}$$

We also have

$$I_n = \operatorname{diag}(1, 1, \dots, 1) = (e_1, e_2, \dots, e_n),$$

where the k-th column of I_n is denoted by e_k. For all square matrices of order n, it holds that

$$AI_n = I_nA = A.$$

[2]Note that when A and B are square symmetric matrices $A \leq B$ in other contexts can mean that the matrix $B - A$ is positive semi-definite.

If the size of the unit matrix is obvious we delete the subscript and just write I.

A matrix A for which all nonzero elements are located in consecutive diagonals is called a **band matrix**. A square matrix A is said to have **upper bandwidth** r and to have **lower bandwidth** s if

$$a_{ij} = 0, \quad j > i + r, \qquad a_{ij} = 0, \quad i > j + s, \tag{7.1.10}$$

respectively. This means that the number of nonzero diagonals above and below the main diagonal are r and s respectively. The maximum number of nonzero elements in any row is then $w = r + s + 1$, which is the **bandwidth** of A.

Note that the bandwidth of a matrix depends on the ordering of its rows and columns. For example, the matrix

$$\begin{pmatrix} a_{11} & a_{12} & & & & \\ a_{21} & a_{22} & a_{23} & & & \\ a_{31} & a_{32} & a_{33} & a_{34} & & \\ & a_{42} & a_{43} & a_{44} & a_{45} & \\ & & a_{53} & a_{54} & a_{55} & a_{56} \\ & & & a_{64} & a_{65} & a_{66} \end{pmatrix}$$

has $r = 1$, $s = 2$ and $w = 4$.

Several classes of band matrices that occur frequently have special names. A matrix for which $r = s = 1$ is called **tridiagonal**; e.g.,

$$A = \begin{pmatrix} a_1 & c_2 & & & \\ b_2 & a_2 & c_3 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-1} & a_{n-1} & c_n \\ & & & b_n & a_n \end{pmatrix}.$$

Note that the $3n - 2$ nonzero elements in A can be conveniently be stored in three vectors a, b and c of length $\leq n$.

If $r = 0$, $s = 1$ ($r = 1$, $s = 0$) the matrix is called lower (upper) **bidiagonal**. A matrix with $s = 1$ ($r = 1$) is called an upper (lower) **Hessenberg** matrix[3]. For example, an upper Hessenberg matrix of order five has the structure

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ 0 & h_{32} & h_{33} & h_{34} & h_{35} \\ 0 & 0 & h_{43} & h_{44} & h_{45} \\ 0 & 0 & 0 & h_{54} & h_{55} \end{pmatrix}.$$

It is convenient to introduce some additional notations for manipulating band matrices.[4]

[3] Named after the German mathematician and engineer Karl Hessenberg (1904–1959). These matrices first appeared in [205].

[4] These notations are taken from MATLAB.

Definition 7.1.2.

If $a = (a_1,\, a_2,\, \dots,\, a_{n-r})^T$ *is a column vector with* $n - r$ *components, then*

$$A = \operatorname{diag}(a, k), \quad |k| < n,$$

is a square matrix of order n *with the elements of* a *on its* k*th diagonal;* $k = 0$ *is the main diagonal;* $k > 0$ *is above the main diagonal;* $k < 0$ *is below the main diagonal.*

Let A *be a square matrix of order* n*. Then*

$$\operatorname{diag}(A, k) \in \mathbf{R}^{(n-k)}, \quad |k| < n,$$

is the column vector formed from the elements of the k*th diagonal of* A*.*

For example, $\operatorname{diag}(A, 0) = \operatorname{diag}(a_{11}, a_{22}, \dots, a_{nn})$, is the main diagonal of A, and if if $0 \le k < n$,

$$\operatorname{diag}(A, k) = \operatorname{diag}(a_{1,k+1}, a_{2,k+2}, \dots, a_{n-k,n}),$$
$$\operatorname{diag}(A, -k) = \operatorname{diag}(a_{k+1,1}, a_{k+2,2}, \dots, a_{n,n-k}),$$

is the kth superdiagonal and subdiagonal of A, respectively.

Vector Spaces

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

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

Otherwise, if a nontrivial linear combination of $v_1, \dots, v_k$ is zero, the vectors are said to be linearly dependent. Then at least one vector v_i will be a linear combination of the rest.

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

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

The scalars ξ_i are called the components or coordinates of v with respect to the basis $\{v_i\}$. If the vector space $\mathcal{V}$ has a basis of n vectors, then every system of linearly independent vectors of $\mathcal{V}$ has at most k elements and any other basis of $\mathcal{V}$ has the same number k of elements. The number k is called the **dimension** of $\mathcal{V}$ and denoted by $\dim(\mathcal{V})$.

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

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

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

A **linear transformation** from the vector space $\mathbf{C}^n$ to $\mathbf{C}^m$ is a function f such that

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

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

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

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

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

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

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

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

Orthogonality

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

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

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

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

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

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

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

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

7.1.2 Submatrices and Block Matrices

A matrix formed by the elements at the intersection of a set of rows and columns of a matrix A is called a **submatrix**. For example, the matrices

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

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

Definition 7.1.3.
A **submatrix** *of $A = (a_{ij}) \in \mathbf{R}^{m\times n}$ is a matrix $B \in \mathbf{R}^{p\times q}$ formed by selecting p rows and q columns of A,*

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

where

$$1 \le i_1 \le i_2 \le \cdots \le i_p \le m, \qquad 1 \le j_1 \le j_2 \le \cdots \le j_q \le n.$$

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

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

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

where A_{IJ} is a matrix of dimension $p_I \times q_J$. We call such a matrix a **block matrix**. The partitioning can be carried out in many ways and is often suggested by the structure of the underlying problem. For square matrices the most important case is when $M = N$, and $p_I = q_I$, $I = 1 : N$. Then the diagonal blocks A_{II}, $I = 1 : N$, are square matrices.

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

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

Therefore, many algorithms defined for matrices with scalar elements have a simple generalization for partitioned matrices, provided that the dimensions of the blocks are such that the operations can be performed. When this is the case, the matrices are said to be partitioned **conformally**. The **colon notation** used in MATLAB is very convenient for handling partitioned matrices and will be used throughout this volume:

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

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

$x(j : k)$	is the vector$[x(j), x(j+1), \dots, x(k)]$,
$A(:, j)$	is the jth column of A,
$A(i, :)$	is the ith row of A,
$A(:, :)$	is the same as A,
$A(:, j : k)$	is the matrix$[A(:, j), A(:, j+1), \dots, A(:, k)]$,
$A(:)$	is all the elements of the matrix A regarded as a single column.

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

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1N} \\ 0 & R_{22} & R_{23} & \cdots & R_{2N} \\ 0 & 0 & R_{33} & \cdots & R_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & R_{NN} \end{pmatrix}.$$

Example 7.1.1.

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

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

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

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

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

Block Elimination

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

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

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}, \quad U = \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}, \tag{7.1.14}$$

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

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}, \qquad U^{-1} = \begin{pmatrix} U_{11}^{-1} & -U_{11}^{-1}U_{12}U_{22}^{-1} \\ 0 & U_{22}^{-1} \end{pmatrix}. \tag{7.1.15}$$

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

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

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

where A and D are square matrices. If A is nonsingular, then the variables x_1 can be eliminated by multiplying the first block equations from the left by $-CA^{-1}$ and adding the result to the second block equation. This is equivalent to **block Gaussian elimination** using the matrix A as pivot. The reduced system for x_2 becomes

$$(D - CA^{-1}B)x_2 = b_2 - CA^{-1}b_1. \tag{7.1.16}$$

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

$$S = D - CA^{-1}B \tag{7.1.17}$$

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

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

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

This gives a factorization of M in a product of a block lower and a block upper triangular matrix,

$$M = \begin{pmatrix} I & 0 \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ 0 & S \end{pmatrix}, \qquad S = D - CA^{-1}B. \tag{7.1.18}$$

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

From $M^{-1} = (LU)^{-1} = U^{-1}L^{-1}$ using the formulas (7.1.15) for the inverses of 2×2 block triangular matrices we get the **Banachiewicz** inversion formula[6]

$$\begin{aligned} M^{-1} &= \begin{pmatrix} A^{-1} & -A^{-1}BS^{-1} \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix} \\ &= \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}. \end{aligned} \tag{7.1.19}$$

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

$$M = \begin{pmatrix} I & BD^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} T & 0 \\ C & D \end{pmatrix}, \qquad T = A - BD^{-1}C, \tag{7.1.20}$$

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

$$M^{-1} = \begin{pmatrix} T^{-1} & -T^{-1}BD^{-1} \\ -D^{-1}CT^{-1} & D^{-1} + D^{-1}CT^{-1}BD^{-1} \end{pmatrix}. \tag{7.1.21}$$

If A and D are both nonsingular, then both triangular factorizations (7.1.18) and (7.1.20) exist.

Modified Linear Systems

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

Theorem 7.1.4.
Let A and D be square nonsingular matrices and let B and C be matrices of appropriate dimensions. If $(A - BD^{-1}C)$ exists and is nonsingular, then

$$(A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}, \tag{7.1.22}$$

which is the **Woodbury formula**.

Proof. The result follows directly by equating the $(1,1)$ blocks in the inverse M^{-1} in (7.1.19) and (7.1.21). □

The identity (7.1.22) appeared in several papers before Woodbury's report [392, 1950]. For a review of the history, see [204, 1981].

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

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

$$(A - BD^{-1}C)\hat{x} = b, \quad B, C^T \in \mathbf{R}^{n\times p}, \tag{7.1.23}$$

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

$$(A - BD^{-1}C)^{-1}b = x + A^{-1}B(D - CA^{-1}B)^{-1}Cx. \tag{7.1.24}$$

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

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

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

In the special case that $p = 1$, $D = \sigma$ is a scalar and the Woodbury formula (7.1.22) becomes

$$(A - \sigma^{-1}uv^T)^{-1} = A^{-1} + \alpha A^{-1}uv^TA^{-1}, \quad \alpha = 1/(\sigma - v^TA^{-1}u), \tag{7.1.25}$$

where $u, v \in \mathbf{R}^n$. This is also known as the **Sherman–Morrison formula**. It follows that the modified matrix $(A - \sigma^{-1}uv^T)$ is nonsingular if and only if $\sigma \neq v^TA^{-1}u$.

Let x be the solution to the linear system $Ax = b$. Using the Sherman–Morrison formula the solution of the modified system

$$(A - \sigma^{-1}uv^T)\hat{x} = b. \tag{7.1.26}$$

can be written

$$\hat{x} = x + \alpha w(v^Tx), \quad \alpha = 1/(\sigma - v^Tw), \tag{7.1.27}$$

where $w = A^{-1}u$. Hence, *computing the inverse* A^{-1} *can be avoided.*

For a survey of applications of the Woodbury and the Sherman–Morrison formulas, see Hager [195, 1989]. Note that these should be used with caution since they can not be expected to be numerically stable in all cases. In particular, accuracy can be lost when the initial problem is more ill-conditioned than the modified one.

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

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

$$(x, y) = x^Hy = \sum_{k=1}^{n} \bar{x}_k y_k, \quad x^H = (\bar{x}_1, \ldots, \bar{x}_n), \tag{7.1.28}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

From (7.1.28) it follows that unitary matrices preserve the Hermitian inner product. In particular, the Euclidian length of a vector is invariant under unitary transformations, i.e., $\|Ux\|_2^2 = \|x\|_2^2$. Note that when the vectors and matrices are real the definitions for the complex case are consistent with those made for the real case.

7.1.3 Permutations and Determinants

The classical definition of the **determinant**[7] requires some elementary facts about permutations, which we now state.

Let $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ be a permutation of the integers $\{1, 2, \ldots, n\}$. The pair α_r, α_s, $r < s$, is said to form an inversion in the permutation if $\alpha_r > \alpha_s$.

[7]Determinants were first introduced by Leibniz (1693) and Cayley (1841). Determinants arise in many parts of mathematics, such as combinatorial enumeration, graph theory, representation theory, statistics, and theoretical computer science. The theory of determinants is covered in a monumental five volume work "The Theory of Determinants in the Historical Order of Development" by Thomas Muir (1844–1934).

For example, in the permutation $\{2, \ldots, n, 1\}$ there are $(n-1)$ inversions $(2,1)$, $(3,1)$,..., $(n,1)$. A permutation α is said to be even and $\operatorname{sign}(\alpha) = 1$ if it contains an even number of inversions; otherwise the permutation is odd and $\operatorname{sign}(\alpha) = -1$.

The product of two permutations σ and τ is the composition $\sigma\tau$ defined by

$$\sigma\tau(i) = \sigma[\tau(i)], \quad i = 1 : n.$$

A **transposition** τ is a permutation which only interchanges two elements. Any permutation can be decomposed into a sequence of transpositions, but this decomposition is not unique.

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

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

where $p_1, \ldots, p_n$ is a permutation of $1, \ldots, n$. Notice that in a permutation matrix every row and every column contains just one unity element. Since P is uniquely represented by the integer vector $p = (p_1, \ldots, p_n)$ it need never be explicitly stored. For example, the vector $p = (2, 4, 1, 3)$ represents the permutation matrix

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

If P is a permutation matrix then PA is the matrix A with its rows permuted and AP is A with its columns permuted. Using the colon notation, these permuted matrix can be written `PA = A(p,:)` and `PA = A(:,p)`, respectively.

The transpose P^T of a permutation matrix is again a permutation matrix. Any permutation may be expressed as a sequence of transposition matrices. Therefore, any permutation matrix can be expressed as a product of transposition matrices $P = I_{i_1,j_1} I_{i_2,j_2} \cdots I_{i_k,j_k}$. Since $I^{-1}_{i_p,j_p} = I_{i_p,j_p}$, we have

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

that is permutation matrices are orthogonal and P^T effects the reverse permutation and thus

$$P^T P = P P^T = I. \tag{7.1.29}$$

Lemma 7.1.5.

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

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

Definition 7.1.6.

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

$$\det(A) = \sum_{\alpha\in S_n} \operatorname{sign}(\alpha)\, a_{1,\alpha_1} a_{2,\alpha_2} \cdots a_{n,\alpha_n}, \tag{7.1.30}$$

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

Note that there are $n!$ terms in (7.1.30) and each term contains exactly one factor from each row and each column in A. For example, if $n = 2$ there are two terms, and

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

From the definition it follows easily that

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

IF we collect all terms in (7.1.30) that contains the element a_{rs} these can be written as $a_{rs}A_{rs}$, where A_{rs} is called the complement of a_{rs}. Since the determinant contains only one element from row r and column s the complement A_{rs} does not depend on any elements in row r and column s. Since each product in (7.1.30) contains precisely one element of the elements a_{r1}, a_{r2},…,a_{rn} in row r it follows that

$$\det(A) = a_{r1}A_{r1} + a_{r2}A_{r2} + \cdots a_{rn}A_{rn}. \tag{7.1.31}$$

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

$$A_{rs} = (-1)^{r+s} D_{rs}, \tag{7.1.32}$$

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

Another scalar-valued function of a matrix is the **permanent**. Its definition is similar to that of the determinant, but in the sum (7.1.30) all terms are to be taken with positive sign; see Marcus and Minc [270, Sec. 2.9]. The permanent has no easy geometric interpretation and is used mainly in combinatorics. The permanent is more difficult to compute than the determinant, which can be computed in polynomial time.

The direct use of the definition (7.1.30) to evaluate $\det(A)$ would require about $nn!$ operations, which rapidly becomes infeasible as n increases. A much more efficient way to compute $\det(A)$ is by repeatedly using the following properties:

Theorem 7.1.7.

(i) *The value of the* $\det(A)$ *is unchanged if a row (column) in* A *multiplied by a scalar is added to another row (column).*

(ii) *The determinant of a triangular matrix equals the product of the elements in the main diagonal, i.e., if* U *is upper triangular*

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

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

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

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

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

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

Theorem 7.1.8.

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

$$x_j = \det(B_j)/\det(A), \quad j = 1:n. \tag{7.1.33}$$

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

Proof. Form the linear combination

$$a_{1j}A_{1r} + a_{2j}A_{2r} + \cdots a_{nj}A_{nr} = \begin{cases} 0 & \text{if } j \neq r, \\ \det(A) & \text{if } j = r. \end{cases} \tag{7.1.34}$$

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

Now take the ith equation in $Ax = b$,

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

multiply by A_{ir} and sum over $i = 1:n$. Then by (7.1.34) the coefficients of x_j, $j \neq r$, vanish and we get

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

The right hand side equals $\det(B_r)$ expanded by its rth column, which proves (7.1.33). □

The expression (7.1.33) is known as **Cramer's rule**.[8] Although elegant, it is both computationally expensive and numerically instable. even for $n = 2$; see Higham [211, p. 13]

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

$$\det(U) = \det(U_{11}) \det(U_{22}) \cdots \det(U_{NN}). \tag{7.1.35}$$

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

Example 7.1.2.

For the 2×2 block matrix M in (7.1.18) and (7.1.20) it follows using (7.1.35) that

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

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

$$\det(A - \lambda xy^T) = \det(A)(1 - \lambda y^T A^{-1} x). \tag{7.1.36}$$

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

The following determinant inequality is due to Hadamard.[9]

Theorem 7.1.9 (*Hadamard's determinant inequality*).

If $A = (a_1 \quad a_2 \quad \cdots \quad a_n) \in \mathbf{R}^{n \times n}$, then

$$|\det(A)| \leq \prod_{j=1}^{n} \|a_j\|_2, \tag{7.1.37}$$

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

Proof. Let $A = QR$, $R = (r_1 \quad r_2 \quad \cdots \quad r_n)$ be the QR factorization of A (see Sec. 8.3). Then since the determinant of an orthogonal matrix equals ± 1 we have

$$|\det(A)| = |\det(R)| = \prod_{j=1}^{n} |r_{jj}| \leq \prod_{j=1}^{n} \|r_j\|_2 = \prod_{j=1}^{n} \|a_j\|_2.$$

Since the rank of A is n we have $\det(A) \neq 0$. Equality holds if and only if $|r_{jj}| = \|a_{jj}\|_2$, which can only happen if the columns are mutually orthogonal. □

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

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

The following determinant identity is a useful tool in many algebraic manipulations.

Theorem 7.1.10 (*Sylvester's determinant identity*).

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

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

Then we have the identity

$$\det(A) \cdot \det(\hat{A}) = \det(A_{11}) \cdot \det(A_{22}) - \det(A_{12}) \cdot \det(A_{21}). \tag{7.1.38}$$

Proof. If the matrix A is square and nonsingular, then

$$\det(A_{ij}) = \pm \det \begin{pmatrix} A & \hat{a}_j \\ a_i^T & \alpha_{ij} \end{pmatrix} = \pm \det(A) \cdot (\alpha_{ij} - a_i^T A^{-1} \hat{a}_j). \tag{7.1.39}$$

with negative sign only possible if $i \neq j$. Then similarly

$$\begin{aligned} \det(A) \cdot \det(\hat{A}) &= \det(A) \cdot \det \begin{pmatrix} A & \hat{a}_{11} & \hat{a}_2 \\ a_1^T & \alpha_{11} & \alpha_{12} \\ a_2^T & \alpha_{21} & \alpha_{22} \end{pmatrix} \\ &= (\det(A))^2 \cdot \det \left[\begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix} - \begin{pmatrix} a_1^T \\ a_2^T \end{pmatrix} A^{-1} \begin{pmatrix} \hat{a}_1 & \hat{a}_2 \end{pmatrix} \right] \\ &= \det \begin{pmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{pmatrix}, \end{aligned}$$

where $\beta_{ij} = \alpha_{ij} - a_i^T A^{-1} \hat{a}_j$. Using (7.1.39) gives (7.1.38), which holds even when A is singular. □

7.1.4 The Singular Value Decomposition

The **singular value decomposition** (SVD) enables the solution of wide variety of matrix problems, which would otherwise be difficult. The SVD provides a diagonal form of a complex or real matrix A under an unitary (orthogonal) equivalence transformation. The history of this matrix decomposition goes back more than a century. However, its use in numerical computations was not practical until an algorithm to stably compute it was developed by Golub and Reinsch [181, 1970].

Theorem 7.1.11. (Singular Value Decomposition.)
Every matrix $A \in \mathbf{C}^{m\times n}$ of rank r can be written

$$A = U\Sigma V^H, \qquad \Sigma = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \in \mathbf{R}^{m\times n}, \tag{7.1.40}$$

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

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

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

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

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

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

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

Here $f(x)$ is a real-valued convex function[10] defined on a convex, compact set. It is well known (see, e.g., Ciarlet [68, Sec. 7.4]) that the maximum σ_1 is then attained on an extreme point of the set. Let v_1 be such a point with $\sigma_1 = \|Av_1\|$, $\|v_1\|_2 = 1$. If $\sigma_1 = 0$ then $A = 0$, and (7.1.40) holds with $\Sigma = 0$, and U and V arbitrary unitary matrices. Therefore, assume that $\sigma_1 > 0$, and set $u_1 = (1/\sigma_1) A v_1 \in \mathbf{C}^m$, $\|u_1\|_2 = 1$. Let the matrices

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

be unitary. (Recall that it is always possible to extend an unitary set of vectors to a unitary basis for the whole space.) Since $U_1^H A v_1 = \sigma_1 U_1^H u_1 = 0$ it follows that $U^H A V$ has the following structure:

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

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

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

We have $UA_1 y = AVy = Ax$ and, since U and V are unitary, it follows that

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

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

and hence,

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

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

The geometrical significance of this theorem is as follows. The rectangular matrix A represents a mapping from $\mathbf{C}^n$ to $\mathbf{C}^m$. The theorem shows that, there is an unitary basis in each of these two spaces, with respect to which this mapping is represented by a generalized diagonal matrix Σ.

For the sake of generality we have formulated the SVD for a complex matrix although in most applications A is real. If $A \in \mathbf{R}^{m\times n}$, then U and V are real orthogonal matrices.

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

$$Av_j = \sigma_j u_j, \qquad A^H u_j = \sigma_j v_j, \qquad j = 1, \ldots, r. \tag{7.1.41}$$

If U and V are partitioned according to

$$U = (U_1,\ U_2), \quad U_1 \in \mathbf{C}^{m\times r}, \quad V = (V_1,\ V_2), \quad V_1 \in \mathbf{C}^{n\times r}. \tag{7.1.42}$$

then the SVD can be written in the compact form

$$A = U_1 \Sigma_1 V_1^H = \sum_{i=1}^{r} \sigma_i u_i v_i^H. \tag{7.1.43}$$

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

From (6.2.20) it follows that

$$A^H A = V \Sigma^H \Sigma V^H, \qquad AA^H = U \Sigma \Sigma^H U^H.$$

Thus, σ_j^2, $j = 1, \ldots, r$ are the nonzero eigenvalues of the Hermitian and positive semi-definite matrices $A^H A$ and AA^H, and v_j and u_j are the corresponding eigenvectors. Hence, in principle the SVD can be reduced to the eigenvalue problem for Hermitian matrices. For a proof of the SVD using this relationship see Stewart [1973, p. 319]. *However, this does not lead to a numerically stable way to compute the SVD since small singular of A will not be determined accurately.*

Definition 7.1.12.

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

$$\mathcal{R}(A) = \{ y \in \mathbf{C}^m |\ y = Ax,\ x \in \mathbf{C}^n \}. \tag{7.1.44}$$

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

$$\mathcal{N}(A) = \{x \in \mathbf{C}^n | \ Ax = 0\}. \tag{7.1.45}$$

of dimension $n - r$

Suppose that $x = Ay \in \mathcal{R}(A)$ and $z \in \mathcal{N}(A)$. Then $Az = 0$ and we have

$$x^H z = x^H z = y^H A^H z = y^H (Az)^H = 0,$$

i.e., x is orthogonal to z. Since the sum of the dimensions of the range and null space equals m, we find the well-known relations

$$\mathcal{R}(A)^\perp = \mathcal{N}(A^H), \qquad \mathcal{N}(A)^\perp = \mathcal{R}(A^H),$$

The SVD gives complete information about the four fundamental subspaces associated with A and A^H. It is easy to verify that the range of A and null space of A^H are given by

$$\mathcal{R}(A) = \mathcal{R}(U_1) \qquad \mathcal{N}(A^H) = \mathcal{R}(U_2) \tag{7.1.46}$$
$$\mathcal{R}(A^H) = \mathcal{R}(V_1) \qquad \mathcal{N}(A) = \mathcal{R}(V_2). \tag{7.1.47}$$

The following relationship between the SVD and a symmetric eigenvalue problem plays in an important role in the development of stable algorithm for computing the SVD.

Theorem 7.1.13.
Let the SVD of $A \in \mathbf{R}^{m\times n}$ *be* $A = U\Sigma V^H$, *where* $U = \mathbf{R}^{m\times m}$ *and* $V \in \mathbf{R}^{n\times n}$ *are unitary. Let* $r = \operatorname{rank}(A) \le \min(m, n)$ *and* $\Sigma_1 = \operatorname{diag}(\sigma_1, \dots, \sigma_r) > 0$. *Then it holds that*

$$C = \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} = Q \begin{pmatrix} \Sigma & 0 & 0 \\ 0 & -\Sigma & 0 \\ 0 & 0 & 0 \end{pmatrix} Q^H, \tag{7.1.48}$$

where

$$Q = \frac{1}{\sqrt{2}} \begin{pmatrix} U_1 & U_1 & \sqrt{2}\,U_2 & 0 \\ V_1 & -V_1 & 0 & \sqrt{2}\,V_2 \end{pmatrix}. \tag{7.1.49}$$

and U *and* V *have been partitioned conformally. Hence, the eigenvalues of* C *are* $\pm\sigma_1, \pm\sigma_2, \dots, \pm\sigma_r$, *and zero repeated* $(m + n - 2r)$ *times.*

Proof. Form the product on the right hand side of (7.1.48) and note that $A = U_1 \Sigma_1 V_1^H$ and $A^H = V_1 \Sigma_1 U_1^H$. $\square$

Note that the matrix C in (7.1.48) satisfies

$$C^2 = \begin{pmatrix} A^H A & 0 \\ 0 & AA^H \end{pmatrix}, \tag{7.1.50}$$

that is, C^2 is a block diagonal matrix. Such a matrix is called **two-cyclic**. This shows the relation

In many problems there is a need to compute the SVD of an 2×2 upper triangular matrix. Even though this seems to be an easy task, care must be taken to avoid underflow and overflow.

Lemma 7.1.14.

The singular values of the upper triangular 2×2 matrix

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

are given by

$$\sigma_{1,2} = \frac{1}{2}\left|\sqrt{(r_{11}+r_{22})^2 + r_{12}^2} \pm \sqrt{(r_{11}-r_{22})^2 + r_{12}^2}\,\right|, \tag{7.1.51}$$

The largest singular value is computed using the plus sign, and the smaller is then obtained from

$$\sigma_2 = |r_{11}r_{22}|/\sigma_1. \tag{7.1.52}$$

Proof. The eigenvalues of the matrix

$$S = R^T R = \begin{pmatrix} r_{11}^2 & r_{11}r_{12} \\ r_{11}r_{12} & r_{12}^2 + r_{22}^2 \end{pmatrix}.$$

are $\lambda_1 = \sigma_1^2$ and $\lambda_2 = \sigma_2^2$. It follows that

$$\begin{aligned} \sigma_1^2 + \sigma_2^2 &= \mathrm{trace}\,(S) = r_{11}^2 + r_{12}^2 + r_{22}^2, \\ (\sigma_1\sigma_2)^2 &= \det(S) = (r_{11}r_{11})^2. \end{aligned}$$

It is easily verified that these relations are satisfied by the singular values given by (7.1.51). $\square$

The formulas above should not be used precisely as written. An algorithm that also computes orthogonal left and right singular vectors and guards against overflow and underflow has

The following program computes the SVD of a 2×2 upper triangular matrix with $|r_{11}| \geq |r_{22}|$

$$\begin{pmatrix} c_u & s_u \\ -s_u & c_u \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix} \begin{pmatrix} c_v & -s_v \\ s_v & c_v \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}. \tag{7.1.53}$$

$$\begin{aligned} &[c_u, s_u, c_v, s_v, \sigma_1, \sigma_2] = \mathrm{svd}(r_{11}, r_{12}, r_{22}) \\ &\quad l = (|r_{11}| - |r_{22}|)/|r_{11}|; \\ &\quad m = r_{12}/r_{11}; \quad t = 2 - l; \end{aligned}$$

$$
\begin{aligned}
&s = \sqrt{t^2 + m^2}; \quad r = \sqrt{l^2 + m^2};\\
&a = 0.5(s + r);\\
&\sigma_1 = |r_{11}|a; \quad \sigma_2 = |r_{22}|/a;\\
&t = (1 + a)(m/(s + t) + m/(r + l));\\
&l = \sqrt{t^2 + 4};\\
&c_v = 2/l; \quad s_v = -t/l;\\
&c_u = (c_v - s_v m)/a; \quad s_u = s_v(r_{22}/r_{11})/a;\\
&\text{end}
\end{aligned}
$$

Even this program can be made to fail. A Fortran program that always gives high *relative accuracy* in the singular values and vectors has been developed by Demmel and Kahan. It was briefly mentioned in [96] but not listed there. A listing of the Fortran program and a sketch of its error analysis is found in an appendix of [14].

7.1.5 Norms of Vectors and Matrices

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

Definition 7.1.15.

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

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

The triangle inequality is often used in the form (see Problem 12)

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

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

$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}, \qquad 1 \leq p < \infty. \tag{7.1.54}$$

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

$$
\begin{aligned}
\|x\|_1 &= |x_1| + \cdots + |x_n|,\\
\|x\|_2 &= (|x_1|^2 + \cdots + |x_n|^2)^{1/2} = (x^H x)^{1/2},\\
\|x\|_\infty &= \max_{1\leq i\leq n} |x_i|.
\end{aligned} \tag{7.1.55}
$$

A vector norm$\|\cdot\|$ is called **absolute** if $\|x\| = \|\,|x|\,\|$, and **monotone** if $|x| \le |y| \Rightarrow \|x\| \le \|y\|$. It can be shown that a vector norm is monotone if and only if it is absolute; see Stewart and Sun [348, Theorem II.1.3]. Clearly the vector p-norms are absolute for all $1 \le p < \infty$.

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

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

if Q is unitary.

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

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

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

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

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

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

$$|x^H y| \le \|x\|_p \|y\|_q, \quad \frac{1}{p} + \frac{1}{q} = 1, \quad p \ge 1. \tag{7.1.56}$$

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

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

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

$$|x^H y| = \left|\sum_{i=1}^n x_i^H y_i\right| \le \sum_{i=1}^n |x_i^H y_i| \le \max_i |y_i| \sum_{i=1}^n |x_i| = \|x\|_1 \|y\|_\infty. \tag{7.1.57}$$

Definition 7.1.16.

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

$$\|x\|_D = \max_{y \ne 0} |x^H y| / \|y\|. \tag{7.1.58}$$

The vectors in the set

$$\{y \in \mathbf{C}^n \mid \|y\|_D \|x\| = y^H x = 1\}. \tag{7.1.59}$$

are said to be **dual vectors** *to x with respect to $\|\cdot\|$.*

It can be shown that the dual of the dual norm is the original norm (see [348, Theorem II.1.12]). It follows from the Hölder inequality that the dual of the p-norm is the q-norm, where

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

The dual of the 2-norm can be seen to be itself. It can be shown to be the only norm with this property (see [215, Theorem 5.4.16]).

The vector 2-norm can be generalized by taking

$$\|x\|_{2,G}^2 = (x, Gx) = x^H Gx, \tag{7.1.60}$$

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

$$\|x\|_{p,D} = \|Dx\|_p, \quad D = \mathrm{diag}\,(d_1, \ldots, d_n), \quad d_i \neq 0, \quad i = 1 : n. \tag{7.1.61}$$

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

$$\frac{1}{c}\|x\|' \leq \|x\| \leq c'\|x\|' \quad \forall x \in \mathbf{C}^n. \tag{7.1.62}$$

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

$$\|x\|_q \leq \|x\|_p \leq n^{(1/p-1/q)}\|x\|_q, \quad 1 \leq p \leq q \leq \infty. \tag{7.1.63}$$

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

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

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

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|. \tag{7.1.64}$$

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

$$\|Ax\| \leq \|A\|\,\|x\| \quad \forall x \in \mathbf{C}^n. \tag{7.1.65}$$

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

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

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

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

Theorem 7.1.17.

For $p = 1, 2\ \infty$ the matrix subordinate p-norm are given by

$$\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|, \tag{7.1.66}$$

$$\|A\|_2 = \max_{\|x\|=1} (x^H A^H A x)^{1/2} = \sigma_1(A), \tag{7.1.67}$$

$$\|A\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|. \tag{7.1.68}$$

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

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

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

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

and hence $\|A\|_1 \ge \max_{1\le j\le n} \|a_j\|_1$. This implies (7.1.66). The formula (7.1.68) for the matrix ∞-norm is proved in a similar fashion. The expression for the 2-norm follows from the extremal property

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

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

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

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

$$\|A\|_2 \le (\|A\|_1 \|A\|_\infty)^{1/2}. \tag{7.1.69}$$

The proof of this bound is given as an exercise in Problem 17.

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

Definition 7.1.18.
The **Frobenius norm**[11] *is derived from the vector* 2*-norm*

$$\|A\|_F = \Big(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2\Big)^{1/2} \tag{7.1.70}$$

The Frobenius norm is submultiplicative, but is often larger than necessary, e.g., $\|I_n\|_F = n^{1/2}$. This tends to make bounds derived in terms of the Frobenius norm not as sharp as they might be. From (7.1.75) it follows that

$$\frac{1}{\sqrt{n}}\|A\|_F \le \|A\|_2 \le \|A\|_F, \quad k = \min(m,n). \tag{7.1.71}$$

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

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

$$\operatorname{trace}(A) = \sum_{i=1}^{n} a_{ii}. \tag{7.1.72}$$

Useful properties are of this function are

$$\operatorname{trace}(S^{-1}AS) = \operatorname{trace}(A), \tag{7.1.73}$$
$$\operatorname{trace}(AB) = \operatorname{trace}(BA). \tag{7.1.74}$$

An alternative characterization of the Frobenius norm is

$$\|A\|_F^2 = \operatorname{trace}(A^H A) = \sum_{i=1}^{k}\sigma_i^2(A), \quad k = \min(m,n), \tag{7.1.75}$$

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

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

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

$$\|QAP\| = \|A\|. \tag{7.1.76}$$

Such norms are called **unitarily invariant** and have an interesting history; see Stewart and Sun [348, Sec. II.3].

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

Table 7.1.1. *Numbers* γ_{pq} *such that* $\|A\|_p \le \gamma_{pq}\|A\|_q$, *where* $A \in \mathbf{C}^{m\times n}$ *and* $\operatorname{rank}(A) = r$.

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

Theorem 7.1.19.

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

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

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

Unitarily invariant norms were characterized by von Neumann [289, 1937], who showed that the converse of Theorem 7.1.19 is true: A function $\Phi(\sigma_1, \ldots, \sigma_n)$ which is symmetric in its arguments and satisfies the three properties in the Definition 7.1.15 of a vector norm defines a unitarily invariant matrix norm. In this connection such functions are called **symmetric gauge functions**. Two examples are

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

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

$$x_k = \begin{pmatrix} \xi_1^{(k)} & \xi_2^{(k)} \cdots \xi_n^{(k)} \end{pmatrix}, \quad k = 1, 2, 3 \ldots.$$

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

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

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

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

For a *finite dimensional vector space* it follows from the equivalence of norms that convergence is independent of the choice of norm. The particular choice

$\|\cdot\|_\infty$ shows that convergence of vectors in $\mathbf{C}^n$ is equivalent to convergence of the n sequences of scalars formed by the components of the vectors. By considering matrices in $\mathbf{C}^{m\times n}$ as vectors in $\mathbf{C}^{mn}$ the same conclusion holds for matrices.

7.1.6 Matrix Multiplication

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

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

In most matrix computations the number of floating-point multiplicative operations $(\times, /)$ is usually about the same as the number of additive operations $(+, -)$. Therefore, in older literature, a **flop** was defined to mean roughly the amount of work associated with the floating-point computation

$$s := s + a_{ik}b_{kj},$$

i.e. one addition *and* one multiplication (or division). It is usually ignored that on many computers a division is 5–10 times slower than a multiplication. In more recent textbooks (e.g., Golub and Van Loan [184, 1996]) a flop is defined as one floating-point operation doubling the older flop counts.[12] Hence, , multiplication $C = AB$ of two square matrices of order n requires $2n^3$ flops. The matrix-vector multiplication $y = Ax$, where $A \in \mathbf{R}^{n\times n}$ and $x \in \mathbf{R}^n$, requires $2mn$ flops.[13]

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

[12] Stewart [346, p. 96] uses **flam** (floating-point addition and multiplication) to denote an "old" flop.

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

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

$$C = AB = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{pmatrix} (b_1\ b_2\ \cdots\ b_n) = (c_{ij}), \qquad c_{ij} = a_i^T b_j. \tag{7.1.77}$$

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

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

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

$$C = AB = (a_1\ a_2\ \cdots\ a_p) \begin{pmatrix} b_1^T \\ b_2^T \\ \vdots \\ b_p^T \end{pmatrix} = \sum_{k=1}^{p} a_k b_k^T, \tag{7.1.78}$$

where each term in the sum of (7.1.78) is an *outer product*. A code expressing this is

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

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

A faster method for matrix multiplication would give more efficient algorithms for many linear algebra problems such as inverting matrices, solving linear systems and eigenvalue problems. An intriguing question is whether it is possible to multiply two matrices $A, B \in \mathbf{R}^{n\times n}$ (or solve a linear system of order n) in less than n^3 (scalar) multiplications. The answer is yes!

Strassen [351, 1969] developed a fast algorithm for matrix multiplication. If used recursively to multiply two square matrices of dimension $n = 2^k$, then the number of multiplications is reduced from n^3 to $n^{\log_2 7} = n^{2.807\dots}$; see Sec. 7.5.3. (The number of additions is of the same order.)

7.1.7 Floating-Point Arithmetic

The IEEE 754 standard (see [135, 1985]) for floating-point arithmetic is now used for all personal computers and workstations. It specifies basic and extended formats for floating-point numbers, elementary operations and rounding rules available, conversion between different number formats, and binary-decimal conversion. The handling of exceptional cases like exponent overflow or underflow and division by zero are also specified.

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

$$v = (-1)^s (1.m)_2 2^e, \quad -e_{\min} \le e \le e_{\max}. \tag{7.1.79}$$

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

Four rounding modes are supported by the standard. The default rounding mode is round to nearest representable number, with round to even in case of a tie.(Some computers in case of a tie round away from zero, i.e. raise the absolute value of the number, because this is easier to realize technically.) Chopping is also supported as well as directed rounding to ∞ and to $-\infty$. The latter mode simplifies the implementation of interval arithmetic.

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

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

The largest number that can be represented is approximately $2.0 \cdot 2^{127} \approx 3.4028 \times 10^{38}$ in single precision and $2.0 \cdot 2^{1023} \approx 1.7977 \times 10^{308}$ in double precision. The smallest normalized number is $1.0 \cdot 2^{-126} \approx 1.1755 \times 10^{-38}$ in single precision and $1.0 \cdot 2^{-1022} \approx 2.2251 \times 10^{-308}$ in double precision. For more details on the IEEE floating-point standard and floating point arithmetic, see Volume I, Sec. 2.3.

If x and y are floating-point numbers, we denote by

$$fl\,(x+y), \quad fl\,(x-y), \quad fl\,(x \cdot y), \quad fl\,(x/y)$$

the results of floating addition, subtraction, multiplication, and division, which the machine stores in memory (after rounding or chopping). If underflow or overflow does not occur, then in IEEE floating-point arithmetic, it holds that

$$fl\,(x \text{ op } y) = (x \text{ op } y)(1+\delta), \quad |\delta| \le u, \tag{7.1.80}$$

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

$$fl\,(\sqrt{x}) = \sqrt{x}(1+\delta), \quad |\delta| \le u, \tag{7.1.81}$$

Bounds for roundoff errors for basic vector and matrix operations can easily be derived (Wilkinson [387, pp. 114–118]) using the following basic result:

Lemma 7.1.20. [Higham [211, Lemma 3.1]]

Let $|\delta_i| \le u$, $\rho_i = \pm 1$, $i = 1{:}n$, *and*

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

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

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

$$\bar{\gamma}_k = \frac{cku}{1 - cku/2}, \tag{7.1.82}$$

where c denotes a small integer constant.

For an **inner product** $x^T y$ computed in the natural order we have

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

where

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

The corresponding forward error bound becomes

$$|fl\,(x^T y) - x^T y| < \sum_{i=1}^{n} \gamma_{n+2-i}|x_i||y_i| < \gamma_n |x^T||y|, \tag{7.1.83}$$

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

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

$$|fl\,(xy^T) - xy^T| \le u\,|xy^T|. \tag{7.1.84}$$

This is a satisfactory result for many purposes, but the computed result is not in general a rank one matrix and it is not possible to find perturbations Δx and Δy such that $fl(xy^T) = (x+\Delta x)(x+\Delta y)^T$. This shows that matrix multiplication in floating-point arithmetic is not always backward stable!

Similar error bounds can easily be obtained for matrix multiplication. Let $A \in \mathbf{R}^{m\times n}$, $B \in \mathbf{R}^{n\times p}$, and denote by $|A|$ and $|B|$ matrices with elements $|a_{ij}|$ and $|b_{ij}|$. Then it holds that

$$|fl\,(AB) - AB| < \gamma_n |A||B|. \tag{7.1.85}$$

where the inequality is to be interpreted elementwise. Often we shall need bounds for some norm of the error matrix. From (7.1.85) it follows that

$$\|fl\,(AB) - AB\| < \gamma_n \|\,|A|\,\|\,\|\,|B|\,\|. \tag{7.1.86}$$

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

$$\|fl\,(AB) - AB\| < \gamma_n \|A\|\,\|B\|. \tag{7.1.87}$$

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

$$\|fl\,(AB) - AB\|_2 < n\gamma_n \|A\|_2\,\|B\|_2. \tag{7.1.88}$$

In many matrix algorithms there repeatedly occurs expressions of the form

$$y = \Big(c - \sum_{i=1}^{k-1} a_i b_i\Big)\Big/ d.$$

A simple extension of the roundoff analysis of an inner product in Sec, 2.4.1 (cf. Problem 2.3.7) shows that if the term c is added last, then the computed $\bar{y}$ satisfies

$$\bar{y}d(1+\delta_k) = c - \sum_{i=1}^{k-1} a_i b_i (1+\delta_i), \tag{7.1.89}$$

where

$$|\delta_1| \le \gamma_{k-1}, \qquad |\delta_i| \le \gamma_{k+1-i}, \quad i = 2 : k-1, \qquad |\delta_k| \le \gamma_2. \tag{7.1.90}$$

and $\gamma_k = ku/(1-ku)$ and u is the unit roundoff. Note that in order to prove a backward error result for Gaussian elimination, that does not perturb the right hand side vector b, we have formulated the result so that c is not perturbed. It follows that the forward error satisfies

$$\Big|\bar{y}d - c + \sum_{i=1}^{k-1} a_i b_i\Big| \le \gamma_k \Big(|\bar{y}d| + \sum_{i=1}^{k-1} |a_i||b_i|\Big), \tag{7.1.91}$$

and this inequality holds independent of the summation order.

7.1.8 Complex Matrix Computations

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

$$\begin{aligned} x \pm y &= a \pm c + i(b \pm d), \\ x \times y &= (ac - bd) + i(ad + bc), \\ x/y &= \frac{ac + bd}{c^2 + d^2} + i\frac{bc - ad}{c^2 + d^2}, \end{aligned} \tag{7.1.92}$$

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

Lemma 7.1.21.

Assume that the standard model (7.1.80) *for floating point arithmetic holds. Then, provided that no overflow or underflow occurs, no denormalized numbers are produced, the complex operations computed according to* (7.1.92) *satisfy*

$$\begin{aligned} fl\,(x \pm y) &= (x \pm y)(1+\delta), \quad |\delta| \le u, \\ fl\,(x \times y) &= x \times y(1+\delta), \quad |\delta| \le \sqrt{5}u, \\ fl\,(x/y) &= x/y(1+\delta), \quad |\delta| \le \sqrt{2}\gamma_4, \end{aligned} \tag{7.1.93}$$

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

Proof. See Higham [211, Sec. 3.6]. The result for complex multiplication is due to Brent et al. [51, 2007]. □

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

$$u = \left(\frac{r+x}{2}\right)^{1/2}, \qquad v = \left(\frac{r-x}{2}\right)^{1/2}, \quad r = \sqrt{x^2+y^2}. \tag{7.1.94}$$

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

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

so v can be computed from $v = y/(2u)$. When $x < 0$ we instead compute v from (7.1.94) and set $u = y/(2v)$.

Most rounding error analysis given in this book are formulated for real arithmetic. Since the bounds in Lemma 7.1.21 are of the same form as the standard model for real arithmetic, these can simply be extended to complex arithmetic.

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

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + b \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{7.1.95}$$

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

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

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

$$\begin{pmatrix} a_1 & -b_1 \\ b_1 & a_1 \end{pmatrix} \begin{pmatrix} a_2 & -b_2 \\ b_2 & a_2 \end{pmatrix} = \begin{pmatrix} a_1a_2 - b_1b_2 & -(a_1b_2 + b_1a_2) \\ a_1b_2 + b_1a_2 & a_1a_2 - b_1b_2 \end{pmatrix}. \tag{7.1.96}$$

and is the representation of the complex number $(a_1+ib_1)(a_2+ib_2)$. Every nonzero matrix of this form is invertible and its inverse is again of the same form. The matrices of the form (7.1.95) are therefore afield isomorphic to the field of complex numbers. Note that the complex scalar $u = \cos\theta + i\,\sin\theta = e^{\theta}$ on the unit circle corresponds to the orthogonal matrix

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

which represents a counter-clockwise rotation of θ; see Sec. 8.1.6.

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

$$\tilde{c}_{ij} = \begin{pmatrix} a_{ij} & -b_{ij} \\ b_{ij} & a_{ij} \end{pmatrix}. \tag{7.1.97}$$

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

An obvious drawback of the outlined scheme is that the representation (7.1.97) is redundant and doubles the memory requirement, since the real and imaginary parts are stored twice. To some extent this drawback can easily be circumvented. Consider the multiplication of two complex numbers in (7.1.96). Clearly we only need to compute the first column in the product

$$\begin{pmatrix} a_1 & -b_1 \\ b_1 & a_1 \end{pmatrix}\begin{pmatrix} a_2 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1a_2 - b_1b_2 \\ a_1b_2 + b_1a_2 \end{pmatrix},$$

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

Example 7.1.3.

An alternative representation of complex matrices is as follows. Consider the complex linear system

$$(A + i\,B)(x + i\,y) = c + i\,d.$$

If we separate the real and imaginary parts we obtain the real linear system

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

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

$$C = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}, \tag{7.1.98}$$

This matrix is related to the previous matrix $\tilde{C}$ by a permutation of rows and columns. Note that the structure of the product of two such matrices is again a block matrix with 2×2 blocks of this structure. In (7.1.97) the real part corresponds to the diagonal (symmetric) matrix $a_{ij}I_2$ the imaginary part corresponds to the skew symmetric part.

Review Questions

1.1 Define the concepts:
(i) Real symmetric matrix. (ii) Real orthogonal matrix.
(iii) Real skew-symmetric matrix. (iv) Triangular matrix.
(v) Hessenberg matrix.

1.2 (a) What is the outer product of two column vectors x and y?
(b) How is the Hadamard product of two matrices A and B defined?

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

1.4 Verify the formulas (7.1.19) for the inverse of a 2×2 block triangular matrices.

1.5 What can the Woodbury formula be used for?

1.6 (a) What is meant by a flop in this book? Are there any other definitions used?
(b) How many flops are required to multiply two matrices in $\mathbf{R}^{n\times n}$ if the standard definition is used. Is this scheme optimal?

1.7 (a) Given a vector norm define the matrix subordinate norm.
(b) Give explicit expressions for the matrix p norms for $p = 1, 2, \infty$.

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

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

which are special cases of (7.1.63).

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

Problems

1.1 Show that if $R \in \mathbf{R}^{n\times n}$ is strictly upper triangular, then $R^n = 0$.

1.2 (a) Let $A \in \mathbf{R}^{m\times p}$, $B \in \mathbf{R}^{p\times n}$, and $C = AB$. Show that the column space of C is a subspace of the column space of A, and the row space of C is a subspace of the row space of B.

(b) Show that the rank of a sum and product of two matrices satisfy

$$\operatorname{rank}(AB) \le \min\{\operatorname{rank}(A), \operatorname{rank}(B)\}.$$

1.3 If A and B are square upper triangular matrices show that AB is upper triangular, and that A^{-1} is upper triangular if it exists. Is the same true for lower triangular matrices?

1.4 To solve a linear system $Ax = b$, $A \in \mathbf{R}^n$, by Cramer's rule requires the evaluation of $n+1$ determinants of order n (see (7.1.33)). Estimate the number of multiplications needed for $n = 50$ if the determinants are evaluated in the naive way. Estimate the time it will take on a computer performing 10^9 floating-point operations per second!

1.5 (a) Show that the product of two Hermitian matrices is symmetric if and only if A and B commute, that is, $AB = BA$.

1.6 Let $A \in \mathbf{R}^{n\times n}$ be a given matrix. Show that if $Ax = y$ has *at least one* solution for any $y \in \mathbf{R}^n$, then it has *exactly one* solution for any $y \in \mathbf{R}^n$. (This is a useful formulation for showing uniqueness of approximation formulas.)

1.7 Let $A \in \mathbf{R}^{m\times n}$ have rows a_i^T, i.e., $A^T = (a_1, \dots, a_m)$. Show that

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

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

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

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

is orthogonal. This is known as the **Cayley transform**,
(b) Verify the special 2×2 case

$$S = \begin{pmatrix} 0 & \tan\frac{\theta}{2} \\ -\tan\frac{\theta}{2} & 0 \end{pmatrix} \qquad Q = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

where $0 \le \theta < \pi$.

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

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

1.10 Prove that the following inequalities are valid and best possible:

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

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

1.11 Show that any vector norm is uniformly continuous by proving the inequality

$$|\,\|x\| - \|y\|\,| \le \|x-y\|, \qquad x, y \in \mathbf{R}^n.$$

1.12 Show that for any matrix norm there exists a consistent vector norm.
Hint: Take $\|x\| = \|xy^T\|$ for any vector $y \in \mathbf{R}^n$, $y \neq 0$.

1.13 Derive the formula for $\|A\|_\infty$ given in Theorem 7.1.17.

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

1.15 Prove that for any subordinate matrix norm

$$\|A+B\| \le \|A\| + \|B\|, \qquad \|AB\| \le \|A\|\|B\|.$$

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

1.17 Use the result

$$\|A\|_2^2 = \rho(A^TA) \le \|A^TA\|,$$

valid for any matrix operator norm $\|\cdot\|$, where $\rho(A^TA)$ denotes the spectral radius of A^TA, to deduce the upper bound in (7.1.69).

1.18 Prove the expression (7.1.68) for the matrix norm subordinate to the vector ∞-norm.

1.19 (a) Let T be a nonsingular matrix, and let $\|\cdot\|$ be a given vector norm. Show that the function $N(x) = \|Tx\|$ is a vector norm.
(b) What is the matrix norm subordinate to $N(x)$?
(c) If $N(x) = \max_i |k_i x_i|$, what is the subordinate matrix norm?

1.20 Consider an upper block triangular matrix

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

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

1.21 Use the Woodbury formula to prove the identity

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

1.22 (a) Let A^{-1} be known and let B be a matrix coinciding with A except in one row. Show that if B is nonsingular then B^{-1} can be computed by about $2n^2$ multiplications using the Sherman–Morrison formula (7.1.25).
(b) Use the Sherman–Morrison formula to compute B^{-1} if

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

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

1.23 Show that the eigenvalues of the matrix

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

are equal to $a \pm ib$.

1.24 Show that if $C = A + i\,B$ be a complex nonsingular matrix, then $C^{-1} = A_1 - i\,B_1$ where

$$A_1 = (A + BA^{-1}B)^{-1}, \qquad B_1 = A^{-1}BA_1 = A_1BA^{-1}. \tag{7.1.99}$$

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

1.25 The complex unitary matrix $U = Q_1 + i\,Q_2$ and its conjugate transpose $U^H = Q_1 - i\,Q_2$ can be represented by the real matrices

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

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

7.2 Gaussian Elimination

The closer one looks, the more subtle and remarkable Gaussian elimination appears.
—Lloyd N. Trefethen, Three mysteries of Gaussian elimination. SIGNUM Newsl. (1985).

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

We recall that a linear system $Ax = b$, $A \in \mathbf{R}^{m\times n}$, is **consistent** if and only if $b \in \mathcal{R}(A)$, or equivalently $\operatorname{rank}(A,\ b) = \operatorname{rank}(A)$. Otherwise it is inconsistent and has no solution. Such a system is said to be **overdetermined**. If $\operatorname{rank}(A) = m$ then $\mathcal{R}(A)$ equals $\mathbf{R}^m$ and the system is consistent for all $b \in \mathbf{R}^m$. Clearly a consistent linear system always has *at least one solution* x. The corresponding homogeneous linear system $Ax = 0$ is satisfied by any $x \in \mathcal{N}(A)$ and thus has $(n - r)$ linearly independent solutions. Such a system is said to be **underdetermined** It follows that if a solution to an inhomogeneous system $Ax = b$ exists, it is unique only if $r = n$, whence $\mathcal{N}(A) = \{0\}$.

Underdetermined and overdetermined systems arise quite frequently in practice! Problems where there are more parameters than needed to span the right hand side lead to underdetermined systems. In this case we need additional information in order to decide which solution to pick. On the other hand, overdetermined systems arise when there is more data than needed to determine the solution. In this case the system is inconsistent and has no solution. Underdetermined and overdetermined systems are best treated using *orthogonal* transformations rather than Gaussian elimination. and will be treated in Chapter 8.

7.2.1 Triangular Systems and Gaussian Elimination

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

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

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

Triangular matrices have several nice properties. It is easy to verify that sums, products and inverses of square upper (lower) triangular matrices are again triangular matrices of the same type. The diagonal elements of the product $U = U_1U_2$ of two triangular matrices are just the product of the diagonal elements in U_1 and U_2. From this it follows that the diagonal elements in U^{-1} are the inverse of the diagonal elements in U.

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

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

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

Similarly, a lower triangular matrix has form

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

The solution of a lower triangular linear system $Ly = c$, can be computed by **forward-substitution**.

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

When implementing a matrix algorithm such as (7.2.1) or (7.2.2) on a computer, the *order of operations* in matrix algorithms may be important. One reason for this is the economizing of storage, since even matrices of moderate dimensions have a large number of elements. When the initial data is not needed for future use, computed quantities may overwrite data. To resolve such ambiguities in the description of matrix algorithms it is important to be able to describe computations in a more precise form. For this purpose we will either use MATLAB or an informal programming language, which is sufficiently precise for our purpose but allows the suppression of cumbersome details. The back-substitution (7.2.1) is implemented in the following MATLAB function.

Algorithm 7.1. *Back-substitution.*

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

```
function x = trisu(U,b)
% TRISU solves the upper triangular system
% Ux = b by backsubstitution
n = length(b);
x = zeros(n,1);
for i = n:-1:1
   s = 0;
   for k = i+1:n
      s = s + U(i,k)*x(k);
   end
   x(i) = (b(i) - s)/U(i,i);
 end
```

Here the elements in U are accessed in row-wise order. Note that in order to minimize round-off errors b_i is added *last* to the sum; compare the error bound (2.4.3).

By changing the order of the two loops above a column-oriented algorithm is obtained.

```
for k = n:-1:1
   x(k) = b(k)/U(k,k);
   for i = k-1:-1:1
      x(i) = x(i) + U(i,k)*x(k);
   end
end
```

Here the elements in U are accessed column-wise instead of row-wise as in the previous algorithm. Such differences can influence the efficiency when implementing matrix algorithms. For example, if U is stored column-wise as is the convention in Fortran, the second version is to be preferred.

In older textbooks a **flop** was often used to mean roughly the amount of work associated with the computation

$$s := s + a_{ik}b_{kj},$$

i.e., one floating-point addition and multiplication and some related subscript computation. More recently (see, e.g., Higham [211, 2002]) a flop is instead defined as a floating-point add *or* multiply. With this notation solving a triangular system requires n^2 flops.

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

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}. \tag{7.2.3}$$

A fundamental observation is that the following elementary operation can be performed on the system without changing the set of solutions:

- Adding a multiple of the ith equation to the jth equation.
- Interchange two equations.

These correspond in an obvious way to row operations on the augmented matrix $(A,\, b)$. It is also possible to interchange two columns in A provided we make the corresponding interchanges in the components of the solution vector x. We say that the modified system is **equivalent** to the original system.

The idea behind **Gaussian elimination**[14] is to use such elementary operations in a systematic way to eliminate the unknowns in the system $Ax = b$, so that at the end an equivalent upper triangular system is produced, which is then solved by back-substitution. If $a_{11} \neq 0$, then in the first step we eliminate x_1 from the last $(n-1)$ equations by subtracting the multiple

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

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

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

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

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)}, \qquad b_i^{(k+1)} = b_i^{(k)} - l_{ik}b_1^{(k)}, \qquad i = k+1 : n, \quad j = k+1 : m. \tag{7.2.4}$$

where the multipliers are

$$l_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}, \quad i = k+1 : n. \tag{7.2.5}$$

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

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

$$A^{(k)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots & & \vdots \\ & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & & \vdots & & \vdots \\ & & & a_{mk}^{(k)} & \cdots & a_{mn}^{(k)} \end{pmatrix}, \qquad b^{(k)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_k^{(k)} \\ \vdots \\ b_m^{(k)} \end{pmatrix}, \tag{7.2.6}$$

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

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

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

Let A_k denote the kth leading principal submatrix of A. Since the determinant of a matrix does not change under row operations the determinant of A_k equals the product of the diagonal elements then by (7.2.7)

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

For a square system, i.e. $m = n$, this implies that all pivotal elements $a_{ii}^{(i)}$, $i = 1 : n$, in Gaussian elimination are nonzero if and only if $m = n$ and $\det(A_k) \neq 0$, $k = 1 : n$. In this case we can continue the elimination until after $(n-1)$ steps we get the single equation

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

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

$$\det(A) = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}. \tag{7.2.7}$$

If in Gaussian elimination a zero pivotal element is encountered, i.e. $a_{kk}^{(k)} = 0$ for some $k \leq n$ then we cannot proceed. If A is square and nonsingular, then in particular its first k columns are linearly independent. This must also be true for the first k columns of the reduced matrix. Hence, some element $a_{ik}^{(k)}$, $i \geq k$ must be nonzero, say $a_{pk}^{(k)} \neq 0$. By interchanging rows k and p this element can be taken as pivot and it is possible to proceed with the elimination. Note that when rows are interchanged in A the same interchanges must be made in the elements of the vector b. Note that also the determinant formula (7.2.7) must be modified to

$$\det(A) = (-1)^s a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}, \tag{7.2.8}$$

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

In the general case, suppose that $a_{ik}^{(k)} = 0$. If the entire submatrix $a_{ij}^{(k)}$, $i, j = k : n$, is zero, then $\operatorname{rank}(A) = k$ and we stop. Otherwise there is a nonzero element, say $a_{pq}^{(k)} \neq 0$, which can be brought into pivoting position by interchanging rows k and p and columns k and q. (Note that when columns are interchanged in A the same interchanges must be made in the elements of the solution vector x.) Proceeding in this way any matrix A can always be reduced in $r = \operatorname{rank}(A)$ steps to upper **trapezoidal form**,

Theorem 7.2.1.

Let $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$, and consider the linear system $Ax = b$ of m equations in n unknowns $x \in \mathbf{R}^n$. After $r = \operatorname{rank}(A)$ elimination steps and row and column permutations the system $Ax = b$ can be transformed into an equivalent system

$$A^{(r)} = \left(\begin{array}{cccc|ccc} a_{11}^{(1)} & \cdots & a_{1r}^{(1)} & a_{1,r+1}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & \ddots & \vdots & & \vdots & \vdots \\ \vdots & & a_{rr}^{(r)} & a_{r,r+1}^{(r)} & \cdots & a_{rn}^{(r)} \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right), \qquad b^{(r)} = \left(\begin{array}{c} b_1^{(1)} \\ \vdots \\ b_r^{(r)} \\ \hline b_{r+1}^{(r+1)} \\ \vdots \\ b_m^{(r+1)} \end{array}\right) \tag{7.2.9}$$

with $a_{kk}^{(k)} \neq 0$, $k = 1 : r$. The eliminations and row permutations are also carried out on the right hand side b. From the reduced form (7.2.9) we can read off $r = \operatorname{rank}(A)$, which equals the number of linearly independent rows (columns) in A. The two rectangular blocks of zeros in $A^{(r)}$ have dimensions $(m-r)\times r$ and $(m-r)\times(n-r)$,

It becomes apparent when the matrix $A \in \mathbf{R}^{m\times n}$ is reduced to upper **trapezoidal form** that there are three different possibilities for a linear system:

1. If $r = n$ and the system is consistent there is a unique solution. If $m = n = r$ then the system is consistent.

2. If $r < n$ and the system is consistent then it has an infinite number of solutions. Arbitrary values can be assigned to the last $n - r$ components of (the possibly permuted) solution vector x. The first r components are then uniquely determined. Such a system is said to be **underdetermined**.

3. If $b_k^{(r+1)} \neq 0$, for some $k > r$, then the system is inconsistent and has no solution. Such a system is said to be **overdetermined**. We have to be content with finding an x such that the residual vector $r = b - Ax$ is small in some sense.

In theory the reduced trapezoidal form (7.2.9) yields the rank of a the matrix A. It also answers the question whether the given system is consistent or not. However, this is the case *only if exact arithmetic is used.* In floating-point calculations it may be difficult to decide if a pivot element, or an element in the transformed right hand side, should be considered as zero or not. For example, a zero pivot in exact arithmetic will almost invariably be polluted by roundoff errors in such a way that it equals some small nonzero number.

What tolerance to use in deciding when a pivot should be taken to be "numerically zero" will depend on the context. Clearly the "numerical rank" assigned to a matrix should depend on some tolerance, which reflects the error level in the data and/or the precision of the floating-point arithmetic used; see Sec. 7.6.1. In the rest of this chapter we assume that the matrix A is nonsingular unless otherwise stated.

Example 7.2.1.

The rank of the matrix

$$\tilde{A} = \begin{pmatrix} 1 & 1+10^{-8} \\ 1 & 1 \end{pmatrix}$$

equals two. But distance $\|\tilde{A} - A\|_2$ to a singular matrix, for example,

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

is at most 10^{-8}. If the elements in $\tilde{A}$ are uncertain, e.g., due to roundoff, then we should assign it the rank one.

Algorithm 7.2. *Gaussian Elimination.*

The following MATLAB function reduces a nonsingular linear system $Ax = b$ to the upper triangular form $Ux = c$ by Gaussian Elimination (7.2.5)–(7.2.4). It is assumed that all pivot elements $a_{kk}^{(k)}$, $k = 1 : n$, are nonzero.

```
function [U,c] = gauss(A,b);
% GAUSS reduces a nonsingular linear system Ax = b
% by Gaussian eliminationa to upper triangular form
% Ux = c
n = length(b);
for k = 1:n-1
   for i = k+1:n
      L(i,k) = A(i,k)/A(k,k); % multiplier
      for j = k+1:n
         A(i,j) = A(i,j) - L(i,k)*A(k,j);
      end
      b(i) = b(i) - L(i,k)*b(k);
   end
end
U = triu(A); c = b;
```

The final upper triangular system is solved by back-substitution

```
x = trisu(triu(A,0),b);
```

If the multipliers l_{ik} are saved, then the operations on the vector b can be deferred to a later stage. This observation is important in that it shows that *when solving a sequence of linear systems*

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

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

From Algorithm 7.2.1 it follows that $(n-k)$ divisions and $(n-k)^2$ multiplications and additions are used in step k to transform the elements of A. A further $(n-k)$ multiplications and additions are used to transform the elements of b. Summing over k and neglecting low order terms we find that the total number of flops required by Gaussian elimination is

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

for A and each right hand side respectively. Comparing with the approximately n^2 flops needed to solve a triangular system we conclude that, except for very small values of n, *the reduction of A to triangular form dominates the work.*[15]

Algorithm 7.2.1 for Gaussian Elimination algorithm has three nested loops. It is possible to reorder these loops in $3 \cdot 2 \cdot 1 = 6$ ways. Each of those versions perform the same basic operation

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

but in different order. The version given above uses row operations and may be called the "kij" variant, where k refers to step number, i to row index, and j to column index. This version is not suitable for Fortran 77, and other languages in which matrix elements are stored and accessed sequentially by columns. In such a language the form "kji" should be preferred, which is the column oriented variant of Algorithm 7.2.1 (see Problem 5).

We describe two enhancements of the function `gauss`. Note that when l_{ik} is computed the element $a_{ik}^{(k)}$ becomes zero and no longer takes part in the elimination. Thus, memory space can be saved by storing the multipliers in the lower triangular part of the matrix. Further, according to (7.2.4) in step k the elements $a_{ij}^{(k)}$ $i, j = k+1 : n$ are modified with the rank one matrix

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

The efficiency of the function `gauss` is improved if these two observations are incorporate by changing the two inner loops to

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

[15]This conclusion is not in general true for banded and sparse systems (see Sections 7.4 and 7.8, respectively).

7.2.2 LU Factorization

We now show that Gaussian elimination can be interpreted as computing the matrix factorization $A = LU$. The LU factorization is a prime example of the decompositional approach to matrix computation. This approach came into favor in the 1950s and early 1960s and has been named as one of the ten algorithms with most influence on science and engineering in the 20th century.

Assume that $m = n$ and that Gaussian elimination can be carried out without pivoting. We will show that in this case Gaussian elimination provides a factorization of A as a product of a unit lower triangular matrix L and an upper triangular matrix U. Depending on whether the element a_{ij} lies on or above or below the principal diagonal we have

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

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

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}, \quad k = 1 : p, \quad p = \min(i-1, j). \tag{7.2.10}$$

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

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

This can also be written

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

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

$$a_{ij} = \sum_{k=1}^{r} l_{ik} u_{kj}, \quad u_{kj} = a_{kj}^{(k)}, \quad r = \min(i, j). \tag{7.2.11}$$

However, these equations are equivalent to the matrix equation $A = LU$, where $L = (l_{ik})$ and $U = (u_{kj})$ are lower and upper triangular matrices, respectively. Hence, Gaussian elimination computes a factorization of A into a product of a lower and an upper triangular matrix, the **LU factorization** of A.

It was shown in Sec. 7.2.1 that if A is nonsingular, then Gaussian elimination can always be carried through provided row interchanges are allowed. Also, such row interchanges are in general needed to ensure the numerical stability of Gaussian elimination. We now consider how the LU factorization has to be modified when such interchanges are incorporated.

Row interchanges and row permutations can be expressed as pre-multiplication with certain matrices, which we now introduce. A matrix

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

which is equal to the identity matrix except that columns i and j have been interchanged is called a **transposition matrix**. If a matrix A is premultiplied by I_{ij} this results in the interchange of *rows* i and j. Similarly, post-multiplication results in the interchange of *columns* i and j. $I_{ij}^T = I_{ij}$, and by its construction it immediately follows that $I_{ij}^2 = I$ and hence $I_{ij}^{-1} = I_{ij}$.

Assume that in the kth step, $k = 1 : n-1$, we select the pivot element from row p_k, and interchange the rows k and p_k. Notice that in these row interchanges also previously computed multipliers l_{ij} must take part. At completion of the elimination, we have obtained lower and upper triangular matrices L and U. We now make the important observation that these are the same triangular factors that are obtained if we *first* carry out the row interchanges $k \leftrightarrow p_k$, $k = 1 : n-1$, on the *original matrix* A to get a matrix PA, where P is a permutation matrix, and then perform Gaussian elimination on PA *without any interchanges.* This means that Gaussian elimination with row interchanges computes the LU factors of the matrix PA. We now summarize the results and prove the uniqueness of the LU factorization:

Theorem 7.2.2. *The LU factorization*

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

$$PA = LU, \tag{7.2.12}$$

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

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

and

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

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

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

$$PA = L_1U_1 = L_2U_2.$$

Since PA is nonsingular so are the factors, and it follows that $L_2^{-1}L_1 = U_2U_1^{-1}$. The left-hand matrix is the product of two unit lower triangular matrices and is therefore unit lower triangular, while the right hand matrix is upper triangular. It follows that both sides must be the identity matrix. Hence, $L_2 = L_1$, and $U_2 = U_1$. □

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

$$Ly = Pb, \qquad Ux = y, \tag{7.2.13}$$

which involves about $2n^2$ flops.

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

1. The factorization $PA = LU$.

2. Solution of the systems $Ly = Pb$ and $Ux = y$.

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

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

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

$$U^T c = b, \qquad L^T \tilde{x} = c. \tag{7.2.14}$$

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

In the general case when $A \in \mathbf{R}^{m\times n}$ of $\operatorname{rank}(A) = r \le \min\{m, n\}$, it can be shown that matrix $P_r A P_c \in \mathbf{R}^{m\times n}$ can be factored into a product of a unit lower **trapezoidal matrix** $L \in \mathbf{R}^{m\times r}$ and an upper trapezoidal matrix $U \in \mathbf{R}^{r\times n}$. Here P_r and P_c are permutation matrices performing the necessary row and column permutations, respectively. The factorization can be written in block form as

$$P_r A P_c = LU = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \end{pmatrix}, \tag{7.2.15}$$

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

To solve the system

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

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

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

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

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

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

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

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

The reduction of a matrix to triangular form by Gaussian elimination can be expressed entirely in matrix notations using **elementary elimination matrices**. This way of looking at Gaussian elimination, first systematically exploited by J. H. Wilkinson[16], has the advantage that it suggests ways of deriving other matrix factorizations.

Elementary elimination matrices are lower triangular matrices of the form

$$L_j = I + l_j e_j^T = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{j+1,j} & 1 & & \\ & & \vdots & & \ddots & \\ & & l_{n,j} & & & 1 \end{pmatrix}, \qquad (7.2.16)$$

where only the elements *below* the main diagonal in the jth column differ from the unit matrix. If a vector x is premultiplied by L_j we get

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

i.e., to the last $n - j$ components of x are *added* multiples of the component x_j. Since $e_j^T l_j = 0$ it follows that

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

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

so we have

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

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

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

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

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

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

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

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

$$L_j L_k = I + l_j e_j^T + l_k e_k^T, \quad j \leq k. \tag{7.2.17}$$

Note that no products of the elements l_{ij} occur! However, if $j > k$, then in general $e_j^T l_k \neq 0$, and the product $L_j L_k$ has a more complex structure.

We now show that Gaussian elimination with partial pivoting can be accomplished by premultiplication of A by a sequence of elementary elimination matrices combined with transposition matrices to express the interchange of rows. For simplicity we first consider the case when $\operatorname{rank}(A) = m = n$. In the first step assume that $a_{p_1,1} \neq 0$ is the pivot element. We then interchange rows 1 and p_1 in A by premultiplication of A by a transposition matrix,

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

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

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

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

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

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

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

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

$$U = L_{n-1}^{-1} P_{n-1} \cdots L_2^{-1} P_2 L_1^{-1} P_1 A. \tag{7.2.18}$$

To see that (7.2.18) is equivalent with the LU factorization of PA we first note that since $P_2^2 = I$ we have after the first two steps that

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

where

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

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

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

where the two elementary elimination matrices on the left hand side combine trivially. Proceeding in a similar way it can be shown that (7.2.18) implies

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

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

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

Using the result in (7.2.17), the elimination matrices can trivially be multiplied together and it follows that

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

where the elements in L are given by $l_{ij} = \tilde{l}_{ij}$, $l_{ii} = 1$, $1 \le j < i \le n$. This is the LU factorization of Theorem 7.2.2. It is important to note that nothing new, except the notations, has been introduced. In particular, the transposition matrices and elimination matrices used here are, of course, never explicitly stored in a computer implementation.

7.2.3 Pivoting Strategies

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

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

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

is nonsingular and has the unique solution $x_1 = -x_2 = -1/(1-\epsilon)$. Suppose $\epsilon = 10^{-6}$ is accepted as pivot in Gaussian elimination. Multiplying the first equation by 10^6 and subtracting from the second we obtain $(1-10^6)x_2 = -10^6$. By rounding this could give $x_2 = 1$, which is correct to six digits. However, back-substituting to obtain x_1 we get $10^{-6}x_1 = 1 - 1$, or $x_1 = 0$, which is completely wrong.

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

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

$$|a_{rk}^{(k)}| = \max_{k \le i \le n} |a_{ik}^{(k)}|. \tag{7.2.19}$$

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

$$|a_{rs}^{(k)}| = \max_{k \le i,j \le n} |a_{ij}^{(k)}|. \tag{7.2.20}$$

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

The function for Gaussian elimination can easily be modified to include partial pivoting. The interchanges are recoded in the vector p, which is initialized to be $(1, 2, \ldots, n)$.

```
function [U,c,p] = gaupp(A,b);
% GAUPP reduces a nonsingular linear system Ax = b
% by Gaussian eliminationa with parial pivoting
% to upper triangular form Ux = c. The row permutations
% are stored in the vector p
n = length(b); p = 1:n;
for k = 1:n-1
   % find element of maximum magnitude in k:th column
```

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

A major breakthrough in the understanding of Gaussian elimination came with the backward rounding error analysis of Wilkinson [386, 1961]. Using the standard model for floating-point computation Wilkinson showed that the computed triangular factors $\bar{L}$ and $\bar{U}$ of A, obtained by Gaussian elimination *are the exact triangular factors of a perturbed matrix*

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

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

$$|e_{ij}| \le 3u \min(i-1, j) \max_k |\bar{a}_{ij}^{(k)}|. \tag{7.2.21}$$

Note that the above result holds without any assumption about the size of the multipliers. *This shows that the purpose of any pivotal strategy is to avoid growth in the size of the computed elements* $\bar{a}_{ij}^{(k)}$*, and that the size of the multipliers is of no consequence* (see the remark on possible large multipliers for positive-definite matrices, Sec. 7.4.2).

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

Definition 7.2.3.
Let $a_{ij}^{(k)}$*,* $k = 2 : n$*, be the elements in the* k*th stage of Gaussian elimination applied to the matrix* $A = (a_{ij})$*. Then the* **growth ratio** *in the elimination is*

$$\rho_n = \max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}|. \tag{7.2.22}$$

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

$$|E| \le 3\rho_n u \max_{ij} |a_{ij}| F. \tag{7.2.23}$$

where F is the matrix with elements $f_{i,j} = \min\{i-1, j\}$. Strictly speaking this is not correct unless we use the growth factor $\bar{\rho}_n$ for the *computed elements.* Since this quantity differs insignificantly from the theoretical growth factor ρ_n in (7.2.22), we ignore this difference here and in the following. Slightly refining the estimate

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

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

Theorem 7.2.4.
Let $\bar{L}$ and $\bar{U}$ be the computed triangular factors of A, obtained by Gaussian elimination with floating-point arithmetic with unit roundoff u has been used, there is a matrix E such that

$$\bar{L}\bar{U} = A + E, \quad \|E\|_\infty \le 1.5n^2\rho_n u\|A\|_\infty. \tag{7.2.24}$$

If pivoting is employed so that the computed multipliers satisfy the inequality

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

Then it can be shown that an estimate similar to (7.2.24) holds with the constant 1 instead of 1.5. For both partial and complete pivoting it holds that

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

and the bound $\rho_n \le 2^{n-1}$ follows by induction. For partial pivoting this bound is the best possible, but is attained for special matrices. In practice any substantial growth of elements is extremely uncommon with partial pivoting. Quoting Kahan [230, 1966]:

> *Intolerable pivot-growth (with partial pivoting) is*
> *a phenomenon that happens only to numerical analysts*
> *who are looking for that phenomenon.*

Why large element growth rarely occurs in practice with partial pivoting is a subtle and still not fully understood phenomenon. Trefethen and Schreiber [361, 1990] show that for certain distributions of random matrices the average element growth with partial pivoting is close to $n^{2/3}$

For complete pivoting a much better bound can be proved, and in practice the growth very seldom exceeds n; see Sec. 7.6.4. A pivoting scheme that gives a pivot of size between that of partial and complete pivoting is **rook pivoting**. In this scheme we pick a pivot element which is largest in magnitude in *both its column and its row.*

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

$$|a_{rs}^{(k)}| = \max_{k \le i \le n} |a_{ij}^{(k)}| = \max_{k \le j \le n} |a_{ij}^{(k)}|. \tag{7.2.25}$$

We start by finding the element of maximum magnitude in the first column. If this element is also of maximum magnitude in its row we accept it as pivot. Otherwise we compare the element of maximum magnitude in the row with other elements in its column, etc. The name derives from the fact that the pivot search resembles the moves of a rook in chess; see Figure 7.2.1..

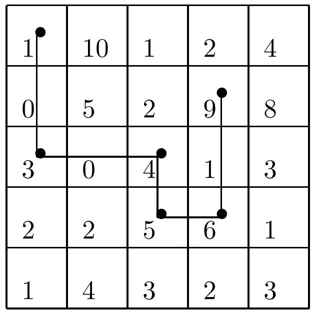

Figure 7.2.1. *Illustration of rook pivoting in a* 5×5 *matrix with positive integer entries as shown. The* $(2,4)$ *element 9 is chosen as pivot.*

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

It is important to realize that the choice of pivots is influenced by the scaling of equations and unknowns. If, for example, the unknowns are physical quantities a different choice of units will correspond to a different scaling of the unknowns and the columns in A. Partial pivoting has the important property of being invariant under column scalings. In theory we could perform partial pivoting by *column* interchanges, which then would be invariant under row scalings. but in practice this turns out to be less satisfactory. Likewise, an unsuitable column scaling can also make complete pivoting behave badly.

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

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

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

Definition 7.2.5. *A matrix A is said to be* **diagonally dominant** *by rows, if*

$$\sum_{j\neq i} |a_{ij}| \leq |a_{ii}|, \quad i = 1 : n. \tag{7.2.26}$$

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

Theorem 7.2.6.

Let A be nonsingular and diagonally dominant by rows or columns. Then A has an LU factorization without pivoting and the growth ratio $\rho_n(A) \leq 2$. If A is diagonally dominant by columns, then the multipliers in this LU factorization satisfy $|l_{ij}| \leq 1$, for $1 \leq j < i \leq n$.

Proof. (Wilkinson [386, pp. 288–289])

Assume that A is nonsingular and diagonally dominant by columns. Then $a_{11} \neq 0$, since otherwise the first column would be zero and A singular. In the first stage of Gaussian elimination without pivoting we have

$$a_{ij}^{(2)} = a_{ij} - l_{i1}a_{1j}, \quad l_{i1} = a_{i1}/a_{11}, \quad i, j \geq 2, \tag{7.2.27}$$

where

$$\sum_{i=2}^{n} |l_{i1}| \leq \sum_{i=2}^{n} |a_{i1}|/|a_{11}| \leq 1. \tag{7.2.28}$$

For $j = i$, using the definition and (7.2.28), it follows that

$$\begin{aligned} |a_{ii}^{(2)}| \geq |a_{ii}| - |l_{i1}|\,|a_{1i}| &\geq \sum_{j\neq i} |a_{ji}| - \Big(1 - \sum_{j\neq 1,i} |l_{j1}|\Big)|a_{1i}| \\ &= \sum_{j\neq 1,i} \big(|a_{ji}| + |l_{j1}||a_{1i}|\big) \geq \sum_{j\neq 1,i} |a_{ji}^{(2)}|. \end{aligned}$$

Hence, the reduced matrix $A^{(2)} = (a_{ij}^{(2)})$, is also nonsingular and diagonally dominant by columns. It follows by induction that all matrices $A^{(k)} = (a_{ij}^{(k)})$, $k = 2 : n$ are nonsingular and diagonally dominant by columns.

Further using (7.2.27) and (7.2.28), for $i \geq 2$,

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

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

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

It follows that

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

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

We conclude that for a row or column diagonally dominant matrix Gaussian elimination without pivoting is backward stable. If A is diagonally dominant by rows then the multipliers can be arbitrarily large, but this does not affect the stability.

If (7.2.26) holds with strict inequality for all i, then A is said to be **strictly diagonally dominant** by rows. If A is strictly diagonally dominant, then it can be shown that all reduced matrices have the same property. In particular, all pivot elements must then be strictly positive and the nonsingularity of A follows. We mention a useful result for strictly diagonally dominant matrices.

Lemma 7.2.7.

Let A be strictly diagonally dominant by rows, and set

$$\alpha = \min_i \alpha_i, \quad \alpha_i := |a_{ii}| - \sum_{j \ne i} |a_{ij}| > 0, \quad i = 1 : n. \tag{7.2.29}$$

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

Proof. By the definition of a subordinate norm (7.1.64) we have

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

Assume that equality holds in (7.2.29) for $i = k$. Then

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

◻

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

7.2.4 Computational Variants

In Gaussian Elimination, as described in Sec. 7.2.2, the kth step consists of modifying the unreduced part of the matrix by an *outer product* of the vector of multipliers and the pivot row. Using the equivalence of Gaussian elimination and LU factorization it is easy to see that the the calculations can be arranged in several different ways so that the elements in L and U are determined directly.

For simplicity, we first assume that any row or column interchanges on A have been carried out in advance. The matrix equation $A = LU$ can be written in component-wise form (see (7.2.11))

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

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

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

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj}, \quad j \ge k, \qquad l_{ik} u_{kk} = a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk}, \quad i > k. \quad (7.2.30)$$

This algorithm is usually referred to as Doolittle's algorithm [113, 1878]. The more well-known Crout's algorithm [80, 1941] is similar except that instead the upper triangular matrix U is normalized to have a unit diagonal.[17]

The main work in Doolittle's algorithm is performed in the inner products of rows of L and columns in U. Since the LU factorization is unique this algorithm produces the same factors L and U as Gaussian elimination. In fact, successive partial sums in the equations (7.2.30) equal the elements $a_{ij}^{(k)}$, $j > k$, in Gaussian elimination. It follows that if each term in (7.2.30) is rounded separately, the compact algorithm is also *numerically equivalent* to Gaussian elimination. If the inner products can be accumulated in higher precision, then the compact algorithm is less affected by rounding errors.

Doolittle's algorithm can be modified to include partial pivoting. Changing the order of operations, we first calculate the elements $\tilde{l}_{ik} = l_{ik} u_{kk}$, $i = k : n$, and determine that of maximum magnitude. The corresponding row is then permuted to pivotal position. In this row exchange the already computed part of L and remaining part of A also take part. Next we normalize by setting $l_{kk} = 1$, which determines l_{ik}, $i = 1 : k$, and also u_{kk}. Finally, the remaining part of the kth row in U is computed.

Using the index conventions inMATLAB Doolittle's algorithm with partial pivoting becomes:

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

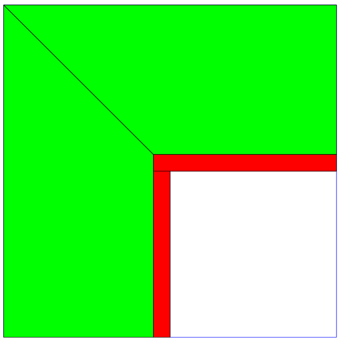

Figure 7.2.2. *Computations in the kth step of Doolittle's method.*

Algorithm 7.3. *Doolittle's Algorithm.*

```
function [L,U,p] = dool(A);
% DOOL computes the LU factorization of a nonsingular
% matrix PA using partial pivoting. The row permutations
% are stored in the vector p
n = size(A,1); p = 1:n;
L = zeros(n,n); U = L;
for k = 1:n
   for i = k:n
      L(i,k) = A(i,k) - L(i,1:k-1)*U(1:k-1,k);
   end
   [piv,q] = max(abs(L(k:n,k)));
   q = k-1+q;
   if q > k   % switch rows k and q in L and A
      L([k q],1:k) = L([q k],1:k);
      A([k q],k+1:n) = A([q k],k+1:n);
      p([k q]) = p([q k]);
   end
   U(k,k) = L(k,k); L(k,k) = 1;
   for j = k+1:n
      L(j,k) = L(j,k)/U(k,k);
      U(k,j) = A(k,j) - L(k,1:k-1)*U(1:k-1,j);
   end
end
```

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

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

Here the entries indicate in which step a certain element l_{ij} and u_{ij} is computed. The first example corresponds to the ordering in the algorithm given above. (Compare the comments after Algorithm 7.2.1.) Note that it is *not* easy to do complete pivoting with either of the last two variants.

The Bordering Method

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

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

Identifying the (1,2)-blocks we find

$$L_{11}u_{1k} = a_{1k}, \tag{7.2.31}$$

which is a lower triangular system for u_{1k}. Identifying the (2,1)-blocks and transposing gives

$$U_{11}^T l_{k1} = a_{k1}, \tag{7.2.32}$$

another lower triangular system for l_{k1}. Finally, from the (2,2)-block we get $l_{k1}^T u_{1k} + u_{kk} = \alpha_{kk}$, or

$$u_{kk} = \alpha_{kk} - l_{k1}^T u_{1k}. \tag{7.2.33}$$

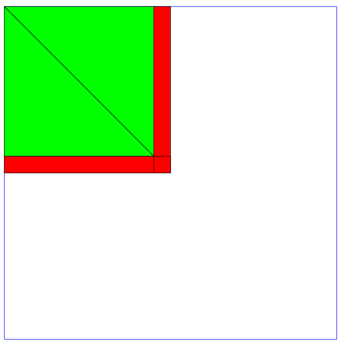

Figure 7.2.3. *Computations in the kth step of the bordering method.*

The main work in this variant is done in solving the triangular systems (7.2.31) and (7.2.32). A drawback of the bordering method is that it cannot be combined with partial pivoting.

The Sweep Methods

In the **column sweep** method at the kth step the first k columns of L and U in LU-factorization of A are computed. Assume that we have computed L_{11}, L_{21}, and U_{11} in the factorization

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

As in the bordering method, identifying the (1,2)-blocks we find

$$L_{11}u_{1k} = a_{1k}, \tag{7.2.34}$$

a lower triangular system for u_{1k}. From the (2,2)-blocks we get $L_{21}u_{1k} + l_{2k}u_{kk} = a_{2k}$, or

$$l_{2k}u_{kk} = a_{2k} - L_{21}u_{1k}. \tag{7.2.35}$$

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

Partial pivoting can be implemented with this method as follows. When the right hand side in (7.2.35) has been evaluated we determine the element of maximum modulus in the vector $a_{2k} - L_{21}u_{1k}$. We then permute this element to top position and perform the same row exchanges in L_{21}^T and the unprocessed part of A.

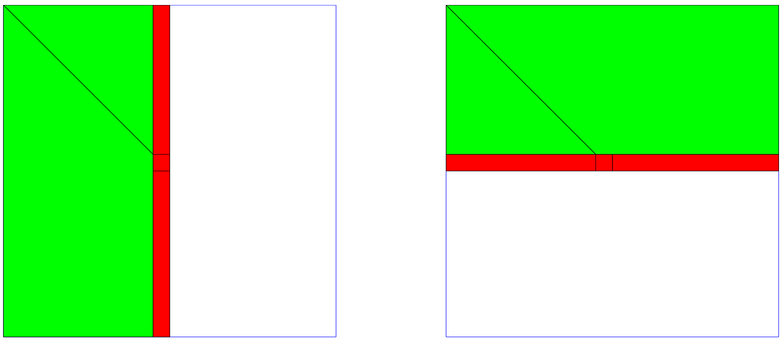

Figure 7.2.4. *Computations in the kth step of the sweep methods. Left: The column sweep method. Right: The row sweep method.*

In the column sweep method L and U are determined column by column. It is possible to determine L and U row by row. In the kth step of this **row sweep** method the kth row of A is processed and we write

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

Identifying the (2,1)- and (2,2)-blocks we get

$$U_{11}^T l_{k1} = a_{k1}, \tag{7.2.36}$$

and

$$u_{2k}^T = a_{k2}^T - l_{k1}U_{12}. \tag{7.2.37}$$

Note that Doolittle's method can be viewed as alternating between the two sweep methods.

Consider now the case when $A \in \mathbf{R}^{m\times n}$ is a rectangular matrix with rank $(A) = r = \min(m,n)$. If $m > n$ it is advantageous to process the matrix column by column. Then after n steps we have $AP_c = LU$, where L is lower trapezoidal,

$$L = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \in \mathbf{R}^{m\times n}, \tag{7.2.38}$$

and $U \in \mathbf{R}^{n\times n}$ is square upper triangular. If $m < n$ and the matrix is processed row by row, we have after n steps an LU factorization with $L \in \mathbf{R}^{m\times m}$ and

$$U = \begin{pmatrix} U_{11} & U_{12} \end{pmatrix} \in \mathbf{R}^{m\times n}$$

upper trapezoidal.

7.2.5 Computing the Inverse

If the inverse matrix A^{-1} is known, then the solution of $Ax = b$ can be obtained through a matrix vector multiplication by $x = A^{-1}b$. This is theoretically satisfying, but in most practical computational problems it is unnecessary and inadvisable to compute A^{-1}. As succinctly expressed by Forsythe and Moler [139, 1967]:

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

The work required to compute A^{-1} is about $2n^3$ flops, i.e., three times greater than for computing the LU factorization. (If A is a band matrix, then the savings can be much more spectacular; see Sec. 7.4.) To solve a linear system $Ax = b$ the matrix vector multiplication $A^{-1}b$ requires $2n^2$ flops. This is exactly the same as for the solution of the two triangular systems $L(Ux) = b$ resulting from LU factorization of A. (Note, however, that on some parallel computers matrix multiplication can be performed much faster than solving triangular systems.)

One advantage of computing the inverse matrix is that A^{-1} can be used to get a strictly reliable error estimate for a computed solution $\bar{x}$. A similar estimate is not directly available from the LU factorization. However, alternative ways to obtain error estimates are the use of a condition estimator (Sec. 7.6.2) or iterative refinement (Sec. 7.7.3).

Not only is the inversion approach three times more expensive but if A is ill-conditioned the solution computed from $A^{-1}b$ usually is much less accurate than than that computed from the LU factorization. Using LU factorization the relative precision of the residual vector of the computed solution will usually be of the order of machine precision even when A is ill-conditioned.

Nevertheless, there are some applications where A^{-1} is required, e.g., in some methods for computing the matrix square root and the logarithm of a matrix; see Sec. 9.2.4. The inverse of a symmetric positive definite matrix is needed to obtain estimates of the covariances in regression analysis. However, usually only certain elements of A^{-1} are needed and not the whole inverse matrix.

We first consider computing the inverse of a lower triangular matrix L. Setting $L^{-1} = Y = (y_1, \dots, y_n)$, we have $LY = I = (e_1, \dots, e_n)$. This shows that the columns of Y satisfy

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

These lower triangular systems can be solved by forward substitution. Since the vector e_j has $(j-1)$ leading zeros the first $(j-1)$ components in y_j are zero. Hence, L^{-1} is also a lower triangular matrix, and its elements can be computed recursively from

$$y_{jj} = 1/l_{jj}, \qquad y_{ij} = \Big(-\sum_{k=j}^{i-1} l_{ik} y_{kj} \Big) \Big/ l_{ii}, \quad i = j+1 : n, \tag{7.2.39}$$

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

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

$$z_{jj} = 1/u_{jj}, \qquad z_{ij} = \Big(-\sum_{k=i+1}^{j} u_{ik} z_{kj} \Big) \Big/ u_{ii}, \quad i = j-1 : -1 : 1. \tag{7.2.40}$$

If the columns are computed in the order $j = n : -1 : 1$, the Z can overwrite U in storage. The number of flops required to compute L^{-1} or U^{-1} is approximately equal to $n^3/3$. Variants of the above algorithm can be obtained by reordering the loop indices.

Now let $A^{-1} = X = (x_1, \dots, x_n)$ and assume that an LU factorization $A = LU$ has been computed. Then

$$Ax_j = L(Ux_j) = e_j, \quad j = 1 : n, \tag{7.2.41}$$

and the columns of A^{-1} are obtained by solving n linear systems, where the right hand sides equal the columns in the unit matrix. Setting (7.2.41) is equivalent to

$$Ux_j = y_j, \qquad Ly_j = e_j, \quad j = 1 : n. \tag{7.2.42}$$

This method for inverting A requires $2n^3/3$ flops for inverting L and n^3 flops for solving the n upper triangular systems giving again a total of $2n^3$ flops.

A second method uses the relation

$$A^{-1} = (LU)^{-1} = U^{-1} L^{-1}. \tag{7.2.43}$$

Since the matrix multiplication $U^{-1}L^{-1}$ requires $n^3/3$ flops (show this!) the total work to compute A^{-1} by the second method method (7.2.43) is also n^3 flops. If we take advantage of that $y_{jj} = 1/l_{jj} = 1$, and carefully sequence the computations then L^{-1}, U^{-1} and finally A^{-1} can overwrite A so that no extra storage is needed.

There are many other variants of computing the inverse $X = A^{-1}$. From $XA = I$ we have

$$XLU = I \quad \text{or} \quad XL = U^{-1}.$$

In the MATLABfunction `inv(A)`, U^{-1} is first computed by a column oriented algorithm. Then the system $XL = U^{-1}$ is solved for X. The stability properties of this and several other different matrix inversion algorithms are analyzed in [114, 1992]; see also Higham [211, Sec. 14.2].

In Gaussian elimination we use in the kth step the pivot row to eliminate elements *below* the main diagonal in column k. In **Gauss–Jordan elimination**[18] the elements *above* the main diagonal are eliminated simultaneously. After $n-1$ steps the matrix A has then been transformed into a *diagonal* matrix containing the nonzero pivot elements. Gauss–Jordan elimination was used in many early versions of linear programming and also for implementing stepwise regression in statistics.

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

$$M_j = \begin{pmatrix} 1 & & & l_{1j} & & & \\ & \ddots & & \vdots & & & \\ & & 1 & l_{j-1,j} & & & \\ & & & 1 & & & \\ & & & l_{j+1,j} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & l_{n,j} & & & 1 \end{pmatrix}. \tag{7.2.44}$$

If partial pivoting is carried out we can write, cf. (7.2.18)

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

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

$$A^{-1} = D^{-1} M_n M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1. \tag{7.2.45}$$

This expresses the inverse of A as a product of elimination and transposition matrices, and is called the **product form of the inverse**. The operation count for this elimination process is $\approx n^3$ flops, i.e., higher than for the LU factorization by Gaussian elimination. For some parallel implementations Gauss–Jordan elimination may still have advantages.

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

$$x = A^{-1} b = D^{-1} M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1 b. \tag{7.2.46}$$

This requires $2n^2$ flops. Note that no back-substitution is needed!

The inverse can also be obtained from the Gauss–Jordan factorization. Using (7.2.46) where b is taken to be the columns of the unit matrix, we compute

$$A^{-1} = D^{-1} M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1 (e_1, \ldots, e_n).$$

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

Again $2n^3$ flops are required if the computations are properly organized. The method can be arranged so that the inverse emerges in the original array. However, the numerical properties of this method are not as good as for the methods described above.

If row interchanges have been performed during the LU factorization, we have $PA = LU$, where $P = P_{n-1}\cdots P_2P_1$ and P_k are transposition matrices. Then $A^{-1} = (LU)^{-1}P$. Hence, we obtain A^{-1} by performing the interchanges in *reverse order on the columns* of $(LU)^{-1}$.

The stability of Gauss–Jordan elimination has been analyzed by Peters and Wilkinson [310, 1975]. They remark that the residuals $b - A\bar{x}$ corresponding to the Gauss–Jordan solution $\bar{x}$ can be a larger by a factor $\kappa(A)$ than those corresponding to the solution by Gaussian elimination. Although the method is not backward stable in general it can be shown to be stable for so-called diagonally dominant matrices (see Definition 7.2.5). It is also forward stable, i.e., will give about the same numerical accuracy in the computed solution $\bar{x}$ as Gaussian elimination.

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

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

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

$$\|B^{k+1}\| \le \|B\|^{k+1} \to 0, \quad k \to \infty,$$

and hence the **Neumann expansion**

$$(I - B)^{-1} = I + B + B^2 + B^3 + \cdots, \tag{7.2.47}$$

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

$$(I - B)^{-1} = (I + B)(I + B^2)(I + B^4)\cdots. \tag{7.2.48}$$

It can be shown by induction that

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

Finally, we mention an iterative method for computing the inverse, the **Newton–Schultz iteration**

$$X_{k+1} = X_k(2I - AX_k) = (2I - AX_k)X_k. \tag{7.2.49}$$

This is an analogue to the iteration $x_{k+1} = x_k(2 - ax_k)$, for computing the inverse of a scalar. It can be shown that if $X_0 = \alpha_0 A^T$ and $0 < \alpha_0 < 2/\|A\|_2^2$, then $\lim_{k\to\infty} X_k = A^{-1}$. Convergence can be slow initially but ultimately quadratic,

$$E_{k+1} = E_k^2, \quad E_k = I - AX_k \ \text{ or } \ I - X_kA.$$

Since about $2\log_2 \kappa_2(A)$ (see [338, 1974]) iterations are needed for convergence it cannot in general compete with direct methods for dense matrices. However, a few steps of the iteration (7.2.49) can be used to improve an approximate inverse.

Review Questions

2.1 How many operations are needed (approximately) for

(a) The LU factorization of a square matrix?

(b) The solution of $Ax = b$, when the triangular factorization of A is known?

2.2 Show that if the kth diagonal entry of an upper triangular matrix is zero, then its first k columns are linearly dependent.

2.3 What is meant by partial and complete pivoting in Gaussian elimination? Mention two classes of matrices for which Gaussian elimination can be performed stably without any pivoting?

2.4 What is the LU-decomposition of an n by n matrix A, and how is it related to Gaussian elimination? Does it always exist? If not, give sufficient conditions for its existence.

2.5 How is the LU-decomposition used for solving a linear system? What are the advantages over using the inverse of A? Give an approximate operation count for the solution of a dense linear system with p different right hand sides using the LU-decomposition.

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

Problems

2.1 (a) Compute the LU factorization of A and $\det(A)$, where

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \end{pmatrix}.$$

(b) Solve the linear system $Ax = b$, where $b = (2, 10, 44, 190)^T$.

2.2 (a) Show that $P = (e_n, \dots, e_2, e_1)$ is a permutation matrix and that $P = P^T = P^{-1}$, and that Px reverses the order of the elements in the vector x.

(b) Let the matrix A have an LU factorization. Show that there is a related factorization $PAP = UL$, where U is upper triangular and L lower triangular.

2.3 In Algorithm 7.2.1 for Gaussian elimination the elements in A are assessed in row-wise order in the innermost loop over j. If implemented in Fortran this algorithm may be inefficient since this language stores two-dimensional arrays by columns. Modify Algorithm 7.2.1 so that the innermost loop instead involves a fixed column index and a varying row index.

2.4 What does M_j^{-1}, where M_j is defined in (7.2.44), look like?

2.5 Compute the inverse matrix A^{-1}, where

$$A = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 4 & 1 & 2 \end{pmatrix},$$

(a) By solving $AX = I$, using Gaussian elimination with partial pivoting.

(b) By LU factorization and using $A^{-1} = U^{-1}L^{-1}$.

7.3 Symmetric Matrices

7.3.1 Symmetric Positive Definite Matrices

Gaussian elimination can be adopted to several classes of matrices of special structure. As mentioned in Sec. 7.2.3, one case when Gaussian elimination can be performed stably without any pivoting is when A is Hermitian or real symmetric and positive definite. Solving such systems is one of the most important problems in scientific computing.

Definition 7.3.1.

A matrix $A \in \mathbf{C}^{n\times n}$ is called **Hermitian** *if $A = A^H$, the conjugate transpose of A. If A is Hermitian, then the quadratic form $(x^HAx)^H = x^HAx$ is real and A is said to be* **positive definite** *if*

$$x^HAx > 0 \quad \forall x \in \mathbf{C}^n, \quad x \neq 0, \tag{7.3.1}$$

and **positive semi-definite** *if $x^TAx \geq 0$, for all $x \in \mathbf{R}^n$. Otherwise it is called* **indefinite**.

It is well known that all eigenvalues of an Hermitian matrix are real. An equivalent condition for an Hermitian matrix to be positive definite is that all its eigenvalues are positive

$$\lambda_k(A) > 0, \quad k = 1 : n.$$

Since this condition can be difficult to verify the following sufficient condition is useful. A Hermitian matrix A, which has positive diagonal elements and is diagonally dominant

$$a_{ii} > \sum_{j\neq i} |a_{ij}|, \quad i = 1 : n,$$

can be shown to be positive definite, since it follows from Gerschgorin's Theorem (Theorem 9.2.1 that the eigenvalues of A are all positive.

Clearly a positive definite matrix is nonsingular, since if it were singular there should be a null vector $x \neq 0$ such that $Ax = 0$ and then $x^HAx = 0$. Positive definite (semi-definite) matrices have the following important property:

Theorem 7.3.2. *Let $A \in \mathbf{C}^{n\times n}$ be positive definite (semi-definite) and let $X \in \mathbf{C}^{n\times p}$ have full column rank. Then X^HAX is positive definite (semi-definite). In* *particular, any principal* $p \times p$ *submatrix*

$$\tilde{A} = \begin{pmatrix} a_{i_1 i_1} & \dots & a_{i_1 i_p} \\ \vdots & & \vdots \\ a_{i_p i_1} & \dots & a_{i_p i_p} \end{pmatrix} \in \mathbf{C}^{p\times p}, \quad 1 \le p < n,$$

is positive definite (semi-definite). Taking $p = 1$*, it follows that all diagonal elements in* A *are real positive (nonnegative).*

Proof. Let $z \neq 0$ and let $y = Xz$. Then since X is of full column rank $y \neq 0$ and $z^H(X^HAX)z = y^HAy > 0$ by the positive definiteness of A. Now, any principal submatrix of A can be written as X^HAX, where the columns of X are taken to be the columns $k = i_j$, $j = 1,\dots,p$ of the identity matrix. The case when A is positive semi-definite follows similarly. $\square$

A Hermitian or symmetric matrix A of order n has only $\frac{1}{2}n(n+1)$ independent elements. If A is also positive definite then symmetry can be preserved in Gaussian elimination and the number of operations and storage needed can be reduced by half. Indeed, Gauss derived his original algorithm for the symmetric positive definite systems coming from least squares problems (see Chapter 8). We consider below the special case when A is real and symmetric but all results are easily generalized to the complex Hermitian case. (Complex symmetric matrices also appear in some practical problems. These are more difficult to handle.)

Lemma 7.3.3. *Let* A *be a real symmetric matrix. If Gaussian elimination can be carried through without pivoting, then the reduced matrices*

$$A = A^{(1)}, A^{(2)}, \dots, A^{(n)}$$

are all symmetric.

Proof. Assume that $A^{(k)}$ is symmetric, for some k, where $1 \le k < n$. Then by Algorithm 7.2.1 we have after the k-th elimination step

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

$k+1 \le i,j \le n$. This shows that $A^{(k+1)}$ *is also a symmetric matrix*, and the result follows by induction. $\square$

A more general result is the following. Partition the Hermitian positive definite matrix A as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^H & A_{22} \end{pmatrix}$$

where A_{11} is a square matrix, Then by Theorem 7.3.2 both A_{11} and A_{22} are Hermitian positive definite and therefore nonsingular. It follows that the Schur complement of A_{11} in A, which is

$$S = A_{22} - A_{12}^H A_{11}^{-1} A_{12}$$

exists and is Hermitian. Moreover, for $x \neq 0$, we have

$$x^H(A_{22} - A_{12}^H A_{11}^{-1} A_{12})x = \begin{pmatrix} y^H & -x^H \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^H & A_{22} \end{pmatrix} \begin{pmatrix} y \\ -x \end{pmatrix} > 0$$

where $y = A_{11}^{-1} A_{12} x$, it follows that S is positive definite.

From Lemma 7.3.3 it follows that in Gaussian elimination without pivoting only the elements in $A^{(k)}$, $k = 2 : n$, on and below the main diagonal have to be computed. Since any diagonal element can be brought in pivotal position by a symmetric row and column interchange, the same conclusion holds if pivots are chosen arbitrarily along the diagonal.

Assume that the lower triangular part of the symmetric matrix A is given. The following algorithm computes, *if it can be carried through*, a *unit lower triangular* matrix $L = (l_{ik})$, and a diagonal matrix $D = \mathrm{diag}(d_1, \dots, d_n)$ such that

$$A = LDL^T. \tag{7.3.2}$$

Algorithm 7.4. *Symmetric Gaussian Elimination.*

for $k = 1 : n - 1$
  $d_k := a_{kk}^{(k)}$;
  for $i = k + 1 : n$
    $l_{ik} := a_{ik}^{(k)}/d_k$;
    for $j = k + 1 : i$
      $a_{ij}^{(k+1)} := a_{ij}^{(k)} - l_{ik} d_k l_{jk}$;
    end
  end
end

In inner loop we have substituted $d_k l_{jk}$ for $a_{jk}^{(k)}$.

Note that the elements in L and D can overwrite the elements in the lower triangular part of A, so also the storage requirement is halved to $n(n + 1)/2$. The uniqueness of the LDL^T factorization follows trivially from the uniqueness of the LU factorization.

Using the factorization $A = LDL^T$ the linear system $Ax = b$ decomposes into the two triangular systems

$$Ly = b, \qquad L^T x = D^{-1} y. \tag{7.3.3}$$

The cost of solving these triangular systems is about $2n^2$ flops.

Example 7.3.1.

It may not always be possible to perform Gaussian elimination on a symmetric matrix, using pivots chosen from the diagonal. Consider, for example, the nonsingular symmetric matrix

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

If we take $\epsilon = 0$, then both diagonal elements are zero, and symmetric Gaussian elimination breaks down. If $\epsilon \neq 0$, but $|\epsilon| \ll 1$, then choosing ϵ as pivot will not be stable. On the other hand, a row interchange will in general destroy symmetry!

We will prove that Gaussian elimination without pivoting can be carried out with positive pivot elements if and only if A is real and symmetric positive definite. (The same result applies to complex Hermitian matrices, but since the modifications necessary for this case are straightforward, we discuss here only the real case.) For symmetric semi-definite matrices symmetric pivoting can be used. The *indefinite* case requires more substantial modifications, which will be discussed in Sec. 7.3.4.

Theorem 7.3.4.

The symmetric matrix $A \in \mathbf{R}^{n\times n}$ is positive definite if and only if there exists a unit lower triangular matrix L and a diagonal matrix D with positive elements such that

$$A = LDL^T, \quad D = \operatorname{diag}(d_1, \dots, d_n),$$

Proof. Assume first that we are given a symmetric matrix A, for which Algorithm 7.3.1 yields a factorization $A = LDL^T$ with positive pivotal elements $d_k > 0$, $k = 1 : n$. Then for all $x \neq 0$ we have $y = L^T x \neq 0$ and

$$x^T Ax = x^T LDL^T x = y^T Dy > 0.$$

It follows that A is positive definite.

The proof of the other part of the theorem is by induction on the order n of A. The result is trivial if $n = 1$, since then $D = d_1 = A = a_{11} > 0$ and $L = 1$. Now write

$$A = \begin{pmatrix} a_{11} & a^T \\ a & \tilde{A} \end{pmatrix} = L_1 D_1 L_1^T, \quad L_1 = \begin{pmatrix} 1 & 0 \\ d_1^{-1} a & I \end{pmatrix}, \quad D_1 = \begin{pmatrix} d_1 & 0 \\ 0 & B \end{pmatrix},$$

where $d_1 = a_{11}$, $B = \tilde{A} - d_1^{-1} aa^T$. Since A is positive definite it follows that D_1 is positive definite, and therefore $d_1 > 0$, and B is positive definite. Since B is of order $(n-1)$, by the induction hypothesis there exists a unique unit lower triangular matrix $\tilde{L}$ and diagonal matrix $\tilde{D}$ with positive elements such that $B = \tilde{L}\tilde{D}\tilde{L}^T$. Then it holds that $A = LDL^T$, where

$$L = \begin{pmatrix} 1 & 0 \\ d_1^{-1} a & \tilde{L} \end{pmatrix}, \qquad D = \begin{pmatrix} d_1 & 0 \\ 0 & \tilde{D} \end{pmatrix}.$$

□

Example 7.3.2. The Hilbert matrix $H_n \in \mathbf{R}^{n\times n}$ with elements

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

is positive definite. Hence, , if Gaussian elimination without pivoting is carried out then the pivotal elements are all positive. For example, for $n = 4$, symmetric Gaussian elimination yields the $H_4 = LDL^T$, where

$$D = \mathrm{diag}\,(1, 1/12, 1/180, 1/2800)\,, \qquad L = \begin{pmatrix} 1 & & & \\ 1/2 & 1 & & \\ 1/3 & 1 & 1 & \\ 1/4 & 9/10 & 3/2 & 1 \end{pmatrix}.$$

Theorem 7.3.4 also yields the following useful characterization of a positive definite matrix.

Theorem 7.3.5. Sylvester's Criterion
A symmetric matrix $A \in \mathbf{R}^{n\times n}$ is positive definite if and only if

$$\det(A_k) > 0, \quad k = 1, 2, \ldots, n,$$

where $A_k \in \mathbf{R}^{k\times k}$, $k = 1, 2 : n$, are the leading principal submatrices of A.

Proof. If symmetric Gaussian elimination is carried out without pivoting then

$$\det(A_k) = d_1 d_2 \cdots d_k.$$

Hence, $\det(A_k) > 0$, $k = 1 : n$, if and only if all pivots are positive. However, by Theorem 7.3.2 this is the case if and only if A is positive definite. □

In order to prove a bound on the growth ratio for the symmetric positive definite we first show the following

Lemma 7.3.6. *For a symmetric positive definite matrix $A = (a_{ij}) \in \mathbf{R}^{n\times n}$ the maximum element of A lies on the diagonal.*

Proof. Theorem 7.3.2 and Sylvester's criterion imply that

$$0 < \det \begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix} = a_{ii}a_{jj} - a_{ij}^2, \quad 1 \le i, j \le n.$$

Hence,

$$|a_{ij}|^2 < a_{ii}a_{jj} \le \max_{1\le i\le n} a_{ii}^2,$$

from which the lemma follows. □

Theorem 7.3.7.

Let A be symmetric and positive definite. Then Gaussian elimination without pivoting is backward stable and the growth ratio satisfies $\rho_n \le 1$.

Proof. In Algorithm 7.3.1 the diagonal elements are transformed in the k:th step of Gaussian elimination according to

$$a_{ii}^{(k+1)} = a_{ii}^{(k)} - (a_{ki}^{(k)})^2/a_{kk}^{(k)} = a_{ii}^{(k)}\Big(1 - (a_{ki}^{(k)})^2/\big(a_{ii}^{(k)}a_{kk}^{(k)}\big)\Big).$$

If A is positive definite so are $A^{(k)}$ and $A^{(k+1)}$. Using Lemma 7.3.6 it follows that $0 < a_{ii}^{(k+1)} \le a_{ii}^{(k)}$, and hence the diagonal elements in the successive reduced matrices cannot increase. Thus, we have

$$\max_{i,j,k} |a_{ij}^{(k)}| = \max_{i,k} a_{ii}^{(k)} \le \max_i a_{ii} = \max_{i,j} |a_{ij}|,$$

which implies that $\rho_n \le 1$. $\square$

Any matrix $A \in \mathbf{R}^{n\times n}$ can be written as the sum of a symmetric and a skew-symmetric part, $A = H + S$, where

$$A_H = \tfrac{1}{2}(A + A^T), \qquad A_S = \tfrac{1}{2}(A - A^T). \tag{7.3.4}$$

A is symmetric if and only if $A_S = 0$. Sometimes A is called positive definite if its symmetric part A_H is positive definite. If the matrix A has a positive symmetric part then its leading principal submatrices are nonsingular and Gaussian elimination can be carried out to completion without pivoting. However, the resulting LU factorizing may not be stable as shown by the example

$$\begin{pmatrix} \epsilon & 1 \\ -1 & \epsilon \end{pmatrix} = \begin{pmatrix} 1 & \\ -1/\epsilon & 1 \end{pmatrix} \begin{pmatrix} \epsilon & 1 \\ & \epsilon + 1/\epsilon \end{pmatrix}, \quad (\epsilon > 0).$$

These results can be extended to complex matrices with positive definite Hermitian part $A_H = \frac{1}{2}(A + A^H)$, for which its holds that $x^H A x > 0$, for all nonzero $x \in \mathbf{C}^n$. Of particular interest are complex symmetric matrices, arising in computational electrodynamics, of the form

$$A = B + iC, \quad B, C \in \mathbf{R}^{n\times n}, \tag{7.3.5}$$

where $B = A_H$ and $C = A_S$ both are symmetric positive definite. It can be shown that for this class of matrices $\rho_n < 3$, so LU factorization without pivoting is stable (see [152, 1982]).

7.3.2 Cholesky Factorization

Let A be a symmetric positive definite matrix A. Then the $\mathrm{LDL^T}$ factorization (7.3.2) exists and $D > 0$. Hence, we can write

$$A = LDL^T = (LD^{1/2})(LD^{1/2})^T, \quad D^{1/2} = \mathrm{diag}\,(\sqrt{d_1}, \dots, \sqrt{d_n}). \tag{7.3.6}$$

Defining the upper triangular matrix $R = D^{1/2}L^T$ we obtain the factorization

$$A = R^T R. \tag{7.3.7}$$

If we here take the diagonal elements of L to be positive it follows from the uniqueness of the $\mathrm{LDL^T}$ factorization that this factorization is unique. The factorization (7.3.7) is called the **Cholesky factorization** of A, and R is called the **Cholesky factor** of A.[19]

The Cholesky factorization is obtained if in symmetric Gaussian elimination (Algorithm 7.3.1) we set $d_k = l_{kk} = (a_{kk}^{(k)})^{1/2}$. This gives the outer product version of Cholesky factorization in which in the kth step, the reduced matrix is modified by a rank-one matrix

$$A^{(k+1)} = A^{(k)} - l_k l_k^T,$$

where l_k denotes the column vector of multipliers.

In analogy to the compact schemes for LU factorization (see Sec. 7.2.5) it is possible to arrange the computations so that the elements in the Cholesky factor $R = (r_{ij})$ are determined directly. The matrix equation $A = R^T R$ with R upper triangular can be written

$$a_{ij} = \sum_{k=1}^{i} r_{ki} r_{kj} = \sum_{k=1}^{i-1} r_{ki} r_{kj} + r_{ii} r_{ij}, \quad 1 \le i \le j \le n. \tag{7.3.8}$$

This is $n(n+1)/2$ equations for the unknown elements in R. We remark that for $i = j$ this gives

$$\max_i r_{ij}^2 \le \sum_{k=1}^{j} r_{kj}^2 = a_j \le \max_i a_{ii},$$

which shows that the elements in R are bounded by the maximum diagonal element in A. Solving for r_{ij} from the corresponding equation in (7.3.8), we obtain

$$r_{ij} = \Big(a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}\Big)/r_{ii}, \quad i < j, \qquad r_{jj} = \Big(a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2\Big)^{1/2}.$$

If properly sequenced, these equations can be used in a recursive fashion to compute the elements in R. For example the elements in R can be determined one row or one column at a time.

Algorithm 7.5. *Cholesky Algorithm; column-wise order.*

for $j = 1 : n$
  for $i = 1 : j - 1$

[19] André-Louis Cholesky (1875–1918) was a French military officer involved in geodesy and surveying in Crete and North Africa just before World War I. He developed the algorithm named after him and his work was posthumously published by a fellow officer, Benoit in 1924.

$$r_{ij} = \Big(a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}\Big)/r_{ii};$$

end

$$r_{jj} = \Big(a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2\Big)^{1/2};$$

end

The column-wise ordering has the advantage of giving the Cholesky factors of all leading principal submatrices of A. An algorithm which computes the elements of R in row-wise order is obtained by reversing the two loops in the code above.

Algorithm 7.6. *Cholesky Algorithm; row-wise order.*

for $i = 1 : n$

$$r_{ii} = \Big(a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2\Big)^{1/2};$$

for $j = i + 1 : n$

$$r_{ij} = \Big(a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}\Big)/r_{ii};$$

end

end

These two versions of the Cholesky algorithm are not only mathematically equivalent but also *numerically equivalent*, i.e., they will compute the same Cholesky factor, taking rounding errors into account. In the Cholesky factorization only

$$\begin{pmatrix} 1 & 2 & 4 & 7 & 11 \\ & 3 & 5 & 8 & 12 \\ & & 6 & 9 & 13 \\ & & & 10 & 14 \\ & & & & 15 \end{pmatrix}$$

Figure 7.3.1. *The mapping of array-subscript of an upper triangular matrix of order 5.*

elements in the upper triangular part of A are referenced and only these elements need to be stored. Since most programming languages only support rectangular arrays this means that the lower triangular part of the array holding A is not used. One possibility is then to use the lower half of the array to store R^T and not overwrite the original data. Another option is to store the elements of the upper triangular part of A column-wise in a vector, see Figure 7.3.1, which is known as **packed storage**. This data is then and overwritten by the elements of R during the computations. Using packed storage complicates the index computations somewhat, but should be used when it is important to economizing storage.

Some applications lead to linear systems where $A \in \mathbf{R}^{n\times n}$ is a symmetric positive *semi-definite* matrix ($x^T Ax \geq 0$ for all $x \neq 0$) with $\operatorname{rank}(A) = r < n$. One example is rank deficient least squares problems; see Sec. 8.5. Another example is when the finite element method is applied to a problem where rigid body motion occurs, which implies $r \leq n-1$. In the semi-definite case a Cholesky factorization still exists, but symmetric pivoting needs to be incorporated. In the kth elimination step *a maximal diagonal element* $a_{ss}^{(k)}$ in the reduced matrix $A^{(k)}$ is chosen as pivot, i.e.,

$$a_{ss}^{(k)} = \max_{k\leq i\leq n} a_{ii}^{(k)}. \tag{7.3.9}$$

This pivoting strategy is easily implemented in Algorithm 7.3.1, the outer product version. Symmetric pivoting is also beneficial when A is close to a rank deficient matrix.

Since all reduced matrices are positive semi-definite their largest element lies on the diagonal. Hence, diagonal pivoting is equivalent to complete pivoting in Gaussian elimination. In exact computation the Cholesky algorithm stops when all diagonal elements in the reduced matrix are zero. This implies that the reduced matrix is the zero matrix.

If A has rank $r < n$ the resulting Cholesky factorization has the upper trapezoidal form

$$P^T AP = R^T R, \qquad R = \begin{pmatrix} R_{11} & R_{12} \end{pmatrix} \tag{7.3.10}$$

where P is a permutation matrix and $R_{11} \in \mathbf{R}^{r\times r}$ with positive diagonal elements. The linear system $Ax = b$, or $P^T AP(P^T x) = P^T b$, then becomes

$$R^T R\tilde{x} = \tilde{b}, \quad \tilde{x} = P^T x, \quad \tilde{b} = P^T b.$$

Setting $z = R\tilde{x}$ the linear system reads

$$R^T z = \begin{pmatrix} R_{11}^T \\ R_{12}^T \end{pmatrix} z = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix},$$

and from the first r equations we obtain $z = R_{11}^{-T}\tilde{b}_1$. Substituting this in the last $n-r$ equations we get

$$0 = R_{12}^T z - \tilde{b}_2 = \begin{pmatrix} R_{12}^T R_{11}^{-T} & -I \end{pmatrix} \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix}.$$

These equations are equivalent to $b \perp \mathcal{N}(A)$ and express the condition for the linear system $Ax = b$ to be consistent. If they are not satisfied a solution does not exist. It remains to solve $L^T\tilde{x} = z$, which gives

$$R_{11}\tilde{x}_1 = z - R_{12}\tilde{x}_2.$$

For an arbitrarily chosen $\tilde{x}_2$ we can uniquely determine $\tilde{x}_1$ so that these equations are satisfied. This expresses the fact that a consistent singular system has an infinite number of solutions. Finally, the permutations are undone to obtain $x = P\tilde{x}$.

Rounding errors can cause negative elements to appear on the diagonal in the Cholesky algorithm even when A is positive semi-definite. Similarly, because of rounding errors the reduced matrix will in general be nonzero after r steps even when $\operatorname{rank}(A) = r$. The question arises when to terminate the Cholesky factorization of a semi-definite matrix. One possibility is to stop when

$$\max_{k \le i \le n} a_{ii}^{(k)} \le 0,$$

and set $\operatorname{rank}(A) = k - 1$. But this may cause unnecessary work in eliminating negligible elements. Two other stopping criteria are suggested in [211, Sec. 10.3.2]. Taking computational cost into consideration it is recommended that the stopping criterion

$$\max_{k \le i \le n} a_{ii}^{(k)} \le \epsilon\, r_{11}^2 \tag{7.3.11}$$

is used, where $\epsilon = nu$ and u is the unit roundoff.

7.3.3 Inertia of Symmetric Matrices

Let $A \in \mathbf{C}^{n\times n}$ be an Hermitian matrix. The **inertia** of A is defined as the number triple $\operatorname{in}(A) = (\pi, \nu, \delta)$ of positive, negative, and zero eigenvalues of A. If A is positive definite matrix and $Ax = \lambda x$, we have

$$x^H Ax = \lambda x^H x > 0.$$

Hence, all eigenvalues must be positive and the inertia is $(n, 0, 0)$.

Hermitian matrices arise naturally in the study of quadratic forms $\psi(x) = x^H Ax$. By the coordinate transformation $x = Ty$ this quadratic form is transformed into

$$\psi(Ty) = y^H \hat{A} y, \quad \hat{A} = T^H AT.$$

The mapping of A onto $T^H AT$ is called a **congruence transformation** of A, and we say that A and $\hat{A}$ are **congruent**. (Notice that a congruence transformation with a nonsingular matrix means a transformation to a coordinate system which is usually not rectangular.) Unless T is unitary these transformations do not, in general, preserve eigenvalues. However, Sylvester's famous law of inertia says that the *signs of eigenvalues are preserved by congruence transformations.*

Theorem 7.3.8. Sylvester's Law of Inertia *If $A \in \mathbf{C}^{n\times n}$ is symmetric and $T \in \mathbf{C}^{n\times n}$ is nonsingular then A and $\hat{A} = T^H AT$ have the same inertia.*

Proof. Since A and $\hat{A}$ are Hermitian there exist unitary matrices U and $\hat{U}$ such that

$$U^H AU = D, \qquad \hat{U}^H \hat{A} \hat{U} = \hat{D},$$

where $D = \operatorname{diag}(\lambda_i)$ and $\hat{D} = \operatorname{diag}(\hat{\lambda}_i)$ are diagonal matrices of eigenvalues. By definition we have $\operatorname{in}(A) = \operatorname{in}(D)$, $\operatorname{in}(\hat{A}) = \operatorname{in}(\hat{D})$, and hence, we want to prove that $\operatorname{in}(D) = \operatorname{in}(\hat{D})$, where

$$\hat{D} = S^H DS, \quad S = U^H T\hat{U}.$$

Assume that $\pi \neq \hat{\pi}$, say $\pi \ > \hat{\pi}$, and that the eigenvalues are ordered so that $\lambda_j > 0$ for $j \leq \pi$ and $\hat{\lambda}_j > 0$ for $j \leq \hat{\pi}$. Let $x = S\hat{x}$ and consider the quadratic form $\psi(x) = x^H D x = \hat{x}^H \hat{D}\hat{x}$, or

$$\psi(x) = \sum_{j=1}^{n} \lambda_j |\xi_j|^2 = \sum_{j=1}^{n} \hat{\lambda}_j |\hat{\xi}_j|^2.$$

Let $x^* \neq 0$ be a solution to the $n - \pi + \hat{\pi} < n$ homogeneous linear relations

$$\xi_j = 0, \quad j > \pi, \qquad \hat{\xi}_j = (S^{-1}x)_j = 0, \quad j \leq \hat{\pi}.$$

Then

$$\psi(x^*) = \sum_{j=1}^{\pi} \lambda_j |\xi_j^*|^2 > 0, \quad \psi(x^*) = \sum_{j=\hat{\pi}}^{n} \hat{\lambda}_j |\hat{\xi}_j^*|^2 \leq 0.$$

This is a contradiction and hence the assumption that $\pi \neq \hat{\pi}$ is false, so A and $\hat{A}$ have the same number of positive eigenvalues. Using the same argument on $-A$ it follows that also $\nu = \hat{\nu}$, and since the number of eigenvalues is the same $\delta = \hat{\delta}$. □

Let $A \in \mathbf{R}^{n\times n}$ be a real symmetric matrix and consider the quadratic equation

$$x^T A x - 2bx = c, \quad A \neq 0. \tag{7.3.12}$$

The solution sets of this equation are sometimes called **conical sections**. If $b = 0$, then the surface has its center at the origin and equation (7.3.12) reads $x^T A x = c$. The inertia of A completely determines the geometric type of the conical section.

Sylvester's theorem tells us that the geometric type of the surface can be determined without computing the eigenvalues. Since we can always multiply the equation by -1 we can assume that there are at least one positive eigenvalue. Then, for $n = 2$ there are three possibilities:

$$(2, 0, 0) \text{ ellipse}; \quad (1, 0, 1) \text{ parabola}; \quad (1, 1, 0) \text{ hyperbola}.$$

In n dimensions there will be $n(n+1)/2$ cases, assuming that at least one eigenvalue is positive.

7.3.4 Symmetric Indefinite Matrices

As shown by Example 7.3.1, the LDL^T factorization of a symmetric indefinite matrix, although efficient computationally, may not exist and can be ill-conditioned. This is true even when symmetric row and column interchanges are used, to select at each stage the largest diagonal element in the reduced matrix as pivot. One stable way of factorizing an indefinite matrix is, of course, to compute an unsymmetric LU factorization using Gaussian elimination with partial pivoting. However, this factorization does not give the inertia of A and needs twice the storage space.

The following example shows that in order to enable a stable $\mathrm{LDL^T}$ factorization for a symmetric *indefinite* matrix A, it is necessary to consider a block factorization where D is block diagonal with also 2×2 diagonal blocks..

Example 7.3.3.

The symmetric matrix

$$A = \begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix}, \quad 0 < \epsilon \ll 1,$$

is indefinite since $\det(A) = \lambda_1 \lambda_2 = \epsilon^1 - 1 < 0$. If we compute the LDL^T factorization of A without pivoting we obtain

$$A = \begin{pmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{pmatrix} \begin{pmatrix} \epsilon & 0 \\ 0 & \epsilon - \epsilon^{-1} \end{pmatrix} \begin{pmatrix} 1 & \epsilon^{-1} \\ 0 & 1 \end{pmatrix}.$$

which shows that there is unbounded element growth. However, A is well conditioned with inverse

$$A^{-1} = \frac{1}{\epsilon^2 - 1} \begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix}, \quad 0 < \epsilon \ll 1.$$

It is quite straightforward to generalize Gaussian elimination to use any nonsingular 2×2 principal submatrix as pivot. By a symmetric permutation this submatrix is brought to the upper left corner, and the permuted matrix partitioned as

$$PAP^T = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix}, \quad A_{11} = \begin{pmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{pmatrix}.$$

Then the Schur complement of A_{11}, $S = A_{22} - A_{12}^T A_{11}^{-1} A_{12}$, exists where

$$A_{11}^{-1} = \frac{1}{\delta_{12}} \begin{pmatrix} a_{22} & -a_{21} \\ -a_{21} & a_{11} \end{pmatrix}, \quad \delta_{12} = \det(A_{11}) = a_{11}a_{22} - a_{21}^2. \tag{7.3.13}$$

We obtain the symmetric block factorization

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ L & I \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & L^T \\ 0 & I \end{pmatrix}, \tag{7.3.14}$$

where $L = A_{12}^T A_{11}^{-1}$. This determines the first two columns of a unit lower triangular matrix $L = L_{21} = A_{21} A_{11}^{-1}$, in an LDL^T factorization of A. The block A_{22} is transformed into the symmetric matrix $A_{22}^{(3)} = A_{22} - L_{21} A_{21}^T$ with components

$$a_{ij}^{(3)} = a_{ij} - l_{i1} a_{1j} - l_{i2} a_{2j}, \quad 2 \le j \le i \le n. \tag{7.3.15}$$

It can be shown that $A_{22}^{(3)}$ is the same reduced matrix as if two steps of Gaussian elimination were taken, first pivoting on the element a_{12} and then on a_{21}.

A similar reduction is used if 2×2 pivots are taken at a later stage in the factorization. Ultimately a factorization $A = LDL^T$ is computed in which D is block diagonal with in general a mixture of 1×1 and 2×2 blocks, and L is unit lower triangular with $l_{k+1,k} = 0$ when $A^{(k)}$ is reduced by a 2×2 pivot. Since the effect of taking a 2×2 step is to reduce A by the equivalent of *two* 1×1 pivot steps, the amount of work must be balanced against that. The part of the calculation which dominates the operation count is (7.3.15), and this is twice the work as for an 1×1 pivot. Therefore, the leading term in the operations count is always $n^3/6$, whichever type of pivots is used.

The main issue then is to find a pivotal strategy that will give control of element growth without requiring too much search. One possible strategy is comparable to that of complete pivoting. Consider the first stage of the factorization and set

$$\mu_0 = \max_{ij} |a_{ij}| = |a_{pq}|, \qquad \mu_1 = \max_{i} |a_{ii}| = |a_{rr}|.$$

Then if

$$\mu_1/\mu_0 > \alpha = (\sqrt{17} + 1)/8 \approx 0.6404,$$

the diagonal element a_{rr} is taken as an 1×1 pivot. Otherwise the 2×2 pivot.

$$\begin{pmatrix} a_{pp} & a_{qp} \\ a_{qp} & a_{qq} \end{pmatrix}, \quad p < q,$$

is chosen. In other words if there is a diagonal element not much smaller than the element of maximum magnitude this is taken as an 1×1 pivot. The magical number α has been chosen so as to minimize the bound on the growth per stage of elements of A, allowing for the fact that a 2×2 pivot is equivalent to two stages. The derivation, which is straight forward but tedious (see Higham [211, Sec. 11.1.1]) is omitted here.

With this choice the element growth can be shown to be bounded by

$$\rho_n \leq (1 + 1/\alpha)^{n-1} < (2.57)^{n-1}. \tag{7.3.16}$$

This exponential growth may seem alarming, but the important fact is that the reduced matrices cannot grow abruptly from step to step. No example is known where significant element growth occur at every step. The bound in (7.3.16) can be compared to the bound 2^{n-1}, which holds for Gaussian elimination with partial pivoting. The elements in L can be bounded by $1/(1-\alpha) < 2.781$ and this pivoting strategy therefore gives a backward stable factorization.

Since the complete pivoting strategy above requires the whole active submatrix to be searched in each stage, it requires $O(n^3)$ comparisons. The same bound for element growth (7.3.16) can be achieved using the following partial pivoting strategy due to Bunch and Kaufman [55, 1977]. For simplicity of notations we restrict our attention to the first stage of the elimination. All later stages proceed similarly. First determine the off-diagonal element of largest magnitude in the first column,

$$\lambda = |a_{r1}| = \max_{i \neq 1} |a_{i1}|.$$

If $|a_{11}| \geq \alpha\lambda$, then take a_{11} as pivot. Else, determine the largest off-diagonal element in column r,

$$\sigma = \max_{1 \leq i \leq n} |a_{ir}|, \quad i \neq r.$$

If $|a_{11}| \geq \alpha\lambda^2/\sigma$, then again take a_{11} as pivot, else if $|a_{rr}| \geq \alpha\sigma$, take a_{rr} as pivot. Otherwise take as pivot the 2×2 principal submatrix

$$\begin{pmatrix} a_{11} & a_{1r} \\ a_{1r} & a_{rr} \end{pmatrix}.$$

Note that at most 2 columns need to be searched in each step, and at most $O(n^2)$ comparisons are needed in all.

Normwise backward stability can be shown to hold also for the Bunch–Kaufman pivoting strategy. However, it is no longer true that the elements of L are bounded independently of A. The following example (Higham [211, Sec. 11.1.2]) shows that L is unbounded:

$$A = \begin{pmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 0 & 1 & \\ \epsilon^{-1} & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & \epsilon & \\ \epsilon & 0 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \epsilon^{-1} \\ & 1 & 0 \\ & & 1 \end{pmatrix}. \tag{7.3.17}$$

Note that whenever a 2×2 pivot is used, we have

$$a_{11}a_{rr} \leq \alpha^2 |a_{1r}|^2 < |a_{1r}|^2.$$

Hence, with both pivoting strategies any 2×2 block in the block diagonal matrix D has a negative determinant $\delta_{1r} = a_{11}a_{rr} - a_{1r}^2 < 0$ and by Sylvester's Theorem corresponds to one positive and one negative eigenvalue. Hence, a 2×2 pivot cannot occur if A is positive definite and in this case all pivots chosen by the Bunch–Kaufman strategy will be 1×1.

For solving a linear system $Ax = b$ the $\mathrm{LDL}^{\mathrm{T}}$ factorization produced by the Bunch–Kaufman pivoting strategy is satisfactory. For certain other applications the possibility of a large L factor is not acceptable. A bounded L factor can be achieved with the modified pivoting strategy suggested in [10, 1998]. This symmetric pivoting is roughly similar to rook pivoting and has a total cost of between $O(n^2)$ and $O(n^3)$ comparisons. Probabilistic results suggest that on the average the cost is only $O(n^2)$. In this strategy a search is performed until two indices r and s have been found such that the element a_{rs} bounds in modulus the other off-diagonal elements in the r and s columns (rows). Then either the 2×2 pivot D_{rs} or the largest in modulus of the two diagonal elements as an 1×1 pivot is taken, according to the test

$$\max(|a_{rr}|, |a_{ss}|) \geq \alpha |a_{rs}|.$$

Aasen [1, 1971] has given an algorithm that for a symmetric matrix $A \in \mathbf{R}^{n \times n}$ computes the factorization

$$PAP^T = LTL^T, \tag{7.3.18}$$

where L is unit lower triangular and T symmetric tridiagonal.

None of the algorithms described here preserves the band structure of the matrix A. In this case Gaussian elimination with partial pivoting can be used but as remarked before this will destroy symmetry and does not reveal the inertia. For the special case of a *tridiagonal* symmetric indefinite matrices an algorithm for computing an $\mathrm{LDL}^{\mathrm{T}}$ factorization will be given in Sec. 7.4.3.

A block $\mathrm{LDL^T}$ factorization can also be computed for a real skew-symmetric matrix A. Note that $A^T = -A$ implies that such a matrix has zero diagonal elements. Further, since

$$(x^T Ax)^T = x^T A^T x = -x^T Ax,$$

it follows that all nonzero eigenvalues come in pure imaginary complex conjugate pairs. In the first step of the factorization if the first column is zero there is nothing to do. Otherwise we look for an off-diagonal element $a_{p,q}$, $p > q$ such that

$$|a_{p,q}| = \max\{\max_{1<i\le n} |a_{i,1}|, \max_{1<i\le n} |a_{i,2}|\},$$

and take the 2×2 pivot

$$\begin{pmatrix} 0 & -a_{p,q} \\ a_{p,q} & 0 \end{pmatrix}.$$

It can be shown that with this pivoting the growth ratio is bounded by $\rho_n \le (\sqrt{3})^{n-2}$, which is smaller than for Gaussian elimination with partial pivoting for a general matrix.

Review Questions

3.1 (a) Give two necessary and sufficient conditions for a real symmetric matrix A to be positive definite.

(b) Show that if A is symmetric positive definite so is its inverse A^{-1}.

3.2 What simplifications occur in Gaussian elimination applied to a symmetric, positive definite matrix?

3.3 What is the relation of Cholesky factorization to Gaussian elimination? Give an example of a symmetric matrix A for which the Cholesky decomposition does not exist.

3.4 Show that if A is skew-symmetric, then iA is Hermitian.

3.5 Show that the Cholesky factorization is unique for positive definite matrices provided R is normalized to have positive diagonal entries.

3.6 (a) Formulate and prove Sylvester's law of inertia.

(b) Show that for $n = 3$ there are six different geometric types of conical sections $x^T Ax - 2b^T x = c$, provided that $A \neq 0$ and is normalized to have at least one positive eigenvalue.

Problems

3.1 If A is a symmetric positive definite matrix how should you compute $x^T Ax$ for a given vector x?

3.2 Let the matrix A be symmetric and positive definite. Show that $|a_{ij}| \leq (a_{ii} + a_{jj})/2$.

3.3 Show by computing the Cholesky factorization $A = LL^T$ that the matrix

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$$

is positive definite.

3.4 Let $A = (a_{ij}) \in \mathbf{R}^{n\times n}$ be a symmetric positive definite matrix. Prove the special case of Hadamard's inequality

$$|\det A| \leq \prod_{i=1}^{n} a_{ii}. \tag{7.3.19}$$

where equality holds only if A is diagonal.

Hint: Use the Cholesky decomposition $A = R^T R$ and show that $\det A = (\det R)^2$.

3.5 The Hilbert matrix $H_n \in \mathbf{R}^{n\times n}$ with elements

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

is symmetric positive definite for all n. Denote by $\bar{H}_4$ the corresponding matrix with elements rounded to five decimal places, and compute its Cholesky factor $\bar{L}$. Then compute the difference $(\bar{L}\bar{L}^T - \bar{A})$ and compare it with $(A - \bar{A})$.

3.6 Let $A + iB$ be Hermitian and positive definite, where $A, B \in \mathbf{R}^{n\times n}$. Show that the real matrix

$$C = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}$$

is symmetric and positive definite. How can a linear system $(A+iB)(x+iy) = b + ic$ be solved using a Cholesky factorization of C?

3.7 Implement the Cholesky factorization using packed storage for A and R.

7.4 Banded Linear Systems

7.4.1 Multiplication of Banded Matrices

We recall (see (7.1.10)) that a matrix A is said to have upper bandwidth r and lower bandwidth s if

$$a_{ij} = 0, \quad j > i + r, \qquad a_{ij} = 0, \quad i > j + s,$$

respectively. This means that the number of nonzero diagonals above and below the main diagonal are r and s respectively. The maximum number of nonzero elements in any row is then $w = r + s + 1$, which is the **bandwidth** of A.

Linear systems $Ax = b$ where the matrix A has a small bandwidth arise in problems where each variable x_i is coupled by an equation only to a few other variables x_j such that $|j-i|$ is small. Note that the bandwidth of a matrix depends on the ordering of its rows and columns. An important, but hard, problem is to find an optimal ordering of columns that minimize the bandwidth. However, there are good heuristic algorithms that can be used in practice and give almost optimal results; see Sec. 7.6.2.

Clearly the product of two diagonal matrices D_1 and D_2 is another diagonal matrix where the elements are equal to the elementwise product of the diagonals. What can said of the product of two banded matrices? An elementary, but very useful result tells us which diagonals in the product are nonzero.

Lemma 7.4.1.

Let $A, B \in \mathbf{R}^{n\times n}$ have lower (upper) bandwidth r and s respectively. Then the product AB has lower (upper) bandwidth $r + s$.

Assume that A and B are banded matrices of order n, which both have a small bandwidth compared to n. Then, since there are few nonzero elements in the rows and columns of A and B the usual algorithms for forming the product AB are not effective on vector computers. We now give an algorithm for multiplying matrices by diagonals, which overcomes this drawback. The idea is to write A and B as a sum of their diagonals and multiply crosswise. This gives the following rule:

Lemma 7.4.2.

Let $A = \operatorname{diag}(a, r)$ and $B = \operatorname{diag}(b, s)$ and set $C = AB$. If $|r + s| \ge n$ then $C = 0$; otherwise $C = \operatorname{diag}(c, r+s)$, where the elements of the vector $c \in \mathbf{R}^{(n-|r+s|)}$ are obtained by pointwise multiplication of shifted vectors a and b:

$$c = \begin{cases} (a_1 b_{r+1}, \dots, a_{n-r-s} b_{n-s})^T, & \textit{if } r, s \ge 0, \\ (a_{|s|+1} b_1, \dots, a_{n-|r|} b_{n-|r+s|})^T, & \textit{if } r, s \le 0. \\ (0, \dots, 0, a_1 b_1, \dots, a_{n-s} b_{n-s})^T, & \textit{if } r < 0, \quad s > 0, \quad r + s \ge 0. \\ (0, \dots, 0, a_1 b_1, \dots, a_{n-|r|} b_{n-|r|})^T, & \textit{if } r < 0, \quad s > 0, \quad r + s < 0. \\ (a_1 b_{|r+s|+1}, \dots, a_{n-r} b_{n-|s|}, 0, \dots, 0)^T, & \textit{if } r > 0, \quad s < 0, \quad r + s \ge 0. \\ (a_{r+1} b_1, \dots, a_{n-r} b_{n-|s|}, 0, \dots, 0)^T, & \textit{if } r > 0, \quad s < 0, \quad r + s < 0. \end{cases} \tag{7.4.1}$$

Note that when $rs < 0$, zeros are added at the beginning or end to get a vector c of length $n - |r + s|$.

Example 7.4.1.

The number of cases in this lemma looks a bit forbidding, so to clarify the result we consider a specific case. Let A and B be tridiagonal matrices of size $n \times n$

$$
A = \begin{pmatrix} a_1 & c_1 & & & \\ b_1 & a_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2} & a_{n-1} & c_{n-1} \\ & & & b_{n-1} & a_n \end{pmatrix}, \qquad B = \begin{pmatrix} d_1 & f_1 & & & \\ e_1 & d_2 & f_2 & & \\ & \ddots & \ddots & \ddots & \\ & & e_{n-2} & d_{n-1} & f_{n-1} \\ & & & e_{n-1} & d_n \end{pmatrix}.
$$

Then $C = AB$ will be a banded matrix of upper and lower bandwidth two. The five nonzero diagonals of C are

$$
\begin{aligned}
\operatorname{diag}(C,0) = {} & a(1:n).*d(1:n) + [0,\, b(1:n-1).*f(1:n-1)] \\
& + [c(1:n-1).*e(1:n-1),\, 0], \\
\operatorname{diag}(C,1) = {} & a(1:n-1).*f(1:n-1) + c(1:n-1).*d(2:n)] \\
\operatorname{diag}(C,-1) = {} & b(1:n-1).*d(1:n-1) + a(2:n).*e(1:n-1)] \\
\operatorname{diag}(C,2) = {} & c(1:n-2).*f(2:n-1) \\
\operatorname{diag}(C,-2) = {} & b(2:n-1).*e(1:n-2)
\end{aligned}
$$

The number of operations are exactly the same as in the conventional schemes, but only $3^2 = 9$ pointwise vector multiplications are required.

7.4.2 LU Factorization of Banded Matrices

A matrix A for which all nonzero elements are located in consecutive diagonals is called a **band matrix**.

Many applications give rise to linear systems $Ax = b$, where the nonzero elements in the matrix A are located in a band centered along the principal diagonal. Such matrices are called **band matrices** and are the simplest examples of **sparse** matrices, i.e., matrices where only a small proportion of the n^2 elements are nonzero. Such matrices arise frequently in, for example, the numerical solution of boundary value problems for ordinary and partial differential equations.

Band matrices are well-suited for Gaussian elimination, since if no pivoting is required *the band structure is preserved*. Recall that pivoting is not needed for stability, e.g., when A is diagonally dominant.

Theorem 7.4.3. *Let A be a band matrix with upper bandwidth r and lower band width s. If A has an LU-decomposition $A = LU$, then U has upper bandwidth r and L lower bandwidth s.*

Proof. The factors L and U are unique and can be computed, for example, by Doolittle's method (7.2.17). Assume that the first $k-1$ rows of U and columns of L have bandwidth r and s, that is, for $p = 1 : k-1$

$$
l_{ip} = 0, \quad i > p + s, \qquad u_{pj} = 0, \quad j > p + r. \tag{7.4.2}
$$

The proof is by induction in k. The assumption is trivially true for $k = 1$. Since $a_{kj} = 0$, $j > k + r$ we have from (7.2.11) and (7.4.2)

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj} = 0 - 0 = 0, \quad j > k + r.$$

Similarly, it follows that $l_{ik} = 0$, $i > k+s$, which completes the induction step. $\square$

A band matrix $A \in \mathbf{R}^{n\times n}$ may be stored by diagonals in an array of dimension $n \times (r + s + 1)$ or $(r + s + 1) \times n$. For example, the matrix above can be stored as

$$\begin{matrix} * & * & a_{11} & a_{12} \\ * & a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{42} & a_{43} & a_{44} & a_{45} \\ a_{53} & a_{54} & a_{55} & a_{56} \\ a_{64} & a_{65} & a_{66} & * \end{matrix}, \quad \text{or} \quad \begin{matrix} * & a_{12} & a_{23} & a_{34} & a_{45} & a_{56} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\ a_{21} & a_{32} & a_{43} & a_{54} & a_{65} & * \\ a_{31} & a_{42} & a_{53} & a_{64} & * & * \end{matrix}.$$

Notice that except for a few elements indicated by asterisks in the initial and final rows, only nonzero elements of A are stored. For example, passing along a row in the second storage scheme above moves along a diagonal of the matrix, and the columns are aligned.

For a general band matrix Algorithm 7.2.1, Gaussian elimination without pivoting, should be modified as follows to operate only on nonzero elements: The algorithms given below are written as if the matrix was conventionally stored. It is a useful exercise to rewrite them for the case when A, L, and U are stored by diagonals!

Algorithm 7.7. *Banded Gaussian Elimination.*
Let $A =\in \mathbf{R}^{n\times n}$ be a given matrix with upper bandwidth r and lower bandwidth s. The following algorithm computes the LU factorization of A, *provided it exists.* The element a_{ij} is overwritten by l_{ij} if $i > j$ and by u_{ij} otherwise.

for $k = 1 : n - 1$
  for $i = k + 1 : \min(k + s, n)$
    $l_{ik} := a_{ik}^{(k)} / a_{kk}^{(k)}$;
    for $j = k + 1 : min(k + r, n)$
      $a_{ij}^{(k+1)} := a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$;
    end
  end
end

An operation count shows that this algorithm requires t flops, where

$$t = \begin{cases} 2nr(s+1) - rs^2 - \frac{1}{3}r^3, & \text{if } r \le s; \\ 2ns(s+1) - \frac{4}{3}s^3, & \text{if } r = s; \\ 2ns(r+1) - sr^2 - \frac{1}{3}s^3, & \text{if } r > s. \end{cases}$$

Whenever $r \ll n$ *or* $s \ll n$ this is much less than the $2n^3/3$ flops required in the full case.

Analogous savings can be made in forward- and back-substitution. Let L and U be the triangular factors computed by Algorithm 7.4.2. The solution of the two banded triangular systems $Ly = b$ and $Ux = y$ are obtained from

$$y_i = b_i - \sum_{max(1,i-s)}^{i-1} l_{ij} y_j, \quad i = 1 : n$$

$$x_i := \Big(y_i - \sum_{j=i+1}^{\min(i+r,n)} u_{ij} x_j\Big)/u_{ii}, \quad i = n : (-1) : 1.$$

These algorithms require $2ns - s^2$ and $(2n - r)(r + 1)$ flops, respectively. They are easily modified so that y and x overwrites b in storage.

Unless A is diagonally dominant or symmetric positive definite, partial pivoting should be used. The pivoting will cause the introduction of elements outside the band. This is illustrated below for the case when $s = 2$ and $r = 1$. The first step of the elimination is shown, where it is assumed that a_{31} is chosen as pivot and therefore rows 1 and 3 interchanged:

$$\begin{matrix} a_{31} & a_{32} & a_{33} & a_{34} & & \\ a_{21} & a_{22} & a_{23} & & & \\ a_{11} & a_{12} & & & & \\ & a_{42} & a_{43} & a_{44} & a_{45} & \\ & & a_{53} & a_{54} & a_{55} & a_{56} \\ & & & \cdots & & \end{matrix} \quad \Longrightarrow \quad \begin{matrix} u_{11} & u_{12} & u_{13} & u_{14} & & \\ l_{21} & a_{22}^{(2)} & a_{23}^{(2)} & \mathbf{a}_{24}^{(2)} & & \\ l_{31} & a_{32}^{(2)} & \mathbf{a}_{33}^{(2)} & \mathbf{a}_{34}^{(2)} & & \\ & a_{42} & a_{43} & a_{44} & a_{45} & \\ & & a_{53} & a_{54} & a_{55} & a_{56} \\ & & & \cdots & & \end{matrix} .$$

where fill-in elements are shown in boldface. Hence, *the upper bandwidth of U may increase to $r + s$. The matrix L will still have only s elements below the main diagonal in all columns but no useful band structure.* This can be seen from the example above where, e.g., the elements l_{21} and l_{31} may be subject to later permutations, destroying the band-structure of the first column.

Example 7.4.2.

A class of matrices with unsymmetric band structure is upper (lower) Hessenberg matrices for which $s = 1$ ($r = 1$). These are of particular interest in connection with unsymmetric eigenvalue problems. An upper Hessenberg matrix of order five has the structure

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ 0 & h_{32} & h_{33} & h_{34} & h_{35} \\ 0 & 0 & h_{43} & h_{44} & h_{45} \\ 0 & 0 & 0 & h_{54} & h_{55} \end{pmatrix}.$$

Performing Gaussian elimination the first step will only affect the first two rows of the matrix. The reduced matrix is again Hessenberg and all the remaining steps are similar to the first. If partial pivoting is used then in the first step either h_{11} or h_{21} will be chosen as pivot. Since these rows have the same structure the Hessenberg form will be preserved during the elimination. Clearly only $t = n(n+1)$ flops are needed. Note that with partial pivoting the elimination will not give a factorization $PA = LU$ with L lower bidiagonal. Whenever we pivot, the interchanges should be applied also to L, which will spread out the elements. Therefore, L will be lower triangular with only one nonzero off-diagonal element in each column. However, it is more convenient to leave the elements in L in place.

If $A \in \mathbf{R}^{n\times n}$ is Hessenberg then $\rho_n \leq n$ with partial pivoting. This follows since at the start of the k stage row $k+1$ of the reduced matrix has not been changed and the pivot row has elements of modulus at most k times the largest element of H.

In the special case when A is a symmetric positive definite banded matrix with upper and lower bandwidth $r = s$, the factor L in the Cholesky factorization $A = LL^T$ has lower bandwidth r. From Algorithm 7.3.2 we easily derive the following banded version:

Algorithm 7.8. *Band Cholesky Algorithm.*

for $j\ = 1 : n$
  $p = \max(1, j - r);$
  for $i = p : j - 1$
    $$r_{ij} = \Big(a_{ij} - \sum_{k=p}^{i-1} r_{ki} r_{kj}\Big)/r_{ii};$$
  end
  $$r_{jj} = \Big(a_{jj} - \sum_{k=p}^{j-1} r_{kj}^2\Big)^{1/2};$$
end

If $r \ll n$ this algorithm requires about $\frac{1}{2}nr(r+3)$ flops and n square roots. As input we just need the upper triangular part of A, which can be stored in an $n \times (r+1)$ array.

7.4.3 Tridiagonal Linear Systems

A matrix of the form

$$A = \begin{pmatrix} a_1 & c_2 & & & \\ b_2 & a_2 & c_3 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-1} & a_{n-1} & c_n \\ & & & b_n & a_n \end{pmatrix}. \tag{7.4.3}$$

is called **tridiagonal**. Note that the $3n-2$ nonzero elements in A are conveniently stored in three vectors a, c, and d. A is said to be irreducible if b_i and c_i are nonzero for $i = 2 : n$. Let A be reducible, say $c_k = 0$. Then A can be written as a lower block triangular form

$$A = \begin{pmatrix} A_1 & 0 \\ L_1 & A_2 \end{pmatrix},$$

where A_1 and A_2 are tridiagonal. If A_1 or A_2 is reducible then this blocking can be applied recursively until a block form with irreducible tridiagonal blocks is obtained..

If Gaussian elimination with partial pivoting is applied to A then a factorization $PA = LU$ is obtained, where L has at most one nonzero element below the diagonal in each column and U has upper bandwidth two (cf. the Hessenberg case in Example 7.4.2). If A is diagonally dominant, then no pivoting is required and the factorization $A = LU$ exists. By Theorem 7.4.3 it has the form

$$A = LU = \begin{pmatrix} 1 & & & & \\ \gamma_2 & 1 & & & \\ & \gamma_3 & \ddots & & \\ & & \ddots & 1 & \\ & & & \gamma_n & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 & c_2 & & & \\ & \alpha_2 & c_3 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{n-1} & c_n \\ & & & & \alpha_n \end{pmatrix}. \tag{7.4.4}$$

By equating elements in A and LU it is verified that the upper diagonal in U equals that in A, and for the other elements in L and U we obtain the recursions

$$\alpha_1 = a_1, \quad \gamma_k = b_k/\alpha_{k-1}, \qquad \alpha_k = a_k - \gamma_k c_k, \quad k = 2 : n. \tag{7.4.5}$$

Note that the elements γ_k and α_k can overwrite b_k and a_k, respectively. The solution to the system $Ax = f$ can then be computed by solving $Ly = f$ by $Ux = y$ by back- and forward-substitution

$$y_1 = f_1, \qquad y_i = f_i - \gamma_i y_{i-1}, \quad i = 2 : n, \tag{7.4.6}$$

$$x_n = y_n/\alpha_n, \qquad x_i = (y_i - c_{i+1}x_{i+1})/\alpha_i, \quad i = n-1 : 1. \tag{7.4.7}$$

The total number of flops is about $3n$ for the factorization and $2.5n$ for the solution.

If A is tridiagonal then it is easily proved by induction that $\rho_n \leq 2$ with partial pivoting. This result is a special case of a more general result.

Theorem 7.4.4. [Bothe [47, 1975]] *If $A \in \mathbf{C}^{n\times n}$ has upper and lower bandwidth p then the growth factor in Gaussian elimination with partial pivoting satisfies*

$$\rho_n \le 2^{2p-1} - (p-1)2^{p-2}.$$

In particular, for a tridiagonal matrix ($p = 1$) we have $\rho_n \le 2$.

When A is symmetric positive definite and tridiagonal (7.4.3)

$$A = \begin{pmatrix} a_1 & b_2 & & & \\ b_2 & a_2 & b_3 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-1} & a_{n-1} & b_n \\ & & & b_n & a_n \end{pmatrix}, \tag{7.4.8}$$

we can write the factorization

$$A = LDL^T, \quad D = \operatorname{diag}(\alpha_1, \dots, \alpha_n), \tag{7.4.9}$$

where L is as in (7.4.4). The algorithm then reduces to

$$\alpha_1 = a_1, \quad \gamma_k = b_k/\alpha_{k-1}, \qquad \alpha_k = a_k - \gamma_k b_k, \quad k = 2 : n. \tag{7.4.10}$$

Sometimes it is more convenient to write

$$A = U^T D^{-1} U, \quad D = \operatorname{diag}(a_1, \dots, a_n).$$

In the scalar case U is given by (7.4.4) (with $c_k = b_k$), and the elements in U and D are computed from

$$\alpha_1 = a_1, \qquad \alpha_k = a_k - b_k^2/\alpha_{k-1}. \quad k = 2 : n. \tag{7.4.11}$$

The recursion (7.4.5) for the LU factorization of a tridiagonal matrix is highly serial. An algorithm for solving tridiagonal systems, which has considerable inherent parallelism, is **cyclic reduction** also called **odd-even reduction**. This is the most preferred method for solving large tridiagonal systems on parallel computers.

The basic step in cyclic reduction is to eliminate all the odd unknowns to obtain a reduced tridiagonal system involving only even numbered unknowns. This process is repeated recursively until a system involving only a small order of unknowns remains. This is then solved separately and the other unknowns can then be computed in a back-substitution process. We illustrate this process on a tridiagonal system $Ax = f$ of order $n = 2^3 - 1 = 7$. If P is a permutation matrix such that $P(1, 2, \dots, 7) = (1, 3, 5, 7, 2, 4, 6)^T$ the transformed system $PAP^T(Px) = P^T f$, will have the form

$$\left(\begin{array}{cccc|ccc} a_1 & & & & c_2 & & \\ & a_3 & & & b_3 & c_4 & \\ & & a_5 & & & b_5 & c_6 \\ & & & a_7 & & & b_7 \\ \hline b_2 & c_3 & & & a_2 & & \\ & b_4 & c_5 & & & a_4 & \\ & & b_6 & c_7 & & & a_6 \end{array}\right) \begin{pmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \\ \hline x_2 \\ x_4 \\ x_6 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_3 \\ f_5 \\ f_7 \\ \hline f_2 \\ f_4 \\ f_6 \end{pmatrix}.$$

It is easily verified that after eliminating the odd variables from the even equations the resulting system is again tridiagonal. Rearranging these as before the system becomes

$$\left(\begin{array}{cc|c} a_2' & & c_4' \\ & a_6' & b_6' \\ \hline b_4' & c_6' & a_4' \end{array}\right) = \begin{pmatrix} x_2 \\ x_6 \\ x_4 \end{pmatrix} = \begin{pmatrix} f_2' \\ f_6' \\ f_4' \end{pmatrix}.$$

After elimination we are left with one equation in one variable

$$a_4'' x_4 = f_4''.$$

Solving for x_4 we can compute x_2 and x_6 from the first two equations in the previous system. Substituting these in the first four equations we get the odd unknowns x_1, x_3, x_5, x_7. Clearly this scheme can be generalized. For a system of dimension $n = 2^p - 1$, p steps are required in the reduction. Note, however, that it is possible to stop at any stage, solve a tridiagonal system and obtain the remaining variables by substitution. Therefore, it can be used for any dimension n.

The derivation shows that cyclic reduction is equivalent to Gaussian elimination without pivoting on a reordered system. Thus, it is stable if the matrix is diagonally dominant or symmetric positive definite. In contrast to the conventional algorithm there is some fill in during the elimination and about 2.7 times more operations are needed.

Example 7.4.3.

Consider the linear system $Ax = b$, where A is a symmetric positive definite tridiagonal matrix. Then A has positive diagonal elements and the symmetrically scaled matrix DAD, where $D = \operatorname{diag}(d_1, \ldots, d_n)$, $d_i = 1/\sqrt{a_i}$, has unit diagonal elements. After an odd-even permutation the system has the 2×2 block form

$$\begin{pmatrix} I & F \\ F^T & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix}, \tag{7.4.12}$$

with F lower bidiagonal. After block elimination the Schur complement system becomes

$$(I - F^T F)x = d - F^T c.$$

Here $I - F^T F$ is again a positive definite tridiagonal matrix. Thus, the process can be repeated recursively.

Boundary value problems, where the solution is subject to *periodic boundary conditions*, often lead to matrices of the form

$$B = \left(\begin{array}{ccccc|c} a_1 & c_2 & & & & b_1 \\ b_2 & a_2 & c_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & b_{n-1} & a_{n-1} & c_n \\ \hline c_1 & & & & b_n & a_n \end{array}\right), \tag{7.4.13}$$

which are tridiagonal except for the two corner elements b_1 and c_1. We now consider the real symmetric case, $b_i = c_i$, $i = 1 : n$. Partitioning B in 2×2 block form as above, we seek a factorization

$$B = \begin{pmatrix} A & u \\ v^T & a_n \end{pmatrix} = \begin{pmatrix} L & 0 \\ y^T & 1 \end{pmatrix} \begin{pmatrix} U & z \\ 0 & d_n \end{pmatrix}$$

where $u = b_1 e_1 + c_n e_{n-1}$, $v = c_1 e_1 + b_n e_{n-1}$. Multiplying out we obtain the equations

$$A = LU, \quad u = Lz, \quad v^T = y^T U, \quad a_n = y^T z + d_n$$

Assuming that no pivoting is required the factorization $A = LU$, where L and U are bidiagonal, is obtained using (7.4.5). The vectors y and z are obtained from the lower triangular systems

$$Lz = b_1 e_1 + c_n e_{n-1}, \qquad U^T y = c_1 e_1 + c_n e_{n-1},$$

and $d_n = a_n - y^T z$. Note that y and z will be full vectors.

Cyclic reduction can be applied to systems $Bx = f$, where B has the tridiagonal form in (7.4.13). If n is even the reduced system obtained after eliminating the odd variables in the even equations will again have the form (7.4.13). For example, when $n = 2^3 = 8$ the reordered system is

$$\left(\begin{array}{cccc|cccc} a_1 & & & & c_2 & & & b_1 \\ & a_3 & & & b_3 & c_4 & & \\ & & a_5 & & & b_5 & c_6 & \\ & & & a_7 & & & b_7 & c_8 \\ \hline b_2 & c_3 & & & a_2 & & & \\ & b_4 & c_5 & & & a_4 & & \\ & & b_6 & c_7 & & & a_6 & \\ c_1 & & & b_8 & & & & a_8 \end{array}\right) \left(\begin{array}{c} x_1 \\ x_3 \\ x_5 \\ x_7 \\ \hline x_2 \\ x_4 \\ x_6 \\ x_8 \end{array}\right) = \left(\begin{array}{c} f_1 \\ f_3 \\ f_5 \\ f_7 \\ \hline f_2 \\ f_4 \\ f_6 \\ f_8 \end{array}\right).$$

If $n = 2^p$ the process can applied recursively. After p steps one equation in a single unknown is obtained. Cyclic reduction here does not require extra storage and also has a slightly lower operation count than ordinary Gaussian elimination.

We finally consider the case when A is a **symmetric indefinite tridiagonal** matrix. It would be possible to use LU factorization with partial pivoting, but this destroys symmetry and gives no information about the inertia of A. Instead a block factorization $A = LDL^T$ can be computed using no interchanges as follows. Set $\sigma = \max_{1 \le i \le n} |a_{ij}|$ and $\alpha = (\sqrt{5} - 1)/2 \approx 0.62$. In the first stage we take a_{11} as pivot if $\sigma |a_{11}| \ge a_{21}^2$. Otherwise we take the 2×2 pivot

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

This factorization can be shown to be normwise backward stable and is a good way to solve such symmetric indefinite tridiagonal linear systems.

7.4.4 Inverses of Banded Matrices

It is important to note that *the inverse A^{-1} of a banded matrix in general has no zero elements*. Hence, one should never attempt to explicitly compute the elements of the inverse of a band matrix. Since banded systems often have very large dimensions even storing the elements in A^{-1} may be infeasible!

The following theorem states that the lower triangular part of the inverse of an upper Hessenberg matrix has a very simple structure.

Theorem 7.4.5.
Let $H \in \mathbf{R}^{n\times n}$ be an upper Hessenberg matrix with nonzero elements in the subdiagonal, $h_{i+1,i} \neq 0$, $i = 1 : n-1$. Then there are vectors p and q such that

$$(H^{-1})_{ij} = p_i q_j, \quad i \geq j. \tag{7.4.14}$$

Proof. See Ikebe [223, 1979] $\square$

A tridiagonal matrix A is both lower and upper Hessenberg. Hence, if A is irreducible it follows that there are vectors x, y, p and q such that

$$(A^{-1})_{ij} = \begin{cases} x_i y_j, & i \leq j, \\ p_i q_j, & i \geq j. \end{cases} \tag{7.4.15}$$

Note that $x_1 \neq 0$ and $y_n \neq 0$, since otherwise the entire first row or last column of A^{-1} would be zero, contrary to the assumption of the nonsingularity of A. The vectors x and y (as well as p and q) are unique up to scaling by a nonzero factor. There is some redundancy in this representation since $x_i y_i = p_i q_i$. It can be shown that $3n-2$ parameters are needed to represent the inverse, which equals the number of nonzero elements in A.

The following algorithm has been suggested by N. J. Higham to compute the vectors x, y, p and q:

1. Compute the LU factorization of A.

2. Use the LU factorization to solve for the vectors y and z, where $A^T y = e_1$ and $Az = e_n$. Similarly, solve for p and r, where $Ap = e_1$ and $A^T r = e_n$.

3. Set $q = p_n^{-1} r$ and $x = y_n^{-1} z$.

This algorithm is not foolproof and can fail because of overflow.

Example 7.4.4. Let A be a symmetric, positive definite tridiagonal matrix with elements $a_1 = 1$,

$$a_i = 2, \quad b_i = c_i = -1, \quad i = 2 : 5.$$

Although the Cholesky factor L of A is bidiagonal the inverse

$$A^{-1} = \begin{pmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 4 & 3 & 2 & 1 \\ 3 & 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

is full. Here $x = p$, $y = q$, can be determined up to a scaling factor from the first and last columns of A^{-1}.

The inverse of any banded matrix has a special structure related to low rank matrices. The first study of inverse of general banded matrices was Asplund [11, 1959].

Review Questions

4.1 Give an example of matrix multiplication by diagonals.

4.2 (a) If a is a column vector what is meant by diag (a, k)?

(b) If A is a square matrix what is meant by diag (A, k)?

4.3 (a) Let $A \in \mathbf{R}^{n\times n}$ be a banded matrix with upper bandwidth p and lower bandwidth q. Show how A can be efficiently stored when computing the LU factorization.

(b) Assuming that the LU factorization can be carried out without pivoting, what are the structures of the resulting L and U factors of A?

(c) What can you say about the structure of the inverses of L and U?

4.4 Let $A \in \mathbf{R}^{n\times n}$ be a banded matrix with upper bandwidth p and lower bandwidth q. Assuming that the LU factorization of A can be carried out without pivoting, roughly how many operations are needed? You need only give the dominating term when $p, q \ll n$.

4.5 Give a bound for the growth ratio ρ_n in Gaussian elimination with partial pivoting, when the matrix A is: (a) Hessenberg; (b) tridiagonal.

Problems

4.1 (a) Let $A, B \in \mathbf{R}^{n\times n}$ have lower (upper) bandwidth r and s respectively. Show that the product AB has lower (upper) bandwidth $r + s$.

(b) An upper Hessenberg matrix H is a matrix with lower bandwidth $r = 1$. Using the result in (a) deduce that the product of H and an upper triangular matrix is again an upper Hessenberg matrix.

4.2 Show that an irreducible nonsymmetric tridiagonal matrix A can be written $A = DT$, where T is symmetric tridiagonal and $D = \operatorname{diag}(d_k)$ is diagonal with elements

$$d_1 = 1, \quad d_k = \prod_{j=2}^{k} c_j/b_j, \quad k = 2 : n. \tag{7.4.16}$$

4.3 (a) Let $A \in \mathbf{R}^{n\times n}$ be a symmetric, tridiagonal matrix such that $\det(A_k) \neq 0$, $k = 1 : n$. Then the decomposition $A = LDL^T$ exists and can be computed by the formulas given in (7.4.10). Use this to derive a recursion formula for computing $\det(A_k)$, $k = 1 : n$.

(b) Determine the largest n for which the symmetric, tridiagonal matrix

$$A = \begin{pmatrix} 2 & 1.01 & & & \\ 1.01 & 2 & 1.01 & & \\ & 1.01 & \ddots & \ddots & \\ & & \ddots & \ddots & 1.01 \\ & & & 1.01 & 2 \end{pmatrix} \in \mathbf{R}^{n\times n}$$

is positive definite.

4.4 (a) Show that for $\lambda \geq 2$ it holds that $B = \mu LL^T$, where

$$B = \begin{pmatrix} \mu & -1 & & & \\ -1 & \lambda & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \lambda & -1 \\ & & & -1 & \lambda \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & & & & \\ -\sigma & 1 & & & \\ & -\sigma & \ddots & & \\ & & \ddots & 1 & \\ & & & -\sigma & 1 \end{pmatrix},$$

and

$$\mu = \lambda/2 \pm (\lambda^2/4 - 1)^{1/2}, \quad \sigma = 1/\mu.$$

Note that L has constant diagonals.

(b) Suppose we want to solve the system $Ax = b$, where the matrix A differs from B in the element (1,1),

$$A = B + \delta e_1 e_1^T, \quad \delta = \lambda - \mu, \quad e_1^T = (1, 0, \dots, 0).$$

Show, using the Sherman–Morrison formula (7.1.25), that the solution $x = A^{-1}b$ can be computed from

$$x = y - \gamma L^{-T} f, \quad \gamma = \delta(e_1^T y)/(\mu + \delta f^T f)$$

where y and f satisfies $\mu LL^T y = b$, $Lf = e_1$.

4.5 Consider the symmetric tridiagonal matrix

$$A_n = \begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & 4 & 1 \\ & & & 1 & 4 \end{pmatrix}.$$

For $n = 20, 40$ use the Cholesky factorization of A_n and Higham's algorithm to determine vectors x and y so that $(A_n^{-1})_{ij} = x_i y_j$ for $i, j = 1 : n$. Verify that there is a range of approximately θ^n in the size of the components of these vectors, where $\theta = 2 + \sqrt{3}$.

4.5 (a) Write a function implementing the multiplication $C = AB$, where $A = \operatorname{diag}(a, r)$ and $B = \operatorname{diag}(b, s)$ both consist of a single diagonal. Use the formulas in Lemma 7.4.2.

(b) Write a function for computing the product $C = AB$ of two banded matrices using the $w_1 w_2$ calls to the function in (a), where w_1 and w_2 are the bandwidth of A and B, respectively.

4.6 Derive expressions for computing δ_k, $k = 1 : n-1$ and α_n in the factorization of the periodic tridiagonal matrix A in (7.4.13).

4.7 Let B be a symmetric matrix of the form (7.4.13). Show that

$$B = T + \sigma u u^T, \quad u = (1, 0, \ldots, 0, -1)^T.$$

where T is a certain symmetric, tridiagonal matrix. What is σ and T. Derive an algorithm for computing L by modifying the algorithm (7.4.10).

7.5 Block Algorithms

7.5.1 Linear Algebra Software

The first collection of high quality software was a series of algorithms written in Algol 60 that appeared in a handbook edited by Wilkinson and Reinsch [390, 1971]. This contains 11 subroutines for linear systems, least squares, and linear programming and 18 routines for the algebraic eigenvalue problem.

The collection LINPACK of Fortran subroutines for linear systems that followed contained several important innovations; see Dongarra et al. [107, 1979]. The routines were kept machine independent partly by performing as much of the computations as possible by calls to so-called Basic Linear Algebra Subprograms (BLAS) [258, 1979]. These identified frequently occurring vector operations in linear algebra such as scalar product, adding of a multiple of one vector to another. For example, the operations

$$y := \alpha x + y, \qquad \alpha := \alpha + x^T y$$

in single precision was named SAXPY. By carefully optimizing these BLAS for each specific computer, performance could be enhanced without sacrificing portability.

LINPACK was followed by EISPACK, a collection of routines for the algebraic eigenvalue problem; see Smith et al. [336, 1976], B. S. Garbow et al. [149, 1977].

The original BLAS, now known as level-1 BLAS, were found to be unsatisfactory when vector computers were introduced in the 1980s. This brought about the development of level-2 BLAS for matrix-vector operations in 1988 [109, 1988]. The level-2 BLAS are operations involving one matrix A and one or several vectors x and y, e.g., the real matrix-vector products

$$y := \alpha Ax + \beta y, \qquad y := \alpha A^T x + \beta y,$$

and

$$x := Tx, \qquad x := T^{-1}x, \qquad x := T^T x,$$

where α and β are scalars, x and y are vectors, A a matrix and T an upper or lower triangular matrix. The corresponding operations on complex data are also provided.

In most computers in use today the key to high efficiency is to avoid as much as possible data transfers between memory, registers and functional units, since these can be more costly than arithmetic operations on the data. This means that the operations have to be carefully structured. The LAPACK collection of subroutines [7, 1999] was initially released in 1992 to address these questions. LAPACK was designed to supersede and integrate the algorithms in both LINPACK and EISPACK. The subroutines are restructured to achieve much greater efficiency on modern high-performance computers. This is achieved by performing as much as possible of the computations by calls to so-called Level-2 and 3 BLAS. These enables the LAPACK routines to combine high performance with portable code and is also an aid to clarity, portability and modularity.

Level-2 BLAS involve $O(n^2)$ data, where n is the dimension of the matrix involved, and the same number of arithmetic operations. However, when RISC-type microprocessors with hierarchical memories were introduced, they failed to obtain adequate performance. Then level-3 BLAS were introduced in [108, 1990]. These were derived in a fairly obvious manner from some level-2 BLAS, by replacing the vectors x and y by matrices B and C,

$$C := \alpha AB + \beta C, \qquad C := \alpha A^T B + \beta C, \qquad C := \alpha AB^T + \beta C,$$

and

$$B := TB, \qquad B := T^{-1}B, \qquad B := T^T B,$$

Level-3 BLAS use $O(n^2)$ data but perform $O(n^3)$ arithmetic operations. Therefore, they give a surface-to-volume effect for the ratio of data movement to operations. This avoids excessive data movements between different parts of memory hierarchy. Level-3 BLAS are used in LAPACK, which achieves close to optimal performance on a large variety of computer architectures.

LAPACK is continually improved and updated and is available for free from `http://www.netlib.org/lapack95/`. Several special forms of matrices are supported by LAPACK:

General

General band
Positive definite
Positive definite packed
Positive definite band
Symmetric (Hermitian) indefinite
Symmetric (Hermitian) indefinite packed
Triangular
General tridiagonal
Positive definite tridiagonal

The LAPACK subroutines form the backbone of Cleve Moler's MATLABsystem, which has simplified matrix computations tremendously.

LAPACK95 is a Fortran 95 interface to the Fortran 77 LAPACK library. It is relevant for anyone who writes in the Fortran 95 language and needs reliable software for basic numerical linear algebra. It improves upon the original user-interface to the LAPACK package, taking advantage of the considerable simplifications that Fortran 95 allows. LAPACK95 Users' Guide provides an introduction to the design of the LAPACK95 package, a detailed description of its contents, reference manuals for the leading comments of the routines, and example programs.

7.5.2 Block and Blocked Algorithms

For linear algebra algorithms to achieve high performance on modern computer architectures they need to be rich in matrix-matrix multiplications. This has the effect of reducing data movement, since a matrix-matrix multiplication involves $O(n^2)$ data and does $O(n^3)$ flops. One way to achieve this is to to use block matrix algorithms.

In the following we make a distinction between two different classes of block algorithms,which have different stability properties. As a first example, consider the inverse of a block lower triangular matrix

$$L = \begin{pmatrix} L_{11} & & & \\ L_{21} & L_{22} & & \\ \vdots & \vdots & \ddots & \\ L_{n,1} & L_{n,2} & \cdots & L_{nn} \end{pmatrix}, \tag{7.5.1}$$

If the diagonal blocks L_{ii}, $i = 1 : 2$, are nonsingular, it is easily verified that the inverse also will be block lower triangular,

$$L^{-1} = \begin{pmatrix} Y_{11} & & & \\ Y_{21} & Y_{22} & & \\ \vdots & \vdots & \ddots & \\ Y_{n,1} & Y_{n,2} & \cdots & Y_{nn} \end{pmatrix}, \tag{7.5.2}$$

In Sec. 7.1.2 we showed that the inverse in the 2×2 case is

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}.$$

Note that we do not assume that the diagonal blocks are lower triangular.

In the general case the blocks in the inverse can be computed a block column at a time from a straightforward extension of the scalar algorithm (7.2.39). Identifying blocks in the jth block column, of the equation $LY = I$, we for $j = 1 : n$,

$$L_{jj}Y_{jj} = I, \qquad L_{ii}Y_{ij} = -\sum_{k=j}^{i-1} L_{ik}Y_{kj}, \quad i = j+1 : n. \tag{7.5.3}$$

These equations can be solved for $Y_{jj}, \dots, Y_{nj}$, by the scalar algorithms described in Sec. 7.2. The main arithmetic work will take place in the matrix-matrix multiplications $L_{ik}Y_{kj}$. This is an example of a true **block algorithm**, which is a obtained by substituting in a scalar algorithm operations on blocks of partitioned matrices regarded as non-commuting scalars.

In the special case that L is a lower triangular matrix this implies that all diagonal blocks L_{ii} and Y_{ii}, $i = 1 : n$, are lower triangular. In this case the equations in (7.5.3) can be solved by back-substitution. The resulting algorithm is then just a scalar algorithm in which the operations have been grouped and reordered into matrix operations. Such an algorithm is called a **blocked algorithm**. Blocked algorithms have the same stability properties as their scalar counterparts. This is not true for general block algorithms, which is why the distinction is important to make.

In Sec. 7.1.2 we gave, using slightly different notations, the block LU factorization

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & S \end{pmatrix}, \tag{7.5.4}$$

for a 2×2 block matrix, with square diagonal blocks. Here $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is the Schur complement. Note that the diagonal blocks in the block lower triangular factor in (7.5.4) are the identity matrix. Hence, , this is a true block algorithm.

In a blocked LU factorization algorithm, the LU factors should have the form

$$A = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix},$$

where L_{11}, L_{22} are unit lower triangular and U_{11} , U_{22} are upper triangular. Such a factorization can be computed as follows. We first compute the scalar LU factorization $A_{11} = L_{11}U_{11}$, and then compute

$$L_{21} = A_{21}U_{11}^{-1}, \qquad U_{12} = L_{11}^{-1}A_{12}, \qquad S_{22} = A_{22} - L_{21}U_{12}.$$

Finally, compute the scalar factorization $S_{22} = L_{22}U_{22}$.

In the general case a blocked algorithm for the LU factorization of a block matrix

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{pmatrix}, \tag{7.5.5}$$

with square diagonal blocks. Let L and U be partitioned conformally with A. Equating blocks in the product $A = LU$, we obtain, assuming that all inverses exist, the following block LU algorithm:

Algorithm 7.9. *Blocked LU Factorization.*

$$
\begin{array}{l}
\textbf{for } k = 1 : N \\
\quad S_{kk} = A_{kk} - \displaystyle\sum_{p=1}^{k-1} L_{kp} U_{pk}; \\
\quad S_{kk} = L_{kk} U_{kk} \\
\quad \textbf{for } j = k+1 : N \\
\qquad L_{jk} = \Big(A_{jk} - \displaystyle\sum_{p=1}^{k-1} L_{jp} U_{pk} \Big) U_{kk}^{-1}; \\
\quad \textbf{end} \\
\quad \textbf{for } j = 1 : k-1 \\
\qquad U_{jk} = L_{kk}^{-1} \Big(A_{jk} - \displaystyle\sum_{p=1}^{k-1} L_{jp} U_{pj} \Big); \\
\quad \textbf{end} \\
\textbf{end}
\end{array}
$$

Here the LU-decompositions $S_{kk} = L_{kk} U_{kK}$ of the modified diagonal blocks are computed by a scalar LU factorization algorithm. However, the dominating part of the work is performed in matrix-matrix multiplications. The inverse of the triangular matrices L_{kk}^{-1} and U_{kk}^{-1} are *not* formed but the off-diagonal blocks U_{kj} and L_{jk} (which in general are full matrices) are computed by triangular solves. Pivoting can be used in the factorization of the diagonal blocks. As described the algorithm does not allow for row interchanges between blocks. This point is addressed in the next section.

As with the scalar algorithms there are many possible ways of sequencing the block factorization. The block algorithm above computes in the kth major step the kth block column of L and U. In this variant at step k only the kth block column of A is accessed, which is advantageous from the standpoint of data access.

A block LU factorization algorithm differs from the blocked algorithm above in that the lower block triangular matrix L has diagonal blocks equal to unity. Although such a block algorithm may have good numerical stability properties this cannot be taken for granted, since in general they do not perform the same arithmetic operations as in the corresponding scalar algorithms. It has been shown that block LU factorization can fail even for symmetric positive definite and row diagonally dominant matrices.

One class of matrices for which the block LU algorithm is known to be stable is block tridiagonal matrices that are **block diagonally dominant**.

Definition 7.5.1 ((Demmel et al. [95, 1995])).

A general matrix $A \in \mathbf{R}^{n\times n}$ is said to be block diagonally dominant by columns, with respect to a given partitioning, if it holds i.e.

$$\|A_{jj}^{-1}\|^{-1} \geq \sum_{i\neq j} \|A_{ij}\|, \quad j = 1 : n. \tag{7.5.6}$$

It is said to be strictly block diagonally dominant if (7.5.6) holds with strict inequality.

A is said to be (strictly) block **diagonally dominant** *by rows, if A^T is (strictly) diagonally dominant by columns.*

Note that for block size 1 the usual property of (point) diagonal dominance is obtained. For the 1 and ∞-norms diagonal dominance does not imply block diagonal dominance. Neither does and the reverse implications hold.

Analogous to the Block LU Algorithm in Sec. 7.5.2 block versions of the Cholesky algorithm can be developed. If we assume that A has been partitioned into $N \times N$ blocks with square diagonal blocks we get using a block column-wise order:

Algorithm 7.10. *Blocked Cholesky Algorithm.*

$$
\begin{array}{l}
\textbf{for } j = 1 : N \\
\quad S_{jj} = A_{jj} - \displaystyle\sum_{k=1}^{j-1} R_{jk}^T R_{jk}; \\
\quad S_{jj} = R_{jj}^T R_{jj} \\
\quad \textbf{for } i = j+1 : N \\
\quad\quad R_{ij}^T = \Big(A_{ij} - \displaystyle\sum_{k=1}^{j-1} R_{ik}^T R_{jk}\Big)(R_{jj})^{-1}; \\
\quad \textbf{end} \\
\textbf{end}
\end{array}
$$

Note that the diagonal blocks R_{jj} are obtained by computing the Cholesky factorizations of matrices of smaller dimensions. The right multiplication with $(R_{jj})^{-1}$ in the computation of R_{jk}^T is performed by solving the triangular equations of the form $R_{jj}^T R_{ij} = S^T$. The matrix multiplications dominate the arithmetic work in the block Cholesky algorithm.

In deriving the block LU and Cholesky algorithms we assumed that the block sizes were determined in advance. However, this is by no means necessary. A more flexible way is to advance the computation by deciding at each step the size of the current pivot block. The corresponding blocked formulation then uses a 3×3 block structure, but the partitioning changes after each step.

Suppose that an LU factorization of the first n_1 columns has been computed. We can write the result in the form

$$P_1 A = \begin{pmatrix} L_{11} & & \\ L_{21} & I & \\ L_{31} & 0 & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & \tilde{A}_{22} & \tilde{A}_{23} \\ & \tilde{A}_{32} & \tilde{A}_{33} \end{pmatrix}. \tag{7.5.7}$$

where P_1 is a permutation matrix and L_{11}, $U_{11} \in \mathbf{R}^{n_1 \times n_1}$ has been obtained. The remaining $n-n_1$ columns are partitioned into blocks of n_2 and $n-(n_1+n_2)$ columns.

To advance the factorization an LU factorization with row pivoting is performed

$$P_2 \begin{pmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{pmatrix} = \begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} U_{22}, \tag{7.5.8}$$

where L_{22}, $U_{22} \in \mathbf{R}^{n_2 \times n_2}$. The permutation matrix P_2 has to be applied also to

$$\begin{pmatrix} \tilde{A}_{23} \\ \tilde{A}_{33} \end{pmatrix} := P_2 \begin{pmatrix} \tilde{A}_{23} \\ \tilde{A}_{33} \end{pmatrix}, \qquad \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} := P_2 \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix}.$$

We then solve for U_{23} and update A_{33} using

$$L_{22}U_{23} = \tilde{A}_{23}, \qquad \tilde{A}_{33} = A_{33} - L_{32}U_{23}.$$

The factorization has now been advanced one step to become

$$P_2P_1A = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & A_{33} \end{pmatrix}.$$

We can now repartition so that the first two block-columns in L are joined into a block of n_1+n_2 columns and similarly the first two block-rows in U joined into one block of n_1+n_2 rows. The blocks I and A_{33} in L and U are partitioned into 2×2 block matrices and we advance to the next block-step. This describes the complete algorithm since we can start the algorithm by taking $n_1 = 0$.

The above algorithm is sometimes called **right-looking**, referring to the way in which the data is accessed. The corresponding **left-looking** algorithm goes as follows. Assume that we have computed the first block column in L and U in the factorization (7.5.7). To advance the factorization we solve the triangular system $L_{11}U_{12} = A_{12}$ to obtain U_{12} and compute

$$\begin{pmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{pmatrix} = \begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix} - \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} U_{12},$$

We then compute the partial LU factorization (7.5.8) and replace

$$\begin{pmatrix} \tilde{A}_{23} \\ \tilde{A}_{33} \end{pmatrix} = P_2 \begin{pmatrix} A_{23} \\ A_{33} \end{pmatrix}, \qquad \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} := P_2 \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix}.$$

The factorization has now been advanced one step to become

$$P_2P_1A = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & A_{13} \\ & U_{22} & A_{23} \\ & & A_{33} \end{pmatrix}.$$

Note that in this version the blocks in the last block column of A are referenced only in the pivoting operation, but this can be postponed.

Block LU factorizations appears to have been first proposed for **block tridiagonal** matrices, which often arise from the discretization of partial differential equations. For a symmetric positive definite matrix the recursion (7.4.11) is easily generalized to compute the following block-factorization:

$$A = U^T D^{-1} U, \quad D = \mathrm{diag}\,(\Sigma_1, \ldots, \Sigma_n),$$

of a symmetric positive definite **block-tridiagonal** matrix with square diagonal blocks. We obtain

$$A = \begin{pmatrix} D_1 & A_2^T & & & \\ A_2 & D_2 & A_3^T & & \\ & A_3 & \ddots & \ddots & \\ & & \ddots & \ddots & A_N^T \\ & & & A_N & D_N \end{pmatrix}, \qquad U^T = \begin{pmatrix} \Sigma_1 & & & & \\ A_2 & \Sigma_2 & & & \\ & A_3 & \ddots & & \\ & & \ddots & \ddots & \\ & & & A_N & \Sigma_N \end{pmatrix},$$

where

$$\Sigma_1 = D_1, \qquad \Sigma_k = D_k - A_k \Sigma_{k-1}^{-1} A_k^T, \quad k = 2 : N. \tag{7.5.9}$$

To perform the operations with Σ_k^{-1}, $k = 1 : N$ the Cholesky factorization of these matrices are computed by a scalar algorithm. After this factorization has been computed the solution of the system

$$Ax = U^T D^{-1} U x = b$$

can be obtained by block forward- and back-substitution $U^T z = b$, $Ux = Dz$.

Note that the blocks of the matrix A may again have band-structure, which should be taken advantage of! A similar algorithm can be developed for the unsymmetric block-tridiagonal case.

For block tridiagonal matrices the following result is known:

Theorem 7.5.2. *(*Varah [374, 1972]*)*
Let the matrix $A \in \mathbf{R}^{n\times n}$ be block tridiagonal and have the block LU factorization $A = LU$, where L and U are block bidiagonal, and normalized so that $U_{i,i+1} = A_{i,i+1}$. Then if A is block diagonally dominant by columns

$$\|L_{i,i-1}\| \leq 1, \qquad \|U_{i,i}\| \leq \|A_{i,i}\| + \|A_{i-1,i}\|. \tag{7.5.10}$$

If A is block diagonally dominant by rows

$$\|L_{i,i-1}\| \leq \frac{A_{i-1,i}\|}{\|A_{i,i-1}\|}, \qquad \|U_{i,i}\| \leq \|A_{i,i}\| + \|A_{i-1,i}\|. \tag{7.5.11}$$

These results can be extended to full block diagonally dominant matrices, by using the key property that block diagonal dominance is inherited by the Schur complements obtained in the factorizations.

7.5.3 Recursive Fast Matrix Multiply

Recursive algorithms introduces an automatic blocking, where the block size changes during the execution and can targets several different levels of memory hierarchy; Gustavson [193, 1997].

As a first example of a recursive matrix algorithm we consider the algorithm for fast matrix multiplication of Strassen [351, 1969]. This is based on an algorithm for multiplying 2×2 block matrices. Let A and B be matrices of dimensions $m \times n$ and $n \times p$, respectively, where all dimensions are even. Partition A, B, and the product $C = AB$ into four equally sized blocks

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

Then, as can be verified by substitution, the product C can be computed using the following formulas:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 + P_3 - P_2 + P_6 \end{pmatrix}, \tag{7.5.12}$$

where

$$\begin{aligned} &P_1 = (A_{11} + A_{22})(B_{11} + B_{22}), \qquad P_2 = (A_{21} + A_{22})B_{11}, \\ &P_3 = A_{11}(B_{12} - B_{22}), \qquad P_4 = A_{22}(B_{21} - B_{11}), \\ &P_5 = (A_{11} + A_{12})B_{22}, \qquad P_6 = (A_{21} - A_{11})(B_{11} + B_{12}), \\ &P_7 = (A_{12} - A_{22})(B_{21} + B_{22}). \end{aligned}$$

The key property of **Strassen's algorithm** is that only *seven* matrix multiplications and eighteen matrix additions are needed, instead of the *eight* matrix multiplications and four matrix additions required using conventional block matrix multiplications. Since for large dimensions multiplication of two matrices is much more expensive (n^3) than addition (n^2) this will lead to a saving in operations.

If Strassen's algorithm is used recursively to multiply two square matrices of dimension $n = 2^k$, then the number of multiplications is reduced from n^3 to $n^{\log_2 7} = n^{2.807\ldots}$. (The number of additions is of the same order.) Even with just one level of recursion Strassen's method is faster in practice when n is larger than about 100, see Problem 5. However, there is some loss of numerical stability compared to conventional matrix multiplication, see Higham [211, Ch. 23]. By using the block formulation recursively, and Strassen's method for the matrix multiplication it is possible to perform the LU factorization In practice recursion is only performed down to some level at which the gain in arithmetic operations is outweighed by overheads in the implementation.

The following MATLAB program shows how Strassen's algorithm can be implemented in a simple but efficient recursive MATLAB program. The program uses the fast matrix multiplication as long as n is a power of two and $n > nmin$ when it switches to standard matrix multiplication.

Algorithm 7.11. *Recursive Matrix Multiplication by Strassen's Algorithm.*

```
function C = smult(A,B,nmin);
% SMULT computes the matrix product A*B of two square
% matrices using Strassen's algorithm
n = size(A,1);
if rem(n,2) == 0 & n > nmin
  Recursive multiplication
   n = n/2; u = 1:n; v = n+1:2*n;
   P1 = smult(A(u,u) + A(v,v),B(u,u) + B(v,v),nmin);
   P2 = smult(A(v,u) + A(v,v),B(u,u),nmin);
   P3 = smult(A(u,u),B(u,v) - B(v,v),nmin);
   P4 = smult(A(v,v),B(v,u) - B(u,u),nmin);
   P5 = smult(A(u,u) + A(u,v),B(v,v),nmin);
   P6 = smult(A(v,u) - A(u,u),B(u,u) + B(u,v),nmin);
   P7 = smult(A(u,v) - A(v,v),B(v,u) + B(v,v),nmin);
   C(u,u) = P1 + P4 - P5 + P7;
   C(u,v) = P3 + P5;
   C(v,u) = P2 + P4;
   C(v,v) = P1 + P3 - P2 + P6;
else
   C = A*B;
end
```

For $nmin = 1$ the recursion produces a complete binary tree of depth $k+1$, where

$$2^{k-1} < n \leq 2^k.$$

This tree is transversed in pre-order during the execution; see Knuth [241, Sec. 2.3]. Figure 7.5.1 shows the tree and the order in which the nodes are visited for $n = 8$.

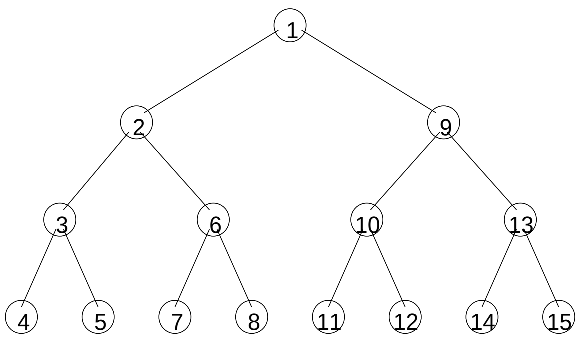

Figure 7.5.1. *Binary tree with nodes in pre-order.*

7.5.4 Recursive Blocked Matrix Factorizations

From 2005 almost all computer manufacturers have changed their computer architecture to multi-core. The number of processors is expected to double about every 15 month. These new designs lead to poor performance for traditional block methods in matrix computations. For this new architectures recursively blocked matrix algorithms and data structures have several advantages.

By using recursive blocking in matrix factorizations we obtain algorithms which express the computations entirely in level-3 BLAS matrix-matrix operations. In this section we exemplify this by looking at the Cholesky and LU factorizations. The idea can applies more generally to other matrix algorithms.

We now develop a recursive algorithm for the Cholesky factorization. of a symmetric positive definite matrix $A \in \mathbf{R}^{n\times n}$. Then no pivoting need to be performed. We assume that A is stored in the upper triangular part of a square matrix. The matrix is partitioned into a 2×2 block matrix with square diagonal blocks A_{11} and A_{22} of order n_1 and $n_2 = n - n_1$, respectively. Equating blocks in the matrix equation

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{12}^T & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{12} \\ 0 & L_{22}^T \end{pmatrix}. \tag{7.5.13}$$

gives the following matrix equations for computing the three nonzero blocks in the Cholesky factor L:

$$\begin{aligned} &L_{11}L_{11}^T = A_{11}, \\ &L_{21}^T = L_{11}^{-1}A_{12}, \\ &\tilde{A}_{22} = A_{22} - L_{12}^T L_{12}, \\ &L_{22}L_{22}^T = \tilde{A}_{22}. \end{aligned}$$

We recall that the submatrices A_{11} and $\tilde{A}_{22}$ are also positive definite. Here L_{11} is the Cholesky factorization of a matrix of size $n_1 \times n_1$. The block L_{21} is obtained by solving an upper triangular matrix equation. Next the block A_{22} is modified by the symmetric matrix $L_{21}L_{21}^T$. Finally, the Cholesky factorization of this modified block of size $n_2 \times n_2$ is computed.

If n is even and $n_1 = n_2 = n/2$, then the two Cholesky factorizations are of size $n/2 \times n/2$ requires $n^3/12$ flops, which is $1/4$ of the total number of $n^3/3$ flops. The triangular solve and modification step each take $n/8$ flops. (Note that only the upper the upper triangular part of $\tilde{A}_{22}$ needs to be computed.)

Using these equations recursively a recursive algorithm for Cholesky factorization is obtained. The algorithm below does not take advantage of the symmetry in the matrices, r.g. in the modification of the $(2, 2)$ block.

Algorithm 7.12. *Recursive Cholesky Factorization.*

Let $A \in \mathbf{R}^{n\times n}$ be a symmetric positive definite matrix. The following recursive algorithm computes the Cholesky factorization of A.

```
function L = rchol(A);
%RCHOL Recursive Cholesky Factorization
```

```
 [n,n] = size(A);
 if n > 1
   n1 = floor(n/2); n2 = n-n1;
   j1 = 1:n1; j2 = n1+1:n;
   L11 = rchol(A(j1,j1));          %recursive call
   L12 = L11\A(j1,j2);             %triangular solve
   A(j2,j2) = A(j2,j2) - L12'*L12; %modify %%(2,2) block
   L22 = rchol(A(j2,j2));          %recursive call
   L = [L11, zeros(n1,n2); L12', L22];
else
   L = sqrt(A);
end
```

Note that in the recursive algorithm *all the work is done in the triangular solves and matrix multiplication*. At each level i, 2 calls to level-3 BLAS are made. In going from level i to $i+1$, the number of BLAS calls doubles and each problem size is halved. Hence, the number of flops done at each level goes down in a geometric progression by a factor of 4. Since the total number of flops must remain the same, this means that a large part of the calculations are made at low levels. But since the MFLOP rate goes down with the problem size the computation time does not quite go down as 1/4. For large problems this does not affect the total efficiency. But for small problems, where most of the calls to level-3 BLAS has small problem size, the efficiency is deteriorated. This can be avoided by calling a standard Cholesky routine if the problem size satisfies $n > nmin$. A recursive algorithm for Cholesky factorization of a matrix in packed storage format is described in [5, 1981]. This is not a toy algorithm, but can be developed into efficient algorithms for parallel high performance computers!

A recursive algorithm for the LU factorization of a matrix A can be obtained similarly. In order to accommodate partial pivoting we need to consider the LU factorization of a rectangular matrix $A \in \mathbf{R}^{m\times n}$ into a product of $L \in \mathbf{R}^{m\times n}$ and $U \in \mathbf{R}^{n\times n}$. The total number of flops required for this factorization is $n^2 m - n^3/3$.

We partition the matrix as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}, \tag{7.5.14}$$

where $A_{11} \in \mathbf{R}^{n_1\times n_1}$ and $A_{22} \in \mathbf{R}^{n_2\times(m-n_1)}$. Then the size of the blocks in the factorization are

$$L_{11}, U_{11} \in \mathbf{R}^{n_1\times n_1}, \qquad U_{22} \in \mathbf{R}^{n_2\times n_2}, \qquad L_{22} \in \mathbf{R}^{(m-n_1)\times n_1}.$$

Equating blocks on the left and right hand side we get

$$\begin{aligned}
&\begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} U_{11} = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}, && n^2 m/4 - n^3/24 \text{ flops} \\
&U_{12} = L_{11}^{-1} A_{12}, && n^3/8 \text{ flops} \\
&\tilde{A}_{22} = A_{22} - L_{21} U_{12}, && n^2 m/2 - n^3/4 \text{ flops} \\
&L_{22} U_{22} = \tilde{A}_{22}. && n^2 m/4 - n^3/6 \text{ flops}
\end{aligned} \tag{7.5.15}$$

The flop counts above are for the case that n is even and $n_1 = n_2 = n/2$. Hence, the LU factorization of $A \in \mathbf{R}^{m\times n}$ is reduced to an LU factorization of the first block of n_1 columns of A which is of size $m \times n_1$. Next U_{12} is computed by a triangular solve and a modification of the block A_{22} is performed. Finally, an LU factorization of the modified block of size $(m - n_1) \times n_2$ is computed by a recursive call. The triangular solve and matrix modification are both performed by level-3 BLAS.

For a recursive algorithm for LU factorization with partial pivoting the same approach as in (7.5.15) can be used. We then perform the two LU factorizations with partial pivoting; (see Gustavson [193, 1997] and Toledo [356, 1997]). As before the recursion will produce a binary tree with of depth $k + 1$ where $2^{k-1} < n \le 2^k$. At each level i, 2 calls to level-3 BLAS are made.

Algorithm 7.13. *Recursive LU Factorization.*

The following recursive algorithm computes the LU factorization of the matrix $A \in \mathbf{R}^{m\times n}$, $m \ge n$.

```
function [LU,p] = rlu(A);
% Recursive LU factorization with partial pivoting
% of A. p holds the final row ordering
[m,n] = size(A);
if n > 1
   n1 = floor(n/2); n2 = n - n1;
   j1 = 1:n1; j2 = n1+1:n;
   [L1,U1,p] = rlu(A(:,j1));            % recursive call
   A(:,j2) = A(p,j2);                   % forward pivot
   U12 = L1(j1,:)\A(j1,j2);             % triangular solve
   i2 = n1+1:m;
   A(i2,j2) = A(i2,j2) - L1(i2,:)*U12;  % modify (2,2) block
   U1 = [U1, U12];
   [L2,U2,p2] = rlu(A(i2,j2));          % recursive call
   p2 = n1 + p2;                        % modify permutation
   L1(i2,:) = L1(p2,:);                 % back pivot
   L2 = [zeros(n1,n2); L2];
   U2 = [zeros(n2,n1), U2];
   L = [L1, L2];  U = [U1; U2];
   p2 = [j1,p2]; p = p(p2);
else
   p = 1:m;                            % initialize permutation
   [piv,k] = max(abs(A(:,1)));         % find pivot element
   if k > 1
      A([1,k],1) = A([k,1],1);         % swap rows 1 and k
      p([1,k]) = p([k,1]);
   end
   U = A(1,1); L = A(:,1)/A(1,1);
end
```

In going from level i to $i+1$, the number of BLAS calls doubles. The problem size in the recursive call are now $m \times n/2$ and $n/2 \times n/2$. From the flop count above it follows that the total number of flops done at each level now goes down by more than a factor of two.

7.5.5 Kronecker Systems

Linear systems where the matrix is a **Kronecker product**[20] arise in several application areas such as signal and image processing, photogrammetry, multidimensional data fitting, etc. Such systems can be solved with great savings in storage and operations. Since often the size of the matrices A and B is large, resulting in models involving several hundred thousand equations and unknowns, such savings may be essential.

Definition 7.5.3.
Let $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{p\times q}$ be two matrices. Then the **Kronecker product** *of A and B is the $mp \times nq$ block matrix*

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}. \tag{7.5.16}$$

We now state without proofs some elementary facts about Kronecker products. From the definition (7.5.16) it follows that

$$\begin{aligned} (A+B)\otimes C &= (A\otimes C) + (B\otimes C), \\ A\otimes(B+C) &= (A\otimes B) + (A\otimes C), \\ A\otimes(B\otimes C) &= (A\otimes B)\otimes C, \\ (A\otimes B)^T &= A^T \otimes B^T. \end{aligned}$$

Further we have the important mixed-product relation, which is not so obvious:

Lemma 7.5.4.
Let $A \in \mathbf{R}^{m\times n}$, $B \in \mathbf{R}^{p\times q}$, $C \in \mathbf{R}^{n\times k}$, and $D \in \mathbf{R}^{q\times r}$. Then the ordinary matrix products AC and BD are defined, and

$$(A\otimes B)(C\otimes D) = AC \otimes BD. \tag{7.5.17}$$

Proof. Let $A = (a_{ik})$ and $C = (c_{kj})$. Partitioning according to the sizes of B and D, $A\otimes B = (a_{ik}B)$ and $C\otimes D = (c_{kj}D)$. Hence, , the (i,j)th block of $(A\otimes B)(C\otimes D)$

[20] Leopold Kronecker (1823–1891) German mathematician. He is known also for his remark "God created the integers, all else is the work of man".

equals

$$\sum_{k=1}^{n} a_{ik} B c_{kj} D = \left(\sum_{k=1}^{n} a_{ik} c_{kj} \right) BD,$$

which is the (i,j)th element of AC times BD, which is the (i,j)th block of $(A \otimes B)(C \otimes D)$. $\square$

If $A \in \mathbf{R}^{n\times n}$ and $B \in \mathbf{R}^{p\times p}$ are nonsingular, then then by Lemma 7.5.4

$$(A^{-1} \otimes B^{-1})(A \otimes B) = I_n \otimes I_p = I_{np}.$$

It follows that $A \otimes B$ is nonsingular and

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \tag{7.5.18}$$

We now introduce an operator closely related to the Kronecker product, which converts a matrix into a vector.

Definition 7.5.5. *Given a matrix $C = (c_1, c_2, \dots, c_n) \in \mathbf{R}^{m\times n}$ we define*

$$\mathrm{vec}\,(C) = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \tag{7.5.19}$$

that is, the vector formed by **stacking** *the columns of C into one long vector.*

We now state an important result which shows how the vec-function is related to the Kronecker product.

Lemma 7.5.6.
If $A \in \mathbf{R}^{m\times n}$, $B \in \mathbf{R}^{p\times q}$, and $C \in \mathbf{R}^{q\times n}$, then

$$(A \otimes B)\mathrm{vec}\, C = \mathrm{vec}\, X, \quad X = BCA^T. \tag{7.5.20}$$

Proof. Denote the kth column of a matrix M by M_k. Then

$$\begin{aligned} (BCA^T)_k = BC(A^T)_k &= B \sum_{i=1}^{n} a_{ki} C_i \\ &= (\, a_{k1}B \quad a_{k2}B \quad \dots a_{kn}B\,)\, \mathrm{vec}\, C, \end{aligned}$$

where $A = (a_{ij})$. But this means that $\mathrm{vec}\,(BCA^T) = (A \otimes B)\mathrm{vec}\, C$. $\square$

Let $A \in \mathbf{R}^{n\times n}$ and $B \in \mathbf{R}^{p\times p}$ be nonsingular, and $C \in \mathbf{R}^{p\times n}$. Consider the Kronecker linear system

$$(A \otimes B)x = \mathrm{vec}\, C, \tag{7.5.21}$$

which is of order np. Then by (7.5.18) the solution can be written

$$x = (A^{-1} \otimes B^{-1})\mathrm{vec}\, C = \mathrm{vec}\,(X), \quad X = B^{-1} C A^{-T}. \tag{7.5.22}$$

where C is the matrix such that $c = \mathrm{vec}\,(C)$. This reduces the operation count for solving (7.5.21) from $O(n^3p^3)$ to $O(n^3 + p^3)$.

Review Questions

5.1 How many operations are needed (approximately) for
(a) The LU factorization of a square matrix?
(b) The solution of $Ax = b$, when the triangular factorization of A is known?

5.2 To compute the matrix product $C = AB \in \mathbf{R}^{m\times p}$ we can either use an outer product or an inner product formulation. Discuss the merits of the two resulting algorithms when A and B have relatively few nonzero elements.

5.3 Is the Hadamard product $A \mathbin{.*} B$ a submatrix of the Kronecker product $A \otimes B$?

Problems

5.1 Assume that for the nonsingular matrix $A_{n-1} \in \mathbf{R}^{(n-1)\times(n-1)}$ we know the LU factorization $A_{n-1} = L_{n-1}U_{n-1}$. Determine the LU factorization of the **bordered matrix** $A_n \in \mathbf{R}^{n\times n}$,

$$A_n = \begin{pmatrix} A_{n-1} & b \\ c^T & a_{nn} \end{pmatrix} = \begin{pmatrix} L_{n-1} & 0 \\ l^T & 1 \end{pmatrix} \begin{pmatrix} U_{n-1} & u \\ 0 & u_{nn} \end{pmatrix}.$$

Here $b, c \in \mathbf{R}^{n-1}$ and a_{nn} are given and $l, u \in \mathbf{R}^{n-1}$ and u_{nn} are to be determined.

5.2 The methods of forward- and back-substitution extend to block triangular systems. Show that the 2×2 block upper triangular system

$$\begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

can be solved by block back-substitution provided that the diagonal blocks U_{11} and U_{22} are square and nonsingular.

5.3 Write a recursive LU Factorization algorithm based on the 2×2 block LU algorithm.

5.4 Show the equality

$$\mathrm{vec}\,(A)^T \mathrm{vec}\,(B) = \mathrm{trace}\,(A^T B). \tag{7.5.23}$$

5.5 (a) Let $A \in \mathbf{R}^{m\times n}$, $B \in \mathbf{R}^{n\times p}$, with m and n even. Show that, whereas conventional matrix multiplication requires mnp multiplications (M) and $m(n-1)p$ additions (A) to form the product $C = AB \in \mathbf{R}^{m\times p}$, Strassen's algorithm, using conventional matrix multiplication at the block level, requires

$$\frac{7}{8}mnp \text{ M } + \frac{7}{8}m(n-2)p + \frac{5}{4}n(m+p) + 2mp \text{ A}.$$

(b) Show, using the result in (a), that if we assume that "M ≈ A", Strassen's algorithm is cheaper than conventional multiplication when $mnp \le 5(mn + np + mp)$.

7.6 Perturbation Theory and Condition Estimation

7.6.1 Numerical Rank of Matrix

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

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

Clearly the **numerical rank** assigned to a matrix should depend on some tolerance δ, which reflects the error level in the data and/or the precision of the floating point arithmetic used. A useful definition is the following:

Definition 7.6.1.
A matrix $A \in \mathbf{R}^{m\times n}$ has numerical δ-rank equal to k ($k \le \min\{m,n\}$) if

$$\sigma_1 \ge \ldots \ge \sigma_k > \delta \ge \sigma_{k+1} \ge \ldots \ge \sigma_n,$$

where σ_i, $i = 1:n$ are the singular values of A. If we write

$$A = U\Sigma V^T = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T,$$

where $\Sigma_2 = \operatorname{diag}(\sigma_{k+1},\ldots,\sigma_n)$ then $\mathcal{R}(V_2) = \operatorname{span}\{v_{k+1},\ldots,v_n\}$ is called the **numerical null space** *of A.*

It follows from Theorem 8.1.14, that if the numerical δ-rank of A equals k, then $\operatorname{rank}(A+E) \ge k$ for all perturbations such that $\|E\|_2 \le \delta$, i.e., such perturbations cannot *lower* the rank. Definition 7.6.1 is only useful when there is a well defined gap between σ_{k+1} and σ_k. This should be the case if the exact matrix A is rank deficient but well-conditioned. However, it may occur that there does not exist a gap for any k, e.g., if $\sigma_k = 1/k$. In such a case the numerical rank of A is not well defined!

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

7.6.2 Conditioning of Linear Systems

Consider a linear system $Ax = b$ where A is nonsingular and $b \neq 0$. The sensitivity of the solution x and the inverse A^{-1} to perturbations in A and b is of practical importance, since the matrix A and vector b are rarely known exactly. They may be subject to observational errors, or given by formulas which involve roundoff errors in their evaluation. (Even if they were known exactly, they may not be represented exactly as floating-point numbers in the computer.)

We start with deriving some results that are needed in the analysis.

Lemma 7.6.2.

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

$$\|(I - E)^{-1}\| \leq 1/(1 - \|E\|). \tag{7.6.1}$$

Proof. If $(I - E)$ is singular there exists a vector $x \neq 0$ such that $(I - E)x = 0$. Then $x = Ex$ and $\|x\| = \|Ex\| \leq \|E\| \, \|x\| < \|x\|$, which is a contradiction since $\|x\| \neq 0$. Hence, $(I - E)$ is nonsingular.

Next consider the identity $(I - E)(I - E)^{-1} = I$ or

$$(I - E)^{-1} = I + E(I - E)^{-1}.$$

Taking norms we get

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

and (7.6.1) follows. (For another proof, see hint in Problem 7.2.19.) $\square$

Corollary 7.6.3.

Assume that $\|B - A\|\|B^{-1}\| = \eta < 1$. Then it holds that

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

Proof. We have $\|A^{-1}\| = \|A^{-1}BB^{-1}\| \leq \|A^{-1}B\| \, \|B^{-1}\|$. The first inequality then follows by taking $E = B^{-1}(B - A) = I - B^{-1}A$ in Lemma 7.6.2. From the identity

$$A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1} \tag{7.6.2}$$

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

Let x be the solution x to a system of linear equations $Ax = b$, and let $x + \delta x$ satisfy the perturbed system

$$(A + \delta A)(x + \delta x) = b + \delta b,$$

where δA and δb are perturbations in A and b. Subtracting out $Ax = b$ we get

$$(A + \delta A)\delta x = -\delta A x + \delta b.$$

Assuming that A and $A + \delta A$ are nonsingular, we can multiply by A^{-1} and solve for δx. This yields

$$\delta x = (I + A^{-1}\delta A)^{-1} A^{-1}(-\delta A x + \delta b), \tag{7.6.3}$$

which is the basic identity for the perturbation analysis.

In the simple case that $\delta A = 0$, we have $\delta x = A^{-1}\delta b$, which implies that $|\delta x| = |A^{-1}|\,|\delta b|$, Taking norms

$$\|\delta x\| \leq \|A^{-1}\|\,\|\delta b\|.$$

Usually it is more appropriate to consider *relative* perturbations,

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A, x)\frac{\|\delta b\|}{\|b\|}, \quad \kappa(A, x) := \frac{\|Ax\|}{\|x\|}\|A^{-1}\|. \tag{7.6.4}$$

Here $\kappa(A, x)$ is the condition number with respect to perturbations in b. It is important to note that this implies that the size of the residual vector $r = b - A\bar{x}$ gives no direct indication of the *error* in an approximate solution $\bar{x}$. For this we need information about A^{-1} or the condition number $\kappa(A, x)$.

The inequality (7.6.4) is sharp in the sense that for any matrix norm and for any A and b there exists a perturbation δb, such that equality holds. From $\|b\| = \|Ax\| \leq \|A\|\,\|x\|$ it follows that

$$\kappa(A, x) \leq \|A\|\,\|A^{-1}\|, \tag{7.6.5}$$

but here *equality will hold only for rather special right hand sides* b. Equation (7.6.5) motivates the following definition:

Definition 7.6.4. *For a square nonsingular matrix A the* **condition number** *is*

$$\kappa = \kappa(A) = \|A\|\,\|A^{-1}\|. \tag{7.6.6}$$

where $\|\cdot\|$ denotes any matrix norm.

Clearly $\kappa(A)$ depends on the chosen matrix norm. If we want to indicate that a particular norm is used, then we write, e.g., $\kappa_\infty(A)$ etc. For the 2-norm we have using the SVD that $\|A\|_2 = \sigma_1$ and $\|A^{-1}\| = 1/\sigma_n$, where σ_1 and σ_n are the largest and smallest singular values of A. Hence,

$$\kappa_2(A) = \sigma_1/\sigma_n. \tag{7.6.7}$$

Note that $\kappa(\alpha A) = \kappa(A)$, i.e., the condition number is invariant under multiplication of A by a scalar. From the definition it also follows easily that

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

Further, for all p-norms it follows from the identity $AA^{-1} = I$ that

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

that is, the condition number is always greater or equal to one.

We now show that $\kappa(A)$ is also the condition number with respect to perturbations in A.

Theorem 7.6.5.

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

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

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

$$\|\delta x\| \leq \frac{\|A^{-1}\|}{1-\eta} \left(\|\delta A\| \, \|x\| + \|\delta b\|\right). \tag{7.6.8}$$

Proof. Taking norms in equation (7.6.3) gives

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

By assumption $\|A^{-1}\delta A\| \leq \|A^{-1}\| \, \|\delta A\| = \eta < 1$. Using Lemma 7.6.2 it follows that $(I + A^{-1}\delta A)$ is nonsingular and

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

which proves the result. □

In most practical situations it holds that $\eta \ll 1$ and therefore $1/(1-\eta] \approx 1$. Therefore, if upper bounds

$$\|\delta A\| \leq \epsilon_A \|A\|, \qquad \|\delta b\| \leq \epsilon_b \|b\|, \tag{7.6.9}$$

for $\|\delta A\|$ and $\|\delta b\|$ are known, then for the normwise relative perturbation it holds that

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

Substituting $b = I$, $\delta b = 0$ and $x = A^{-1}$ in (7.6.8) and proceeding similarly from $(A + \delta A)(X + \delta X) = I$, we obtain the perturbation bound for $X = A^{-1}$

$$\frac{\|\delta X\|}{\|X\|} \leq \frac{\kappa(A)}{1-\eta} \frac{\|\delta A\|}{\|A\|}. \tag{7.6.10}$$

This shows that $\kappa(A)$ is indeed the condition number of A with respect to inversion.

The relative distance of a matrix A to the set of singular matrices in some norm is defined as

$$\operatorname{dist}(A) := \min \left\{ \frac{\|\delta A\|}{\|A\|} \mid (A + \delta A) \text{ singular} \right\}. \tag{7.6.11}$$

The following theorem shows that the reciprocal of the condition number $\kappa(A)$ can be interpreted as a measure of the nearness to singularity of a matrix A.

Theorem 7.6.6 (Kahan [230]).

Let $A \in \mathbf{C}^{n\times n}$ be a nonsingular matrix and $\kappa(A) = \|A\|\|A^{-1}\|$ the condition number with respect to a norm $\|\cdot\|$ subordinate to some vector norm. Then

$$\mathrm{dist}\,(A) = \kappa^{-1}(A). \tag{7.6.12}$$

Proof. If $(A+\delta A)$ is singular, then there is a vector $x \neq 0$ such that $(A+\delta A)x = 0$. Then, setting $y = Ax$, it follows that

$$\|\delta A\| \geq \frac{\|\delta A\,x\|}{\|x\|} = \frac{\|Ax\|}{\|x\|} = \frac{\|y\|}{\|A^{-1}y\|} \geq \frac{1}{\|A^{-1}\|} = \frac{\|A\|}{\kappa(A)},$$

or $\|\delta A\|/\|A\| \geq 1/\kappa(A)$.

Now let x be a vector with $\|x\| = 1$ such that $\|A^{-1}x\| = \|A^{-1}\|$. Set $y = A^{-1}x/\|A^{-1}\|$ so that $\|y\| = 1$ and $Ay = x/\|A^{-1}\|$. Let z be a dual vector to y so that (see Definition 7.1.16) $\|z\|_D\|y\| = z^Hy = 1$, where $\|\cdot\|_D$ is the dual norm. Then $\|z\|_D = 1$, and if we take

$$\delta A = -xz^H/\|A^{-1}\|,$$

it follows that

$$(A+\delta A)y = Ay - xz^Hy/\|A^{-1}\| = (x-x)/\|A^{-1}\| = 0.$$

Hence, $(A+\delta A)$ is singular. Further

$$\|\delta A\|\|A^{-1}\| = \|xz^H\| = \max_{\|v\|=1}\|(xz^H)v\| = \|x\|\max_{\|v\|=1}|z^Hv| = \|z\|_D = 1,$$

and thus $\|\delta A\| = 1/\|A^{-1}\|$, which proves the theorem. $\square$

The result in Theorem 7.6.6 can be used to get a *lower bound* for the condition number $\kappa(A)$, see, Problem 21. For the 2-norm the result follows directly from the SVD $A = U\Sigma V^H$. The closest singular matrix then equals $A+\delta A$, where

$$\delta A = -\sigma_n u_n v_n^H, \quad \|\delta A\|_2 = \sigma_n = 1/\|A^{-1}\|_2. \tag{7.6.13}$$

Matrices with small condition numbers are said to be **well-conditioned**. For any real, orthogonal matrix Q it holds that $\kappa_2(Q) = \|Q\|_2\|Q^{-1}\|_2 = 1$, so Q is perfectly conditioned in the 2-norm. Furthermore, for any orthogonal P and Q, we have $\kappa_2(PAQ) = \kappa_2(A)$, i.e., $\kappa_2(A)$ is invariant under orthogonal transformations.

When a linear system is ill-conditioned, i.e. $\kappa(A) \gg 1$, roundoff errors will in general cause a computed solution to have a large error. It is often possible to show that a small **backward error** in the following sense:

Definition 7.6.7.

An algorithm for solving a linear system $Ax = b$ is said to be (normwise) **backward stable** *if, for any data $A \in \mathbf{R}^{n\times n}$ and $b \in \mathbf{R}^n$, there exist perturbation* *matrices and vectors δA and δb, such that the solution $\bar{x}$ computed by the algorithm is the* exact *solution to a neighbouring system*

$$(A + \delta A)\bar{x} = (b + \delta b), \tag{7.6.14}$$

where

$$\|\delta A\| \le c_1(n)u\|A\|, \qquad \|\delta b\| \le c_2(n)u\|b\|.$$

A computed solution $\bar{x}$ is called a (normwise) stable solution if it satisfies (7.6.14).

Since the data A and b usually are subject to errors and not exact, it is reasonable to be satisfied with the computed solution $\bar{x}$ if the backward errors δA and δb are small in comparison to the uncertainties in A and b. As seen from (7.6.5), this does not mean that $\bar{x}$ is close to the exact solution x.

7.6.3 Component-Wise Perturbation Analysis

In the previous section bounds were derived for the perturbation in the solution x to a linear system $Ax = b$, when the data A and b are perturbed. Sharper bounds can often be obtained in case if the data is subject to the perturbations, which are bounded component-wise. Assume that

$$|\delta a_{ij}| \le \omega e_{ij}, \quad |\delta b_i| \le \omega f_i. \quad i, j = 1 : n,$$

for some $\omega \ge 0$, where $e_{ij} \ge 0$ and $f_i \ge 0$ are known. These bounds can be written as

$$|\delta A| \le \omega E, \quad |\delta b| \le \omega f, \tag{7.6.15}$$

where the absolute value of a matrix A and vector b is defined by

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

The partial ordering "$\le$" for matrices A, B and vectors x, y, is to be interpreted component-wise.[21] It is easy to show that if $C = AB$, then

$$|c_{ij}| \le \sum_{k=1}^{n} |a_{ik}|\,|b_{kj}|,$$

and hence $|C| \le |A|\,|B|$. A similar rule $|Ax| \le |A|\,|x|$ holds for matrix-vector multiplication.

For deriving the componentwise bounds we need the following result.

Lemma 7.6.8.

Let $F \in \mathbf{R}^{n\times n}$ be a matrix for which $\|\,|F|\,\| < 1$. Then the matrix $(I - |F|)$ is nonsingular and

$$|(I - F)^{-1}| \le (I - |F|)^{-1}. \tag{7.6.16}$$

[21] Note that $A \le B$ in other contexts means that $B - A$ is positive semi-definite.

Proof. The nonsingularity follows form Lemma 7.6.2. Using the identity $(I-F)^{-1} = I + F(I-F)^{-1}$ we obtain

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

from which the inequality (7.6.16) follows. □

Theorem 7.6.9.

Consider the perturbed linear system $(A+\delta A)(x+\delta x) = b+\delta b$, where A is nonsingular. Assume that δA and δb satisfy the componentwise bounds in (7.6.15) and that

$$\omega \| \, |A^{-1}| \, E \, \| < 1.$$

Then $(A+\delta A)$ is nonsingular and

$$\|\delta x\| \leq \frac{\omega}{1-\omega\kappa_E(A)} \| \, |A^{-1}|(E\,|x| + f)\|, \tag{7.6.17}$$

where $\kappa_E(A) = \| \, |A^{-1}|E\|$.

Proof. Taking absolute values in (7.6.3) gives

$$|\delta x| \leq |(I + A^{-1}\delta A)^{-1}| \, |A^{-1}|(|\delta A||x| + |\delta b|). \tag{7.6.18}$$

Using Lemma 7.6.8 it follows from the assumption that the matrix $(I - |A^{-1}|\delta A|)$ is nonsingular and from (7.6.18) we get

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

Using the componentwise bounds in (7.6.15) we get

$$|\delta x| \leq \omega(I - \omega|A^{-1}|E)^{-1}|A^{-1}|(E|x| + f), \tag{7.6.19}$$

provided that $\omega\kappa_E(A) < 1$ Taking norms in (7.6.19) and using Lemma 7.6.2 with $F = A^{-1}\delta A$ proves (7.6.17). □

Taking $E = |A|$ and $f = |b|$ in (7.6.15) corresponds to bounds for the **component-wise relative errors** in A and b,

$$|\delta A| \leq \omega|A|, \qquad |\delta b| \leq \omega|b|. \tag{7.6.20}$$

For this special case Theorem 7.6.9 gives

$$\|\delta x\| \leq \frac{\omega}{1-\omega\kappa_{|A|}(A)} \| \, |A^{-1}|(|A|\,|x| + |b|)\,\|, \tag{7.6.21}$$

where

$$\kappa_{|A|}(A) = \| \, |A^{-1}||A| \, \|, \tag{7.6.22}$$

(or cond (A)) is the **Bauer–Skeel condition number** of the matrix A. Note that since $|b| \le |A|\,|x|$, it follows that

$$\|\delta x\| \le 2\omega \|\,|A^{-1}||A|\,|x|\,\| + O(\omega^2) \le 2\omega \kappa_{|A|}(A)\|x\| + O(\omega^2).$$

If $\hat{A} = DA$, $\hat{b} = Db$ where $D > 0$ is a diagonal scaling matrix, then $|\hat{A}^{-1}| = |A^{-1}||D^{-1}|$. Since the perturbations scale similarly, $\delta\hat{A} = D\delta A$, $\delta\hat{b} = D\delta b$, it follows that

$$|\hat{A}^{-1}||\delta\hat{A}| = |A^{-1}||\delta A|, \qquad |\hat{A}^{-1}||\delta\hat{b}| = |A^{-1}||\delta b|.$$

Thus, the bound in (7.6.21) and also $\kappa_{|A|}(A)$ are *invariant under row scalings.*

For the l_1-norm and l_∞-norm it holds that

$$\kappa_{|A|}(A) = \|\,|A^{-1}||A|\,\| \le \|\,|A^{-1}|\,\|\,\|\,|A|\,\| = \|A^{-1}\|\,\|A\| = \kappa(A),$$

i.e., the solution of $Ax = b$ is no more badly conditioned with respect to the component-wise relative perturbations than with respect to normed perturbations. On the other hand, it is possible for $\kappa_{|A|}(A)$ to be much smaller than $\kappa(A)$.

The analysis in Sec. 7.6.2 may not be adequate, when the perturbations in the elements of A or b are of different magnitude, as illustrated by the following example.

Example 7.6.1.

The linear system $Ax = b$, where

$$A = \begin{pmatrix} 1 & 10^4 \\ 1 & 10^{-4} \end{pmatrix}, \quad b = \begin{pmatrix} 10^4 \\ 1 \end{pmatrix},$$

has the approximate solution $x \approx (1, 1)^T$. Assume that the vector b is subject to a perturbation δb such that $|\delta b| \le (1, 10^{-4})^T$. Using the ∞-norm we have $\|\delta b\|_\infty = 1$, $\|A^{-1}\|_\infty = 1$ (neglecting terms of order 10^{-8}). Theorem 7.6.5 then gives the gross overestimate $\|\delta x\|_\infty \le 1$.

Multiplying the first equation by 10^{-4}, we get an equivalent system $\hat{A}x = \hat{b}$ where

$$\hat{A} = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix}, \quad \hat{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The perturbation in the vector b is now $|\delta\hat{b}| \le 10^{-4}(1, 1)^T$, and from $\|\delta\hat{b}\|_\infty = 10^{-4}$, $\|(\hat{A})^{-1}\|_\infty = 1$, we get the sharp estimate $\|\delta x\|_\infty \le 10^{-4}$. The original matrix A is only **artificially ill-conditioned**. By a scaling of the equations we obtain a well-conditioned system. How to scale linear systems for Gaussian elimination is a surprisingly intricate problem, which is further discussed in Sec. 7.7.2.

Consider the linear systems in Example 7.6.1. Neglecting terms of order 10^{-8} we have

$$|\hat{A}^{-1}||\hat{A}| = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix} \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix} = \begin{pmatrix} 1 & 2\cdot 10^{-4} \\ 2\cdot 10^{-4} & 1 \end{pmatrix},$$

By the scaling invariance $\operatorname{cond}(A) = \operatorname{cond}(\hat{A}) = 1 + 2 \cdot 10^{-4}$ in the ∞-norm. Thus, the componentwise condition number correctly reveals that the system is well-conditioned for componentwise small perturbations.

7.6.4 Backward Error Bounds

It is possible to derive a simple **a posteriori** bound for the backward error of a computed solution $\bar{x}$. These bounds are usually much sharper than a priori bounds and hold regardless of the algorithm used to compute $\bar{x}$.

Given $\bar{x}$, there are an infinite number of perturbations δA and δb for which $(A + \delta A)\bar{x} = b + \delta b$ holds. Clearly δA and δb must satisfy

$$\delta A\bar{x} - \delta b = b - A\bar{x} = r,$$

where $r = b - A\bar{x}$ is the residual vector corresponding to the computes solution. An obvious choice is to take $\delta A = 0$, and $\delta b = -r$. If we instead take $\delta b = 0$, we get the following result.

Theorem 7.6.10.
Let $\bar{x}$ be a purported solution to $Ax = b$, and set $r = b - A\bar{x}$. Then if

$$\delta A = r\bar{x}^T / \|\bar{x}\|_2^2, \tag{7.6.23}$$

$\bar{x}$ satisfies $(A + \delta A)\bar{x} = b$ and this has the smallest l_2-norm $\|\delta A\|_2 = \|r\|_2/\|\bar{x}\|_2$ of any such δA.

Proof. Clearly $\bar{x}$ satisfies $(A + \delta A)\bar{x} = b$ if and only if $\delta A\bar{x} = r$. For any such δA it holds that $\|\delta A\|_2\|\bar{x}\|_2 \geq \|r\|_2$ or $\|\delta A\|_2 \geq \|r\|_2/\|\bar{x}\|_2$. For the particular δA given by (7.6.23) we have $\delta A\bar{x} = r\bar{x}^T\bar{x}/\|\bar{x}\|^2 = r$. From

$$\|r\bar{x}^T\|_2 = \sup_{\|y\|_2=1} \|r\bar{x}^T y\|_2 = \|r\|_2 \sup_{\|y\|_2=1} |\bar{x}^T y| = \|r\|_2\|\bar{x}\|_2,$$

it follows that $\|\delta A\|_2 = \|r\|_2/\|\bar{x}\|_2$ and hence the δA in (7.6.23) is of minimum l_2-norm. $\square$

Similar bounds for the l_1-norm and l_∞-norm are given in Problem 5.

It is often more useful to consider the **component-wise backward error** ω of a computed solution. The following theorem shows that also this can be cheaply computed

Theorem 7.6.11. (Oettli and Prager [1964]).
Let $r = b - A\bar{x}$, E and f be nonnegative and set

$$\omega = \max_i \frac{|r_i|}{(E|\bar{x}| + f)_i}, \tag{7.6.24}$$

where $0/0$ *is interpreted as* 0. *If* $\omega \neq \infty$, *there is a perturbation* δA *and* δb *with*

$$|\delta A| \leq \omega E, \qquad |\delta b| \leq \omega f, \tag{7.6.25}$$

such that

$$(A + \delta A)\bar{x} = b + \delta b. \tag{7.6.26}$$

Moreover, ω *is the smallest number for which such a perturbation exists.*

Proof. From (7.6.24) we have

$$|r_i| \leq \omega(E|\bar{x}| + f)_i,$$

which implies that $r = D(E|\bar{x}| + f)$, where $|D| \leq \omega I$. It is then readily verified that

$$\delta A = DE \ \mathrm{diag}\big(\mathrm{sign}(\bar{x}_1), \ldots, \mathrm{sign}(\bar{x}_n)\big), \qquad \delta b = -Df$$

are the required backward perturbations.

Further, given perturbations δA and δb satisfying equations (7.6.25)–(7.6.26) for some ω we have

$$|r| = |b - A\bar{x}| = |\delta A\bar{x} - \delta b| \leq \omega(E|\bar{x}| + f).$$

Hence, $\omega \geq |r_i|/(E|\bar{x}| + f)_i$, which shows that ω as defined by (7.6.24) is optimal. □

In particular, we can take $E = |A|$, and $f = |b|$ in Theorem 7.6.5, to get an expression for the component-wise relative backward error ω of a computed solution. This can then be used in (7.6.20) or (7.6.21) to compute a bound for $\|\delta x\|$.

Example 7.6.2. Consider the linear system $Ax = b$, where

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix}, \qquad b = \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix}.$$

Suppose that we are given the approximate solution $\bar{x} = (0.9911, -0.4870)^T$. The residual vector corresponding to $\bar{x}$ is very small,

$$r = b - A\bar{x} = (-10^{-8}, 10^{-8})^T.$$

However, not a single figure in $\bar{x}$ is correct! The *exact* solution is $x = (2, -2)^T$, as can readily verified by substitution. *Although a zero residual implies an exact solution, a small residual alone does not necessarily imply an accurate solution.* (Compute the determinant of A and then the inverse A^{-1}!)

It should be emphasized that the system in this example is contrived. In practice one would be highly unfortunate to encounter such an ill-conditioned 2×2 matrix.[22]

[22] As remarked by a prominent expert in error-analysis "Anyone unlucky enough to encounter this sort of calamity has probably already been run over by a truck"!

7.6.5 Estimating Condition Numbers

The perturbation analysis has shown that the norm-wise relative perturbation in the solution x of a linear system can be bounded by

$$\|A^{-1}\| \left(\|\delta A\| + \|\delta b\|/\|x\|\right), \tag{7.6.27}$$

or, in case of componentwise analysis, by

$$\| \, |A^{-1}(| \, |E|x| + f|) \, \|. \tag{7.6.28}$$

To compute these upper bounds exactly is costly since $2n^3$ flops are required to compute A^{-1}, even if the LU factorization of A is known (see Sec. 7.2.4). In practice, it will will suffice with an *estimate* of $\|A^{-1}\|$ (or $\| \, |A^{-1}| \, \|$, which need not be very precise.

The first algorithm for condition estimation to be widely used was suggested by Cline, Moler, Stewart, and Wilkinson [71, 1979]. It is based on computing

$$y = (A^T A)^{-1} u = A^{-1}(A^{-T} u) \tag{7.6.29}$$

by solving the two systems $A^T w = u$ and $Ay = w$. A *lower bound* for $\|A^{-1}\|$ is then given by

$$\|A^{-1}\| \geq \|y\|/\|w\|. \tag{7.6.30}$$

If an LU factorization of A is known this only requires $O(n^2)$ flops. The computation of $y = A^{-T} w$ involves solving the two triangular systems

$$U^T v = u, \qquad L^T w = v.$$

Similarly, the vector y and w are obtained by solving the triangular systems

$$Lz = w, \qquad Uy = z,$$

For (7.6.30) to be a reliable estimate the vector u must be carefully chosen so that it reflects any possible ill-conditioning of A. Note that if A is ill-conditioned this is likely to be reflected in U, whereas L, being unit upper triangular, tends to be well-conditioned. To enhance the growth of v we take $u_i = \pm 1$, $i = 1 : n$, where the sign is chosen to maximize $|v_i|$. The final estimate is taken to be

$$1/\kappa(A) \leq \|w\|/(\|A\|\|y\|), \tag{7.6.31}$$

since then a singular matrix is signaled by zero rather than by ∞ and overflow is avoided. We stress that (7.6.31) always *underestimates* $\kappa(A)$. Usually the l_1-norm is chosen because the matrix norm $\|A\|_1 = \max_j \|a_j\|_1$ can be computed from the columns a_j of A. This is often referred to as the LINPACK condition estimator. A detailed description of an implementation is given in the LINPACK Guide, Dongarra et al. [107, 1979, pp. 11-13]. In practice it has been found that the LINPACK condition estimator seldom is off by a factor more than 10. However, counter examples can be constructed showing that it can fail. This is perhaps to be expected for any estimator using only $O(n^2)$ operations.

Equation (7.6.29) can be interpreted as performing one step of the inverse power method on A^TA using the special starting vector u. As shown in Sec. 10.4.2 this is a standard method for computing the largest singular value $\sigma_1(A^{-1}) = \|A^{-1}\|_2$. An alternative to starting with the vector u is to use a *random* starting vector and perhaps carrying out several steps of inverse iteration with A^TA.

An alternative 1-norm condition estimator has been devised by Hager [194, 1984] and improved by Higham [209, 1988]. This estimates

$$\|B\|_1 = \max_j \sum_{i=1}^{n} |b_{ij}|,$$

assuming that Bx and B^Tx can be computed for an arbitrary vector x. It can also be used to estimate the infinity norm since $\|B\|_\infty = \|B^T\|_1$. It is based on the observation that

$$\|B\|_1 = \max_{x\in S} \|Bx\|_1, \quad S = \{x \in \mathbf{R}^n \mid \|x\|_1 \leq 1\}.$$

is the maximum of a convex function $f(x) = \|Bx\|_1$ over the convex set S. This implies that the maximum is obtained at an extreme point of S, i.e. one of the $2n$ points

$$\{\pm e_j \mid j = 1 : n\},$$

where e_j is the jth column of the unit matrix. If $y_i = (Bx)_i \neq 0$, $i = 1 : n$, then $f(x)$ is differentiable and by the chain rule the gradient is

$$\partial f(x) = \xi^T B, \quad \xi_i = \begin{cases} +1 & \text{if } y_i > 0, \\ -1 & \text{if } y_i < 0. \end{cases}$$

If $y_i = 0$, for some i, then $\partial f(x)$ is a subgradient of f at x. Note that the subgradient is not unique. Since f is convex, the inequality

$$f(y) \geq f(x) + \partial f(x)(y - x) \quad \forall x, y \in \mathbf{R}^n.$$

is always satisfied.

The algorithm starts with the vector $x = n^{-1}e = n^{-1}(1, 1, \ldots, 1)^T$, which is on the boundary of S. We set $\partial f(x) = z^T$, where $z = B^T\xi$, and find an index j for which $|z_j| = \max_i |z_i|$. It can be shown that $|z_j| \leq z^Tx$ then x is a local maximum. If this inequality is satisfied then we stop. By the convexity of $f(x)$ and the fact that $f(e_j) = f(-e_j)$ we conclude that $f(e_j) > f(x)$. Replacing x by e_j we repeat the process. Since the estimates are strictly increasing each vertex of S is visited at most once. The iteration must therefore terminate in a finite number of steps. It has been observed that usually it terminates after just four iterations with the exact value of $\|B\|_1$.

We now show that the final point generated by the algorithm is a local maximum. Assume first that $(Bx)_i \neq 0$ for all i. Then $f(x) = \|Bx\|_1$ is linear in a neighborhood of x. It follows that x is a local maximum of $f(x)$ over S if and only if

$$\partial f(x)(y - x) \leq 0 \quad \forall y \in S.$$

If y is a vertex of S, then $\partial f(x)y = \pm\partial f(x)_i$, for some i since all but one component of y is zero. If $|\partial f(x)_i| \leq \partial f(x)x$, for all i, it follows that $\partial f(x)(y - x) \leq 0$ whenever y is a vertex of S. Since S is the convex hull of its vertices it follows that $\partial f(x)(y - x) \leq 0$, for all $y \in S$. Hence, x is a local maximum. In case some component of Bx is zero the above argument must be slightly modified; see Hager [194, 1984].

Algorithm 7.14. *Hager's 1-norm estimator.*

$$
\begin{array}{l}
x = n^{-1}e \\
\textbf{repeat} \\
\quad y = Bx \\
\quad \xi = \mathrm{sign}\,(y) \\
\quad z = B^T\xi \\
\quad \textbf{if} \;\; \|z\|_\infty \leq z^T x \\
\qquad \gamma = \|y\|_1; \;\; \text{quit} \\
\quad \textbf{end} \\
\quad x = e_j, \text{where} |z_j| = \|z\|_\infty \\
\textbf{end}
\end{array}
$$

To use this algorithm to estimate $\|A^{-1}\|_1 = \|\,|A^{-1}|\,\|_1$, we take $B = A^{-1}$. In each iteration we are then required to solve systems $Ay = x$ and $A^T z = \xi$.

It is less obvious that Hager's estimator can also be used to estimate the componentwise relative error (7.6.28). The problem is then to estimate an expression of the form $\|\,|A^{-1}|g\|_\infty$, for a given vector $g > 0$. Using a clever trick devised by Arioli, Demmel and Duff [9, 1989], this can be reduced to estimating $\|B\|_1$ where

$$
B = (A^{-1}G)^T, \quad G = \mathrm{diag}\,(g_1, \ldots, g_n) > 0.
$$

We have $g = Ge$ where $e = (1, 1, \ldots, 1)^T$ and hence

$$
\|\,|A^{-1}|g\|_\infty = \|\,|A^{-1}|Ge\|_\infty = \|\,|A^{-1}G|e\|_\infty = \|\,|A^{-1}G|\|_\infty = \|(A^{-1}G)^T\|_1,
$$

where in the last step we have used that the ∞ norm is absolute (see Sec. 7.1.5). Since Bx and $B^T y$ can be found by solving linear systems involving A^T and A the work involved is similar to that of the LINPACK estimator. This together with ω determined by (7.6.24) gives an approximate bound for the error in a computed solution $\bar{x}$. Hager's condition estimator is used Matlab.

We note that the unit lower triangular matrices L obtained from Gaussian elimination with pivoting are not arbitrary but their off-diagonal elements satisfy $|l_{ij}| \leq 1$. When Gaussian elimination without pivoting is applied to a row diagonally dominant matrix it gives a row diagonally dominant upper triangular factor $U \in \mathbf{R}^{n\times n}$ satisfying

$$
|u_{ii}| \geq \sum_{j=i+1}^{n} |u_{ij}|, \quad i = 1 : n - 1. \tag{7.6.32}
$$

and it holds that $cond(U) \leq 2n - 1$; (see [211, Lemma 8.8].

Definition 7.6.12. *For any triangular matrix T the* **comparison matrix** *is*

$$M(T) = (m_{ij}), \qquad m_{ij} = \begin{cases} |t_{ii}|, & i = j; \\ -|t_{ij}|, & i \neq j; \end{cases}$$

Review Questions

6.1 How is the condition number $\kappa(A)$ of a matrix A defined? How does $\kappa(A)$ relate to perturbations in the solution x to a linear system $Ax = b$, when A and b are perturbed? Outline roughly a cheap way to estimate $\kappa(A)$.

Problems

6.1 (a) Compute the inverse A^{-1} of the matrix A in Problem 6.4.1 and determine the solution x to $Ax = b$ when $b = (4, 3, 3, 1)^T$.

(b) Assume that the vector b is perturbed by a vector δb such that $\|\delta b\|_\infty \leq 0.01$. Give an upper bound for $\|\delta x\|_\infty$, where δx is the corresponding perturbation in the solution.

(c) Compute the condition number $\kappa_\infty(A)$, and compare it with the bound for the quotient between $\|\delta x\|_\infty/\|x\|_\infty$ and $\|\delta b\|_\infty/\|b\|_\infty$ which can be derived from (b).

6.2 Show that the matrix A in Example 7.6.2 has the inverse

$$A^{-1} = 10^8 \begin{pmatrix} 0.1441 & -0.8648 \\ -0.2161 & 1.2969 \end{pmatrix},$$

and that $\kappa_\infty = \|A\|_\infty \|A^{-1}\|_\infty = 2.1617 \cdot 1.5130 \cdot 10^8 \approx 3.3 \cdot 10^8$, which shows that the system is "perversely" ill-conditioned.

6.3 (Higham [211, p. 144]) Consider the triangular matrix

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

Show that $\operatorname{cond}(U) = 5$ but $\operatorname{cond}(U^T) = 1 + 2/\epsilon$. This shows that a triangular system can be much worse conditioned than its transpose.

6.4 Let the matrix $A \in \mathbf{R}^{n\times n}$ be nonnegative, and solve $A^T x = e$, where $e = (1, 1, \dots, 1)^T$. Show that then $\|A^{-1}\|_1 = \|x\|_\infty$.

6.5 Let $\bar{x}$ be a computed solution and $r = b - A\bar{x}$ the corresponding residual. Assume that δA is such that $(A + \delta A)\bar{x} = b$ holds exactly. Show that the error of minimum l_1-norm and l_∞-norm respectively are given by

$$\delta A = r(s_1, \dots, s_n)/\|\bar{x}\|_1, \quad \delta A = r(0, \dots, 0, s_m, 0, \dots, 0)/\|\bar{x}\|_\infty,$$

where $\|\bar{x}\|_\infty = |x_m|$, and $s_i = 1$, if $x_i \geq 0$; $s_i = -1$, if $x_i < 0$.

6.6 Use the result in Theorem 7.6.6 to obtain the lower bound $\kappa_\infty(A) \geq 1.5|\epsilon|^{-1}$ for the matrix

$$A = \begin{pmatrix} 1 & -1 & 1 \\ -1 & \epsilon & \epsilon \\ 1 & \epsilon & \epsilon \end{pmatrix}, \qquad 0 < |\epsilon| < 1.$$

(The true value is $\kappa_\infty(A) = 1.5(1 + |\epsilon|^{-1})$.)

7.7 Rounding Error Analysis

7.7.1 Error Analysis of Gaussian Elimination

In the practical solution of a linear system of equations, rounding errors are introduced in each arithmetic operation and cause errors in the computed solution. In the early days of the computer era around 1946 many mathematicians were pessimistic about the numerical stability of Gaussian elimination. It was argued that the growth of roundoff errors would make it impractical to solve even systems of fairly moderate size. By the early 1950s experience revealed that this pessimism was unfounded. In practice Gaussian elimination with partial pivoting is a remarkably stable method and has become the universal algorithm for solving dense systems of equations.

The bound given in Theorem 7.2.4 is satisfactory only if the growth factor ρ_n is not too large, but this quantity is only known *after* the elimination has been completed. In order to obtain an *a priori bound* on ρ_n we use the inequality

$$|a_{ij}^{(k+1)}| = |a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}| \leq |a_{ij}^{(k)}| + |a_{kj}^{(k)}| \leq 2 \max_k |\bar{a}_{ij}^{(k)}|,$$

valid If partial pivoting is employed. By induction this gives the upper bound $\rho_n \leq 2^{n-1}$, which is attained for matrices $A_n \in \mathbf{R}^{n\times n}$ of the form

$$A_4 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}. \tag{7.7.1}$$

Already for $n = 54$ we can have $\rho_n = 2^{53} \approx 0.9 \cdot 10^{16}$ and can lose all accuracy using IEEE double precision ($u = 1.11 \cdot 10^{-16}$). Hence, the *worst-case behavior of partial pivoting is very unsatisfactory.*

For complete pivoting, Wilkinson [386, 1961] has proved that

$$\rho_n \leq (n \cdot 2^1 3^{1/2} 4^{1/3} \cdots n^{1/(n-1)})^{1/2} < 1.8\sqrt{n} n^{\frac{1}{4}\log n},$$

and that this bound is not attainable. This bound is *much* smaller than that for partial pivoting, e.g., $\rho_{50} < 530$. It was long conjectured that $\rho_n \leq n$ for real matrices and complete pivoting. This was finally disproved in 1991 when a matrix of order 13 was found for which $\rho_n = 13.0205$. A year later a matrix of order 25 was found for which $\rho_n = 32.986$.

Although complete pivoting has a much smaller worst case growth factor than partial pivoting it is more costly. Moreover, complete (as well as rook) pivoting has the drawback that it cannot be combined with the more efficient blocked methods of Gaussian elimination (see Sec. 7.6.3). Fortunately from decades of experience and extensive experiments it can be concluded that substantial growth in elements using partial pivoting occurs only for a tiny proportion of matrices arising naturally. We quote Wilkinson [387, pp. 213–214].

> *It is our experience that any substantial increase in the size of elements of successive $A^{(k)}$ is extremely uncommon even with partial pivoting. No example which has arisen naturally has in my experience given an increase by a factor as large as* 16.

So far only a few exceptions to the experience related by Wilkinson have been reported. One concerns linear systems arising from a class of two-point boundary value problems, when solved by the shooting method. Another is the class of linear systems arising from a quadrature method for solving a certain Volterra integral equation. These examples show that Gaussian elimination with partial pivoting cannot be unconditionally trusted. When in doubt some safeguard like monitoring the element growth should be incorporated. Another way of checking and improving the reliability of Gaussian elimination with partial pivoting is iterative refinement, which is discussed in Sec. 7.7.3.

Why large element growth rarely occurs with partial pivoting is still not fully understood. Trefethen and Schreiber [361, 1990] have shown that for certain distributions of random matrices the average element growth was close to $n^{2/3}$ for partial pivoting.

We now give a component-wise roundoff analysis for the LU factorization of A. Note that all the variants given in Sec. 7.2 for computing the LU factorization of a matrix will essentially lead to the same error bounds, since each does the same operations with the same arguments. Note that also since Gaussian elimination with pivoting is equivalent to Gaussian elimination without pivoting on a permuted matrix, we need not consider pivoting.

Theorem 7.7.1.

If the LU factorization of A runs to completion then the computed factors $\bar{L}$ and $\bar{U}$ satisfy

$$A + E = \bar{L}\bar{U}, \quad |E| \le \gamma_n |\bar{L}|\,|\bar{U}|, \tag{7.7.2}$$

where $\gamma_n = nu/(1 - nu)$.

Proof. In the algorithms in Sec 7.2.5) we set $l_{ii} = 1$ and compute the other elements in L and U from the equations

$$u_{ij} = a_{ij} - \sum_{p=1}^{i-1} l_{ip} u_{pj}, \quad j \ge i;$$

$$l_{ij} = \Big(a_{ij} - \sum_{p=1}^{j-1} l_{ip}u_{pj}\Big)/u_{jj}, \quad i > j,$$

Using (7.1.91) it follows that the computed elements $\bar{l}_{ip}$ and $\bar{u}_{pj}$ satisfy

$$\Big|a_{ij} - \sum_{p=1}^{r} \bar{l}_{ip}\bar{u}_{pj}\Big| \le \gamma_r \sum_{p=1}^{r} |\bar{l}_{ip}|\,|\bar{u}_{pj}|, \quad r = \min(i, j).$$

where $\bar{l}_{ii} = l_{ii} = 1$. These inequalities may be written in matrix form □

To prove the estimate the error in a computed solution $\bar{x}$ of a linear system given in Theorem 7.7.1, we must also take into account the rounding errors performed in the solution of the two triangular systems $\bar{L}y = b$, $\bar{U}x = y$. A lower triangular system $Ly = b$ is solved by forward substitution

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

If we let $\bar{y}$ denote the computed solution, then using (7.1.89) –(7.1.90) it is straightforward to derive a bound for the backward error in solving a triangular system of equations.

Using the bound for the backward error the forward error in solving a triangular system can be estimated. It is a well known fact that the computed solution is far more accurate than predicted by the normwise condition number. This has been partly explained by Stewart [346, p. 231] as follows:

> "When a matrix is decomposed by Gaussian elimination with partial pivoting for size, the resulting L-factor tends to be well conditioned while any ill-conditioning in the U-factor tends to be artificial."

This observation does not hold in general, for counter examples exist. However, it is true of many special kinds of triangular matrices.

Theorem 7.7.2. *If the lower triangular system $Ly = b$, $L \in \mathbf{R}^{n\times n}$ is solved by substitution with the summation order outlined above, then the computed solution $\bar{y}$ satisfies*

$$(L + \Delta L)\bar{y} = b, \quad |\Delta l_{ki}| \le \begin{cases} \gamma_2|l_{ki}|, & i = k \\ \gamma_{k+1-i}|l_{ki}|, & i = 1 : k-1 \end{cases}, \quad k = 1 : n. \tag{7.7.3}$$

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

An analogue result holds for the computed solution to an upper triangular systems. We conclude the backward stability of substitution for solving triangular systems. Note that it is not necessary to perturb the right hand side.

Theorem 7.7.3.

Let $\bar{x}$ be the computed solution of the system $Ax = b$, using LU factorization and substitution. Then $\bar{x}$ satisfies exactly

$$(A + \Delta A)\bar{x} = b, \tag{7.7.4}$$

where δA is a matrix, depending on both A and b, such that

$$|\Delta A| \le \gamma_n(3 + \gamma_n)|\bar{L}|\,|\bar{U}|. \tag{7.7.5}$$

Proof. From Theorem 7.7.2 it follows that the computed $\bar{y}$ and $\bar{x}$ satisfy

$$(\bar{L} + \delta\bar{L})\bar{y} = b, \qquad (\bar{U} + \delta\bar{U})\bar{x} = \bar{y},$$

where

$$|\delta\bar{L}| \le \gamma_n|\bar{L}|, \qquad |\delta\bar{U}| \le \gamma_n|\bar{U}|. \tag{7.7.6}$$

Note that $\delta\bar{L}$ and $\delta\bar{U}$ depend upon b. Combining these results, it follows that the computed solution $\bar{x}$ satisfies

$$(\bar{L} + \delta\bar{L})(\bar{U} + \delta\bar{U})\bar{x} = b,$$

and using equations (7.7.2)–(7.7.6) proves the backward error

$$|\Delta A| \le \gamma_n(3 + \gamma_n)|\bar{L}|\,|\bar{U}|. \tag{7.7.7}$$

for the computed solution $\bar{x}$ given in Theorem 7.7.1. □

Note that although the perturbation δA depends upon b the *bound* on $|\delta A|$ is independent on b. The elements in $\bar{U}$ satisfy $|\bar{u}_{ij}| \le \rho_n\|A\|_\infty$, and with partial pivoting $|\bar{l}_{ij}| \le 1$. Hence,

$$\|\,|\bar{L}|\,|\bar{U}|\,\|_\infty \le \tfrac{1}{2}n(n+1)\rho_n,$$

and neglecting terms of order $O((nu)^2)$ in (7.7.7) it follows that

$$\|\delta A\|_\infty \le 1.5n(n+1)\gamma_n\rho_n\|A\|_\infty. \tag{7.7.8}$$

By taking b to be the columns $e_1, e_2, \ldots . e_n$ of the unit matrix in succession we obtain the n computed columns of the inverse X of A. For the kth column we have

$$(A + \Delta A_r)\bar{x}_r = e_r,$$

where we have written ΔA_r to emphasize that the perturbation *is different for each column*, Hence we cannot say that Gaussian elimination computes the exact inverse corresponding to some matrix $A + \Delta A$. We obtain the estimate

$$\|A\bar{X} - I\|_\infty \le 1.5n(n+1)\gamma_n\rho_n\|A\|_\infty\|\bar{X}\|_\infty. \tag{7.7.9}$$

From $A\bar{X}-I = E$ it follows that $\bar{X}-A^{-1} = A^{-1}E$ and $\|\bar{X}-A^{-1}\|_\infty \leq \|A^{-1}\|_\infty \|E\|_\infty$, which together with (7.7.9) can be used to get a bound for the error in the computed inverse. We should stress again that we recommend that computing explicit inverses is avoided.

The residual for the computed solution satisfies $\bar{r} = b - A\bar{x} = \delta A\bar{x}$, and using (7.7.8) it follows that

$$\|\bar{r}\|_\infty \leq 1.5n(n+1)\gamma_n\rho_n\|A\|_\infty\|\bar{x}\|_\infty.$$

This shows the remarkable fact that *Gaussian elimination will give a small relative residual even for ill-conditioned systems.* Unless the growth factor is large the quantity

$$\|b - A\bar{x}\|_\infty/(\|A\|_\infty\|\bar{x}\|_\infty)$$

will in practice be of the order nu. It is important to realize that this property of Gaussian elimination is not shared by most other methods for solving linear systems. For example, if we first compute the inverse A^{-1} and then $x = A^{-1}b$ the residual $\bar{r}$ may be much larger even if the accuracy in $\bar{x}$ is about the same.

The error bound in Theorem 7.7.1 is instructive in that it shows that a particularly favourable case is when $|\bar{L}|\,|\bar{U}| = |\bar{L}\bar{U}|$. This is true when $\bar{L}$ and $\bar{U}$ are nonnegative. Then

$$|\bar{L}|\,|\bar{U}| = |\bar{L}\bar{U}| = |A + \Delta A| \leq |A| + |\bar{L}|\,|\bar{U}|,$$

and neglecting terms of order $O((nu)^2)$ we find that the computed $\bar{x}$ satisfies

$$(A + \Delta A)\bar{x} = b, \quad |\Delta A| \leq 3\gamma_n|A|.$$

A class of matrices for which Gaussian elimination without pivoting gives positive factors L and U is the following.

Definition 7.7.4.

A matrix $A \in \mathbf{R}^{n\times n}$ is called **totally positive** *if the determinant of every square submatrix of A is positive.*

It is known (see de Boor and Pinkus [89, 1977]) that if A is totally positive, then it has an LU factorization with $L > 0$ and $U > 0$. Since the property of a matrix being totally positive is destroyed under row permutations, *pivoting should not be used when solving such systems.* Totally positive systems occur in spline interpolation.

In many cases there is no a priori bound for the matrix $|\bar{L}|\,|\bar{U}|$ appearing in the componentwise error analysis. It is then possible to compute its ∞-norm in $O(n^2)$ operations without forming the matrix explicitly, since

$$\|\,|\bar{L}|\,|\bar{U}|\,\|_\infty = \|\,|\bar{L}|\,|\bar{U}|e\,\|_\infty = \|\,|\bar{L}|\,(|\bar{U}|e)\,\|_\infty.$$

This useful observation is due to Chu and George [67, 1985].

An error analysis for the Cholesky factorization of a symmetric positive definite matrix $A \in \mathbf{R}^{n\times n}$ is similar to that for LU factorization.

Theorem 7.7.5.

Suppose that the Cholesky factorization of a symmetric positive definite matrix $A \in R^{n\times n}$ runs to completion and produces a computed factor $\bar{R}$ and a computed solution $\bar{x}$ to the linear system. Then it holds that

$$A + E_1 = \bar{L}\bar{U}, \quad |E_1| \le \gamma_{n+1}|\bar{R}^T|\,|\bar{R}|, \tag{7.7.10}$$

and

$$(A + E_2)\bar{x} = b, \quad |E_2| \le \gamma_{3n+1}|\bar{R}^T|\,|\bar{R}|. \tag{7.7.11}$$

Theorem 7.7.6. [J. H. Wilkinson [389, 1968]]

Let $A \in R^{n\times n}$ be a symmetric positive definite matrix. The Cholesky factor of A can be computed without breakdown provided that $2n^{3/2}u\kappa(A) < 0.1$. The computed $\bar{L}$ satisfies

$$\bar{L}\bar{L}^T = A + E, \quad \|E\|_2 < 2.5n^{3/2}u\|A\|_2, \tag{7.7.12}$$

and hence is the exact Cholesky factor of a matrix close to A.

This is essentially the best normwise bounds that can be obtained, although Meinguet [277, 1983] has shown that for large n the constants 2 and 2.5 in Theorem 7.7.6 can be improved to 1 and 2/3, respectively.

In practice we can usually expect much smaller backward error in the computed solutions than the bounds derived in this section. It is appropriate to recall here a remark by J. H. Wilkinson (1974):

> "All too often, too much attention is paid to the precise error bound that has been established. The main purpose of such an analysis is either to establish the essential numerical stability of an algorithm or to show why it is unstable and in doing so expose what sort of change is necessary to to make it stable. The precise error bound is not of great importance."

7.7.2 Scaling of Linear Systems

In a linear system of equations $Ax = b$ the ith equation may be multiplied by an arbitrary positive scale factor d_i, $i = 1 : n$, without changing the exact solution. In contrast, such a scaling will usually change the computed numerical solution. In this section we show that *a proper row scaling is important for Gaussian elimination with partial pivoting to give accurate computed solutions*, and give some rules for scaling.

We first show that *if the pivot sequence is fixed* then Gaussian elimination is unaffected by such scalings, or more precisely:

Theorem 7.7.7.

Denote by $\bar{x}$ and $\bar{x}'$ the computed solutions obtained by Gaussian elimination in floating-point arithmetic to the two linear systems of equations

$$Ax = b, \qquad (D_r A D_c)x' = D_r b,$$

where D_r and D_c are diagonal scaling matrices. Assume that the elements of D_r and D_c are powers of the base of the number system used, so that no rounding errors are introduced by the scaling. Then if the same pivot sequence is used and no overflow or underflow occurs we have exactly $\bar{x} = D_c \bar{x}'$, i.e., the components in the solution differ only in the exponents.

Proof. The proof follows by examination of the scaling invariance of the basic step in Algorithm 7.2.1

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - (a_{ik}^{(k)} a_{kj}^{(k)})/a_{kk}^{(k)}.$$

□

This result has the important implication that scaling will affect the accuracy of a computed solution only if it leads to a change in the selection of pivots. When partial pivoting is used the row scaling may affect the choice of pivots; indeed we can always find a row scaling which leads to *any predetermined pivot sequence*. However, since only elements in the pivotal column are compared, the choice of pivots is independent of the column scaling. Since a bad choice of pivots can give rise to large errors in the computed solution, it follows that for Gaussian elimination with partial pivoting to give accurate solutions *a proper row scaling is important*.

Example 7.7.1. The system $Ax = b$ in Example 7.6.1 has the solution $x = (0.9999, 0.9999)^T$, correctly rounded to four decimals. Partial pivoting will here select the element a_{11} as pivot. Using three-figure floating-point arithmetic, the computed solution becomes

$$\bar{x} = (0, 1.00)^T \qquad \text{(Bad!)}.$$

If Gaussian elimination instead is carried out on the scaled system $\hat{A}x = \hat{b}$, then a_{21} will be chosen as pivot, and the computed solution becomes

$$\bar{x} = (1.00, 1.00)^T \qquad \text{(Good!)}.$$

From the above discussion we conclude that the need for a proper scaling is of great importance for Gaussian elimination to yield good accuracy. As discussed in Sec. 7.6.3, an estimate of $\kappa(A)$ is often used to access the accuracy of the computed solution. If the perturbation bound (7.6.5) is applied to the scaled system $(D_r A D_c)x' = D_r b$, then we obtain

$$\frac{\|D_c^{-1}\delta x\|}{\|D_c^{-1}x\|} \leq \kappa(D_r A D_c)\frac{\|D_r \delta b\|}{\|D_r b\|}. \tag{7.7.13}$$

Hence, if $\kappa(D_rAD_c)$ can be made smaller than $\kappa(A)$, then it seems that we might expect a correspondingly more accurate solution. Note however that in (7.7.13) the perturbation in x is measured in the norm $\|D_c^{-1}x\|$, and we may only have found a norm in which the error *looks* better! We conclude that the column scaling D_c should be chosen in a way that reflects the importance of errors in the components of the solution. If $|x| \approx c$, and we want the same relative accuracy in all components we may take $D_c = diag(c)$.

We now discuss the choice of row scaling. A scheme which is sometimes advocated is to choose $D_r = \operatorname{diag}(d_i)$ so that each row in D_rA has the same l_1-norm, i.e.,

$$d_i = 1/\|a_i^T\|_1, \quad i = 1:n. \tag{7.7.14}$$

(Sometimes the l_∞-norm, of the rows are instead made equal.) This scaling, called **row equilibration**, can be seen to avoid the bad pivot selection in Example 7.6.1. However, suppose that through an unfortunate choice of physical units the solution x has components of widely varying magnitude. Then, as shown by the following example, row equilibration can lead to a *worse* computed solution than if no scaling is used!

Example 7.7.2. Consider the following system

$$A = \begin{pmatrix} 3\cdot 10^{-6} & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & -1 \end{pmatrix}, \qquad b = \begin{pmatrix} 3 + 3\cdot 10^{-6} \\ 6 \\ 2 \end{pmatrix} \qquad |\epsilon| \ll 1$$

which has the exact solution $x = (1, 1, 1)^T$. The matrix A is *well-conditioned*, $\kappa(A) \approx 3.52$, but the choice of a_{11} as pivot leads to a disastrous loss of accuracy. Assume that through an unfortunate choice of units, the system has been changed into

$$\hat{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2\cdot 10^{6} & 2 & 2 \\ 10^{6} & 2 & -1 \end{pmatrix},$$

with exact solution $\hat{x} = (10^{-6}, 1, 1)^T$. If now the rows are equilibrated, the system becomes

$$\tilde{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 2\cdot 10^{-6} & 2\cdot 10^{-6} \\ 1 & 2\cdot 10^{-6} & -10^{-6} \end{pmatrix}, \qquad \tilde{b} = \begin{pmatrix} 3 + 3\cdot 10^{-6} \\ 6\cdot 10^{-6} \\ 2\cdot 10^{-6} \end{pmatrix}.$$

Gaussian elimination with column pivoting will now choose a_{11} as pivot. Using floating-point arithmetic with precision $u = 0.47\cdot 10^{-9}$ we get the computed solution of $\hat{A}x = \hat{b}$

$$\bar{x} = (0.999894122\cdot 10^{-6},\ 0.999983255,\ 1.000033489)^T.$$

This has only about four correct digits, so almost six digits have been lost!

A theoretical solution to the row scaling problem in Gaussian elimination with partial pivoting has been given by R. D. Skeel [334, 1979]. He shows a pivoting rule in Gaussian elimination should depend not only on the coefficient matrix but also on the solution. His scaling rule is instead based on minimizing a bound on the backward error that contains the quantity

$$\frac{\max_i(|D_rA||\bar{x}|)_i}{\min_i(|D_rA||\bar{x}|)_i}.$$

Scaling Rule: (R. D. Skeel) Assume that $\min_i(|A||x|)_i > 0$. Then scale the rows of A and b by $D_r = \operatorname{diag}(d_i)$, where

$$d_i = 1/(|A||x|)_i, \quad i = 1 : n. \tag{7.7.15}$$

A measure of **ill-scaling** of the system $Ax = b$ is

$$\sigma(A, x) = \max_i(|A||x|)_i / \min_i(|A||x|)_i. \tag{7.7.16}$$

This scaling rule gives infinite scale factors for rows which satisfy $(|A||x|)_i = 0$. This may occur for sparse systems, i.e., when A (and possibly also x) has many zero components. In this case a large scale factor d_i should be chosen so that the corresponding row is selected as pivot row at the first opportunity.

Unfortunately scaling according to this rule is not in general practical, since it assumes that the solution x is at least approximately known. If the components of the solution vector x are known to be of the same magnitude then we can take $|x| = (1, \dots, 1)^T$ in (7.7.15), which corresponds to row equilibration. Note that this assumption is violated in Example 7.7.2.

7.7.3 Iterative Refinement of Solutions

So far we have considered ways of *estimating* the accuracy of computed solutions. We now consider methods for *improving* the accuracy. Let $\bar{x}$ be any approximate solution to the linear system of equations $Ax = b$ and let $r = b - A\bar{x}$ be the corresponding residual vector. Then one can attempt to improve the solution by solving the system $A\delta = r$ for a correction δ and taking $x_c = \bar{x} + \delta$ as a new approximation. If no further rounding errors are performed in the computation of δ this is the exact solution. Otherwise this refinement process can be iterated. In floating-point arithmetic with base β this process of **iterative refinement** can be described as follows:

$s := 1;\quad x^{(s)} := \bar{x};$

repeat

$\quad r^{(s)} := b - Ax^{(s)};\qquad$ (in precision $u_2 = \beta^{-t_2}$)

$\quad$ solve $A\delta^{(s)} = r^{(s)};\qquad$ (in precision $u_1 = \beta^{-t_1}$)

$\quad x^{(s+1)} := x^{(s)} + \delta^{(s)};$

$\quad s := s + 1;$

end

When $\bar{x}$ has been computed by Gaussian elimination this approach is attractive since we can use the computed factors $\bar{L}$ and $\bar{U}$ to solve for the corrections

$$\bar{L}(\bar{U}\delta^{(s)}) = r^{(s)}, \quad s = 1, 2, \dots .$$

The computation of $r^{(s)}$ and $\delta^{(s)}$, therefore, only takes $2n^2 + 2 \cdot n^2 = 4n^2$ flops, which is an order of magnitude less than the $2n^3/3$ flops required for the initial solution.

We note the possibility of using *extended precision* $t_2 > t_1$ for computing the residuals $r^{(s)}$; these are then rounded to single precision u_1 before solving for $\delta^{(s)}$. Since $x^{(s)}$, A and b are stored in single precision, only the accumulation of the inner product terms are in precision u_2, and no multiplications in extended precision occur. This is also called *mixed precision iterative refinement* as opposed to *fixed precision iterative refinement* when $t_2 = t_1$.

Since the product of two t digit floating-point numbers can be exactly represented with at most $2t$ digits inner products can be computed in extended precision without much extra cost. If fl_e denotes computation with extended precision and u_e the corresponding unit roundoff then the forward error bound for an inner product becomes

$$|fl\,(fl_e((x^Ty)) - x^Ty| < u|x^Ty| + \frac{nu_e}{1 - nu_e/2}(1 + u)|x^T||y|, \tag{7.7.17}$$

where the first term comes form the final rounding. If $|x^T||y| \le u|x^Ty|$ then the computed inner product is almost as accurate as the correctly rounded exact result. However, since computations in extended precision are machine dependent it has been difficult to make such programs portable.[23] The development of extended and mixed Precision BLAS (Basic Linear Algebra Subroutines) (see [267, 2000]) has madee this feasible. A portable and paralleliazble implementation of the mixed precision algorithm is described in [93].

In the ideal case that the rounding errors committed in computing the corrections can be neglected we have

$$x^{(s+1)} - x = (I - (\bar{L}\bar{U})^{-1}A)^s(\bar{x} - x).$$

where $\bar{L}$ and $\bar{U}$ denote the computed LU factors of A. Hence, the process converges if

$$\rho = \|(I - (\bar{L}\bar{U})^{-1}A\| < 1.$$

This roughly describes how the refinement behaves in the *early stages*, if extended precision is used for the residuals. If $\bar{L}$ and $\bar{U}$ have been computed by Gaussian elimination using precision u_1, then by Theorem 7.2.4 we have

$$\bar{L}\bar{U} = A + E, \quad \|E\|_\infty \le 1.5n^2\rho_n u_1\|A\|_\infty,$$

and ρ_n is the growth factor. It follows that an upper bound for the initial rate of convergence is given by

$$\rho = \|(\bar{L}\bar{U})^{-1}E\|_\infty \le n^2\rho_n u_1\kappa(A).$$

[23]It was suggested that the IEEE 754 standard should require inner products to be precisely specified, but that did not happen.

When also rounding errors in computing the residuals $r^{(s)}$ and the corrections $\delta^{(s)}$ are taken into account, the analysis becomes much more complicated. The behavior of iterative refinement, using t_1-digits for the factorization and $t_2 = 2t_1$ digits when computing the residuals, can be summed up as follows:

1. Assume that A is not too ill-conditioned so that the first solution has some accuracy, $\|x - \bar{x}\|/\|x\| \approx \beta^{-k} < 1$ in some norm. Then the relative error diminishes by a factor of roughly β^{-k} with each step of refinement until we reach a stage at which $\|\delta_c\|/\|x_c\| < \beta^{-t_1}$, when we may say that the solution is correct to working precision.

2. In general the attainable accuracy is limited to $\min(k + t_2 - t_1, t_1)$ digits, which gives the case above when $t_2 \geq 2t_1$. Note that although the computed solution improves progressively with each iteration this is *not always reflected* in a corresponding decrease in the norm of the residual, which may stay about the same.

Iterative refinement can be used to compute a more accurate solution, in case A is ill-conditioned. However, unless A and b are exactly known this may not make much sense. The exact answer to a poorly conditioned problem may be no more appropriate than one which is correct to only a few places.

In many descriptions of iterative refinement it is stressed that it is essential that the residuals are computed with a higher precision than the rest of the computation, for the process to yield a more accurate solution. This is true if the initial solution has been computed by a backward stable method, such as Gaussian elimination with partial pivoting, and provided that the system is well scaled. However, iterative refinement using single precision residuals, *can considerably improve the quality of the solution, for example, when the system is ill-scaled*, i.e., when $\sigma(A, x)$ defined by (7.7.16) is large, or if the pivot strategy has been chosen for the preservation of sparsity, see Sec. 7.6.

Example 7.7.3.

As an illustration consider again the badly scaled system in Example 7.7.1

$$\tilde{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 2\cdot 10^{-6} & 2\cdot 10^{-6} \\ 1 & 2\cdot 10^{-6} & -10^{-6} \end{pmatrix}, \qquad \tilde{b} = \begin{pmatrix} 3 + 3\cdot 10^{-6} \\ 6\cdot 10^{-6} \\ 2\cdot 10^{-6} \end{pmatrix},$$

with exact solution $\tilde{x} = (10^{-6}, 1, 1)^T$. Using floating-point arithmetic with unit roundoff $u = 0.47\cdot 10^{-9}$ the solution computed by Gaussian elimination with partial pivoting has only about four correct digits. From the residual $r = \tilde{b} - \tilde{A}\bar{x}$ we compute the Oettli–Prager backward error $\omega = 0.28810\cdot 10^{-4}$. The condition estimate computed by (7.6.31) is $3.00\cdot 10^6$, and wrongly indicates that the loss of accuracy should be blamed on ill-conditioning.

With one step of iterative refinement using a single precision residual we get

$$\tilde{x} = \bar{x} + d = (\,0.999999997\cdot 10^{-6} \quad 1.000000000 \quad 1.000000000\,)^T.$$

This should be recomputed using IEEE single and double precision! This is almost as good as for Gaussian elimination with column pivoting applied to the system $Ax = b$. The Oettli–Prager error bound for $\tilde{x}$ is $\omega = 0.54328 \cdot 10^{-9}$, which is close to the machine precision. Hence, one step of iterative refinement sufficed to correct for the bad scaling. If the ill-scaling is worse or the system is also ill-conditioned then several steps of refinement may be needed.

The following theorem states that if Gaussian elimination with partial pivoting is combined with iterative refinement in single precision then the resulting method will give a small relative backward error provided that the system is not too ill-conditioned or ill-scaled.

Theorem 7.7.8. (R. D. Skeel.)
As long as the product of $\operatorname{cond}(A^{-1}) = \||A||A^{-1}|\|_\infty$ *and* $\sigma(A, x)$ *is sufficiently less than* $1/u$, *where* u *is the machine unit, it holds that*

$$(A + \delta A)x^{(s)} = b + \delta b, \quad |\delta a_{ij}| < 4n\epsilon_1 |a_{ij}|, \quad |\delta b_i| < 4n\epsilon_1 |b_i|, \tag{7.7.18}$$

for s *large enough. Moreover, the result is often true already for* $s = 2$, *i.e., after only one improvement.*

Proof. For exact conditions under which this theorem holds, see Skeel [335, 1980]. □

As illustrated above, Gaussian elimination with partial or complete pivoting may not provide all the accuracy that the data deserves. How often this happens in practice is not known. In cases where accuracy is important the following scheme, which offers improved reliability for a small cost is recommended.

1. Compute the Oettli–Prager backward error ω using (7.6.24) with $E = |A|$, $f = |b|$, by simultaneously accumulating $r = b - A\bar{x}$ and $|A||\bar{x}| + |b|$. If ω is not sufficiently small go to 2.

2. Perform one step of iterative refinement using the single precision residual r computed in step 1 to obtain the improved solution $\tilde{x}$. Compute the backward error $\tilde{\omega}$ of $\tilde{x}$. Repeat until the test on $\tilde{\omega}$ is passed.

7.7.4 Interval Matrix Computations

In order to treat multidimensional problems we introduce interval vectors and matrices. An interval vector is denoted by $[x]$ and has interval components $[x_i] = [\underline{x_i}, \overline{x_i}])$, $i = 1 : n$. Likewise an interval matrix $[A] = ([a_{ij}])$ has interval elements $[a_{ij}] = [\underline{a}_{ij}, \overline{a}_{ij}]$, $i = 1 : m$, $j = 1 : n$,

Operations between interval matrices and interval vectors are defined in an obvious manner. The interval matrix-vector product $[A][x]$ is the smallest interval vector, which contains the set $\{Ax \mid A \in [A],\ x \in [x]\}$, but normally does not coincide with it. By the inclusion property it holds that

$$\{Ax \mid A \in [A],\ x \in [x]\} \subseteq [A][x] = \Bigg(\sum_{j=1}^{n} [a_{ij}][x_j]\Bigg). \tag{7.7.19}$$

In general, there will be an overestimation in enclosing the image with an interval vector caused by the fact that the image of an interval vector under a transformation in general is not an interval vector. This phenomenon, intrinsic to interval computations, is called the **wrapping effect**.

Example 7.7.4.
We have

$$A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad [x] = \begin{pmatrix} [0,1] \\ [0,1] \end{pmatrix}, \quad \Rightarrow \quad A[x] = \begin{pmatrix} [0,2] \\ [-1,1] \end{pmatrix}.$$

Hence, $b = (\,2 \quad -1\,)^T \in A[x]$, but there is no $x \in [x]$ such that $Ax = b$. (The solution to $Ax = b$ is $x = (\,3/2 \quad 1/2\,)^T$.)

The magnitude of an interval vector or matrix is interpreted component-wise and defined by

$$|\,[x]\,| = (|\,[x_1]\,|, |\,[x_2]\,|, \ldots, |\,[x_n]\,|)^T,$$

where the magnitude of the components are defined by

$$|\,[a,b]\,| = \max\{|x| \mid x \in [a,b]\}, \tag{7.7.20}$$

The ∞-norm of an interval vector or matrix is defined as the ∞-norm of their magnitude,

$$\|\,[x]\,\|_\infty = \|\,|\,[x]\,|\,\|_\infty, \qquad \|\,[A]\,\|_\infty = \|\,|\,[A]\,|\,\|_\infty. \tag{7.7.21}$$

Using interval arithmetic it is possible to compute strict enclosures of the product of two matrices. Note that this is needed also in the case of the product of two point matrices since rounding errors will in general occur. In this case we want to compute an interval matrix $[C]$ such that $fl(A \cdot B) \subset [C] = [C_{\text{inf}}, C_{\text{sup}}]$. The following simple code does that using two matrix multiplications:

$$\begin{array}{ll} \text{setround}(-1); & C_{\text{inf}} = A \cdot B; \\ \text{setround}(1); & C_{\text{sup}} = A \cdot B; \end{array}$$

Here and in the following we assume that the command setround(i), $i = -1, 0, 1$ sets the rounding mode to $-\infty$, to nearest, and to $+\infty$, respectively.

We next consider the product of a point matrix A and an interval matrix $[B] = [B_{\text{inf}}, B_{\text{sup}}]$. In implementing matrix multiplication it is important to avoid case distinctions in the inner loops, because that would make it impossible to use fast vector and matrix operations. The following code, suggested by A. Neumeier, performs this task efficiently using four matrix multiplications:

$$
\begin{aligned}
&A_- = \min(A,\ 0); \quad A_+ = \max(A,\ 0);\\
&\text{setround}(-1);\\
&C_{\text{inf}} = A_+ \cdot B_{\text{inf}} + A_- \cdot B_{\text{sup}};\\
&\text{setround}(1);\\
&C_{\text{sup}} = A_- \cdot B_{\text{inf}} + A_+ \cdot B_{\text{sup}};
\end{aligned}
$$

(Note that the commands $A_- = \min(A,\ 0)$ and $A_+ = \max(A,\ 0)$ acts componentwise.) Rump [324, 1999] gives an algorithm for computing the product of two interval matrices using eight matrix multiplications. He also gives several faster implementations, provided a certain overestimation can be allowed.

A square interval matrix $[A]$ is called nonsingular if it does not contain a singular matrix. An interval linear system is a system of the form $[A]\,x = [b]$, where A is a nonsingular interval matrix and b an interval vector. The solution set of such an interval linear system is the set

$$
\mathcal{X} = \{x \mid Ax = b,\ A \in [A],\ b \in [b]\}. \tag{7.7.22}
$$

Computing this solution set can be shown to be an intractable problem (NP-complete). Even for a 2×2 linear system this set may not be easy to represent.

Example 7.7.5. (E. Hansen [199, Chapter 4])

Consider a linear system with

$$
[A] = \begin{pmatrix} [2,3] & [0,1] \\ [1,2] & [2,3] \end{pmatrix}, \quad [b] = \begin{pmatrix} [0,120] \\ [60,240] \end{pmatrix}. \tag{7.7.23}
$$

The solution set $\mathcal{X}$ in (7.7.22) is the star shaped region in Figure 7.7.1.

An enclosure of the solution set of an interval linear system can be computed by a generalization of Gaussian elimination adopted to interval coefficients. The solution of the resulting interval triangular system will give an inclusion of the solution set. Realistic bounds can be obtained in this way only for special classes of matrices, e.g., for diagonally dominant matrices and tridiagonal matrices; see Hargreaves [202, 2002]. For general systems this approach unfortunately tends to give interval sizes which grow exponentially during the elimination. For example, if $[x]$ and $[y]$ are intervals then in the 2×2 reduction

$$
\begin{pmatrix} 1 & [x] \\ 1 & [y] \end{pmatrix} \sim \begin{pmatrix} 1 & [x] \\ 0 & [y]-[x] \end{pmatrix}.
$$

If $[x] \approx [y]$ the size of the interval $[y]-[x]$ will be twice the size of $[x]$ and will lead to exponential growth of the inclusion intervals. Even for well-conditioned linear systems the elimination can break down prematurely, because all remaining possible pivot elements contain zero.

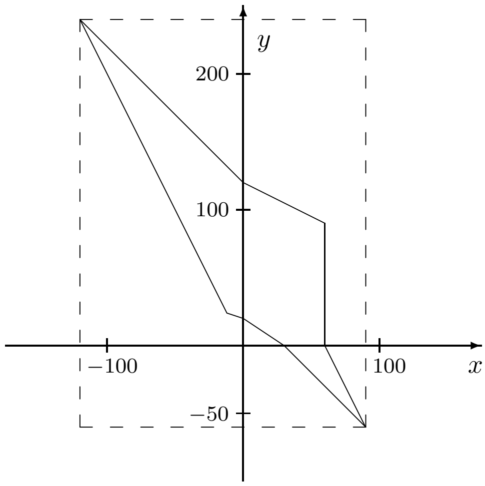

Figure 7.7.1. *The solution set (solid line) and its enclosing hull (dashed line) for the linear system in Example* 7.6.5.

A better way to compute verified bounds on a point or interval linear system uses an idea that goes back to E. Hansen [198, 1965]. In this an approximate inverse C is used to precondition the system. Assuming that an initial interval vector $[x^{(0)}]$ is known, such that $[x^{(0)}] \supseteq \mathcal{X}$ where $\mathcal{X}$ is the solution set (7.7.22). An improved enclosure can then be obtained as follows:

By the inclusion property of interval arithmetic, for all $\tilde{A} \in [A]$ and $\tilde{b} \in [b]$ it holds that

$$[x^{(1)}] = \tilde{A}^{-1}\tilde{b} = C\tilde{b} + (I - C\tilde{A})\tilde{A}^{-1}\tilde{b} \in C\,[b] + (I - C\,[A])[x^{(0)}]$$

This suggests the iteration known as **Krawczyck's method**

$$[x^{(i+1)}] = \Big(C\,[b] + (I - C[A])[x^{(i)}]\Big) \cap [x^{(i)}], \quad i = 0, 1, 2, \ldots, \tag{7.7.24}$$

for computing a sequence of interval enclosures $[x^{(i)}]$ of the solution. Here the interval vector $[c] = C\,[b]$ and interval matrix $[E] = I - C\,[A]$ need only be computed once. The dominating cost per iteration is one interval matrix-vector multiplication.

As approximate inverse we can take the inverse of the midpoint matrix $C = (\mathrm{mid}\,[A])^{-1}$. An initial interval can be chosen of the form

$$[x^{(0)}] = C\mathrm{mid}\,[b] + [-\beta, \beta]e, \quad e = (1, 1, \ldots, 1),$$

with β sufficiently large. The iterations are terminated when the bounds are no longer improving. A measure of convergence can be computed as $\rho = \|[E]\|_\infty$.

Rump [324, 323] has developed a MATLABtoolbox INTLAB (INTerval LABoratory). This is very efficient and easy to use and includes many useful subroutines. INTLAB uses a variant of Krawczyck's method, applied to a residual system, to compute an enclosure of the difference between the solution and an approximate solution $x_m = C\mathrm{mid}\,[b]$; see Rump [324, 1999].

Example 7.7.6.

A method for computing an enclosure of the inverse of an interval matrix can be obtained by taking $[b]$ equal to the identity matrix in the iteration (7.7.24) and solving the system $[A][X] = I$. For the symmetric interval matrix

$$[A] = \begin{pmatrix} [0.999, 1.01] & [-0.001, 0.001] \\ [-0.001, 0.001] & [0.999, 1.01] \end{pmatrix}$$

the identity $C = \text{mid}\,[A] = I$ is an approximate point inverse. We find

$$[E] = I - C[A] = \begin{pmatrix} [-0.01, 0.001] & [-0.001, 0.001] \\ [-0.001, 0.001] & [-0.01, 1.001] \end{pmatrix},$$

and as an enclosure for the inverse matrix we can take

$$[X^{(0)}] = \begin{pmatrix} [0.98, 1.02] & [-0.002, 0.002] \\ [-0.002, 0.002] & [0.98, 1.02] \end{pmatrix}.$$

The iteration

$$[X^{(i+1)}] = \Big(I + E[X^{(i)}]\Big) \cap [X^{(i)}], \quad i = 0, 1, 2, \ldots.$$

converges rapidly in this case.

Review Questions

7.1 The result of a roundoff error analysis of Gaussian elimination can be stated in the form of a backward error analysis. Formulate this result. (You don't need to know the precise expression of the constants involved.)

7.2 (a) Describe the main steps in iterative refinement with extended precision for computing more accurate solutions of linear system.

(b) Sometimes it is worthwhile to do a step of iterative refinement in using fixed precision. When is that?

Problems

7.1 Compute the LU factors of the matrix in (7.7.1).

7.8 Sparse Linear Systems

A matrix $A \in \mathbf{R}^{n\times n}$ is called **sparse** if only a small fraction of its elements are nonzero. Similarly, a linear system $Ax = b$ is called sparse if its matrix A is sparse. The simplest class of sparse matrices is the class of banded matrices treated in Sec. 7.4. These have the property that in each row all nonzero elements are

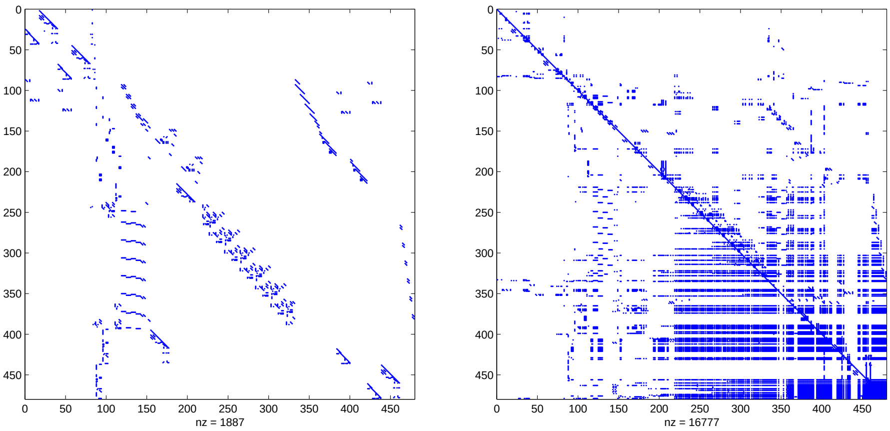

Figure 7.8.1. *Nonzero pattern of a matrix W and its LU factors.*

contained in a relatively narrow band centered around the main diagonal. Matrices of small bandwidth occur naturally, since they correspond to a situation where only variables "close" to each other are coupled by observations.

Large sparse linear systems of more general structure arise in numerous areas of application such as the numerical solution of partial differential equations, mathematical programming, structural analysis, chemical engineering, electrical circuits and networks, etc. Large could imply a value of n in the range 1,000–1,000,000. Typically, A will have only a few (say, 5–30) nonzero elements in each row, regardless of the value of n. In Figure 7.8.1 we show a sparse matrix of order 479 with 1887 nonzero elements and its LU factorization. It is a matrix W called west0479 in the Harwell–Boeing sparse matrix test collection, see Duff, Grimes and Lewis [117, 1989]. It comes from a model due to Westerberg of an eight stage chemical distillation column. Other applications may give a pattern with quite different characteristics.

For many sparse linear systems iterative methods (see Chapter 11) may be preferable to use. This is particularly true of linear systems derived by finite difference methods for partial differential equations in two and three dimensions. In this section we will study elimination methods for sparse systems. These are easier to develop as black box algorithms. Iterative methods, on the other hand, often have to be specially designed for a particular class of problems.

When solving sparse linear systems by direct methods it is important to avoid storing and operating on the elements which are known to be zero. One should also try to minimize **fill-in** as the computation proceeds, which is the term used to denote the creation of new nonzeros during the elimination. For example, as shown in Figure 7.8.1, the LU factors of W contain 16777 nonzero elements about nine times as many as in the original matrix. The object is to reduce storage and the number of arithmetic operations. Indeed, without exploitation of sparsity, many large problems would be totally intractable.

7.8.1 Storage Schemes for Sparse Matrices

A simple scheme to store a sparse matrix is to store the nonzero elements in an unordered one-dimensional array AC together with two integer vectors ix and jx containing the corresponding row and column indices.

$$ac(k) = a_{i,j}, \quad i = ix(k), \quad j = jx(k), \quad k = 1 : nz.$$

Hence, A is stored in "coordinate form" as an unordered set of triples consisting of a numerical value and two indices. This scheme is very convenient for the initial representation of a general sparse matrix. Note that further nonzero elements are easily added to the structure. This coordinate form is very convenient for the original input of a sparse matrix. A drawback is that using this storage structure it is difficult to access the matrix A by rows or by columns, which is needed for the implementation of Gaussian elimination.

Example 7.8.1. The matrix

$$A = \begin{pmatrix} a_{11} & 0 & a_{13} & 0 & 0 \\ a_{21} & a_{22} & 0 & a_{24} & 0 \\ 0 & a_{32} & a_{33} & 0 & a_{35} \\ 0 & a_{42} & 0 & a_{44} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} \end{pmatrix},$$

is stored in coordinate form as

$$\begin{aligned} AC &= (a_{13}, a_{22}, a_{21}, a_{33}, a_{35}, a_{24}, a_{32}, a_{42}, a_{44}, a_{55}, a_{54}, a_{11}) \\ i &= (1, 2, 2, 3, 3, 2, 3, 4, 4, 5, 5, 1) \\ j &= (3, 2, 1, 3, 5, 4, 2, 2, 4, 5, 4, 1) \end{aligned}$$

In some applications, one encounters matrices of banded structure, where the bandwidth differs from row to row. For this class of matrices, called **variable-band matrices**, we define

$$f_i = f_i(A) = \min\{j \mid a_{ij} \neq 0\}, \quad l_j = l_j(A) = \min\{i \mid a_{ij} \neq 0\}. \tag{7.8.1}$$

Here f_i is the column subscript of the first nonzero in the i-th *row* of A, and similarly l_j the row subscript of the first nonzero in the jth *column* of A. We assume here and in the following that A has a zero free diagonal. From the definition it follows that $f_i(A) = l_i(A^T)$. Hence, for a symmetric matrix A we have $f_i(A) = l_i(A)$, $i = 1 : n$.

Definition 7.8.1.

The **envelope** *(or profile) of A is the index set*

$$\text{Env}\,(A) = \{(i, j) \mid f_i \leq j \leq i; \text{ or } l_j \leq i < j; \}. \tag{7.8.2}$$

The envelope of a symmetric matrix is defined by the envelope of its lower (or upper) triangular part including the main diagonal.

For a variable band matrix it is convenient to use a storage scheme, in which every element a_{ij}, $(i,j) \in \mathrm{Env}\,(A)$ is stored. This means that zeros outside the envelope are exploited, but those inside the envelope are stored. This storage scheme is useful because of the important fact that only zeros inside the envelope will suffer fill-in during Gaussian elimination.

The proof of the following theorem is left as an exercise.

Theorem 7.8.2.

Assume that the triangular factors L and U of A exist. Then it holds that

$$\mathrm{Env}\,(L+U) = \mathrm{Env}\,(A),$$

i.e., the nonzero elements in L and U are contained in the envelope of A.

One of the main objectives of a sparse matrix data structure is to economize on storage while at the same time facilitating subsequent operations on the matrix. We now consider storage schemes that permit rapid execution of the elimination steps when solving general sparse linear systems. Usually the pattern of nonzero elements is very irregular, as illustrated in Figure 7.8.1. We first consider a storage scheme for a sparse vector x. The nonzero elements of x can be stored in **compressed form** in a vector xc with dimension nnz, where nnz is the number of nonzero elements in x. Further, we store in an integer vector ix the indices of the corresponding nonzero elements in xc. Hence, the sparse vector x is represented by the triple (nnz, xc, ix), where

$$xc_k = x_{ix(k)}, \quad k = 1 : nnz.$$

Example 7.8.2. The vector $x = (0,4,0,0,1,0,0,0,6,0)$ can be stored in compressed form as

$$xc = (1,4,6), \qquad ix = (5,2,9), \qquad nnz = 3$$

Operations on sparse vectors are simplified if *one* of the vectors is first **uncompressed**, i.e, stored in a full vector of dimension n. Clearly this operation can be done in time proportional to the number of nonzeros, and allows direct random access to specified element in the vector. Vector operations, e.g., adding a multiple a of a sparse vector x to an uncompressed sparse vector y, or computing the inner product $x^T y$ can then be performed in *constant time per nonzero element.* Assume, for example, that the vector x is held in compressed form as nnz pairs of values and indices, and y is held in a full length array. Then the operation $y := a * x + y$ may be expressed as

$$\text{for } k = 1 : nnz, \quad y(ix(k)) := a * xc(k) + y(ix(k));$$

A matrix can be stored as a collection of sparse row vectors, where each row vector is stored in AC in compressed form. The corresponding column subscripts are stored in the integer vector jx, i.e., the column subscript of the element ac_k is given in $jx(k)$. Finally, we need a third vector $ia(i)$, which gives the position in the array AC of the first element in the ith row of A. For example, the matrix in Example 7.8.1 is stored as

$$\begin{aligned} AC &= (a_{11}, a_{13} \mid a_{21}, a_{22}, a_{24} \mid a_{32}, a_{33}, a_{35} \mid a_{42}, a_{44} \mid a_{54}, a_{55}), \\ ia &= (1, 3, 6, 9, 11, 13), \\ jx &= (1, 3, 1, 2, 4, 2, 3, 5, 2, 4, 4, 5). \end{aligned}$$

Alternatively a similar scheme storing A as a collection of column vectors may be used. A drawback with these schemes is that it is expensive to insert new nonzero elements in the structure when fill-in occurs.

The components in each row need not be ordered; indeed there is often little advantage in ordering them. To access a nonzero a_{ij} there is no direct method of calculating the corresponding index in the vector AC. Some testing on the subscripts in jx has to be done. However, more usual is that a complete row of A has to be retrieved, and this can be done quite efficiently. This scheme can be used unchanged for storing the lower triangular part of a symmetric positive definite matrix.

If the matrix is stored as a collection of sparse row vectors, the entries in a particular column cannot be retrieved without a search of nearly all elements. This is needed, for instance, to find the rows which are involved in a stage of Gaussian elimination. A solution is then to store also the structure of the matrix as a set of column vectors. If a matrix is input in coordinate form it the conversion to this storage form requires a sorting of the elements, since they may be in arbitrary order. This can be done very efficiently in $O(n) + O(\tau)$ time, where τ is the number of nonzero elements in the factors and n is the order of the matrix.

Another way to avoid extensive searches in data structures is to use a linked list to store the nonzero elements. Associated with each element is a pointer to the location of the next element in its row and a pointer to the location of the next element in its column. If also pointer to the first nonzero in each row and column are stored there is a total overhead of integer storage of $2(\tau + n)$. This allows fill-ins to be added to the data structure with only two pointers being altered. Also the fill-in can be placed anywhere in storage so no reorderings are necessary. Disadvantages are that indirect addressing must be used when scanning a row or column and that the elements in one row or column can be scattered over a wide range of memory.

An important distinction is between **static** storage structures that remain fixed and **dynamic** structures that can accommodate fill-in. If only nonzeros are to be stored, the data structure for the factors must dynamically allocate space for the fill-in during the elimination. A static structure can be used when the location of the nonzeros in the factors can be predicted in advance, as is the case for the Cholesky factorization.

7.8.2 Graph Representation of Matrices.

In solving a sparse positive definite linear system $Ax = b$, an important step is to determine a permutation matrix P such that the matrix $P^T AP$ has a sparse

Cholesky factor R. Then a storage structure for R can be generated. This step should be done symbolically, using only the nonzero structure of A as input. To perform this task the structure of A can be represented by an **undirected graph** constructed as follows: Let $P_1, \ldots, P_n$ be n distinct points in the plane called **nodes**. For each $a_{ij} \neq 0$ in A we connect node P_i to node P_j by means of an **edge** between nodes i and j. The undirected graph of the symmetric matrix A is denoted by $G = (X, E)$, X is the set of nodes and edges E the set of edges (unordered pairs of nodes). The graph is said to be **ordered** (or labeled) if its nodes are labeled. The ordered graph $G(A) = (X, E)$, representing the structure of a symmetric matrix $A \in \mathbf{R}^{n\times n}$, consists of nodes labeled $1 : n$ and edges $(x_i, x_j) \in E$ if and only if $a_{ij} = a_{ji} \neq 0$.

Thus, there is a direct correspondence between nonzero elements in A and edges in the graph $G(A)$; see Figure 7.8.2.

$$A = \begin{pmatrix} \times & \times & & \times & \times & & \\ \times & \times & \times & & & & \\ & \times & \times & & & \times & \times \\ \times & & & \times & & & \\ \times & & & & \times & & \\ & & \times & & & \times & \\ & & \times & & & & \times \end{pmatrix}$$

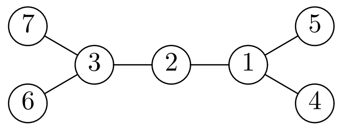

Figure 7.8.2. *The matrix A and its labeled graph.*

Definition 7.8.3.

The ordered undirected graph $G(A) = (X, E)$ of a symmetric matrix $A \in \mathbf{R}^{n\times n}$ consists of a set of n nodes *X together with a set E of* edges*, which are unordered pairs of nodes. The nodes are labeled $1, 2 : n$ where n, and nodes i and j are joined by an edge if and only if $a_{ij} = a_{ji} \neq 0$, $i \neq j$. We then say that the nodes i and j are* adjacent*. The number of edges incident to a node is called the* degree *of the node.*

The important observation is that for any permutation matrix $P \in \mathbf{R}^{n\times n}$ *the graphs $G(A)$ and $G(PAP^T)$ are the same except that the labeling of the nodes are different.* Hence, the unlabeled graph represents the structure of A without any particular ordering. Finding a good permutation for A is equivalent to finding a good labeling for its graph.

The adjacency set of x in G is defined by

$$\mathrm{Adj}_G(x) = \{y \in X \mid x \text{ and } y \text{ are adjacent}\}.$$

The number of nodes adjacent to x is denoted by $|\mathrm{Adj}_G(x)|$, and is called the **degree** of x. A **path** of length $l \geq 1$ between two nodes, u_1 and u_{l+1}, is an ordered set of distinct nodes $u_1, \dots, u_{l+1}$, such that

$$(u_i, u_{i+1}) \in E, \quad i = 1, \dots, l.$$

If there is such a chain of edges between two nodes, then they are said to be **connected**. If there is a path between every pair of distinct nodes, then the graph is connected. A disconnected graph consists of at least two separate connected subgraphs. ($\bar{G} = (\bar{X}, \bar{E})$ is a subgraph of $G = (X, E)$ if $\bar{X} \subset X$ and $\bar{E} \subset E$.) If $G = (X, E)$ is a connected graph, then $Y \subset X$ is called a separator if G becomes disconnected after the removal of the nodes Y.

A symmetric matrix A is said to be **reducible** if there is a permutation matrix P such that $P^T AP$ is block diagonal. It follows that the graph $G(P^T AP)$ is connected if and only if $G(A)$ is connected. It is then easy to prove that A is reducible if and only if its graph $G(A)$ is disconnected.

The structure of an unsymmetric matrix can similarly be represented by a **directed graph** $G = (X, E)$, where the edges now are *ordered pairs* of nodes. A directed graph is **strongly connected** if there is a path between every pair of distinct nodes along directed edges.

7.8.3 Nonzero Diagonal and Block Triangular Form

Definition 7.8.4.

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

$$P^T AP = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix}, \tag{7.8.3}$$

where B and C, are square submatrices, or if $n = 1$ and $A = 0$. Otherwise A is called **irreducible**.

The concept of a reducible matrix can also be illustrated using some elementary notions of graphs; see Sec. 7.8.2. The **directed graph** of a matrix A is constructed as follows: Let $P_1, \dots, P_n$ be n distinct points in the plane called **nodes**. For each $a_{ij} \neq 0$ in A we connect node P_i to node P_j by means of directed **edge** from node i to node j. It can be shown that a matrix A is irreducible if and only if its graph is **connected** in the following sense. Given any two distinct nodes P_i and P_j there exists a path $P_i = P_{i_1}, P_{i_2}, \dots, P_{i_p} = P_j$ along directed edges from P_i to P_j. Note that the graph of a matrix A is the same as the graph of $P^T AP$, where P is a permutation matrix; only the labeling of the node changes.

Assume that a matrix A is reducible to the form (7.8.3), where $B \in \mathbf{R}^{r\times r}$, $B \in \mathbf{R}^{s\times s}$ $(r + s = n)$. Then we have

$$\tilde{A} \begin{pmatrix} I_r \\ 0 \end{pmatrix} = \begin{pmatrix} I_r \\ 0 \end{pmatrix} B, \qquad (\, 0 \quad I_s \,)\, \tilde{A} = D\, (\, 0 \quad I_s \,),$$

that is, the first r unit vectors span a right invariant subspace, and the s last unit vectors span a left invariant subspace of $\tilde{A}$. It follows that the spectrum of A equals the union of the spectra of B and D.

If B and D are reducible they can be reduced in the same way. Continuing in this way until the diagonal blocks are irreducible we obtain the **block triangular form**

$$PAQ = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1,t} \\ & A_{22} & \dots & A_{2,t} \\ & & \ddots & \vdots \\ & & & A_{tt} \end{pmatrix} \tag{7.8.4}$$

with square diagonal blocks $A_{11}, \dots, A_{tt}$. The off-diagonal blocks are possibly nonzero matrices of appropriate dimensions.

Before performing a factorization of a sparse matrix it is often advantageous to perform some pre-processing. For an arbitrary square nonsingular matrix $A \in \mathbf{R}^{n\times n}$ there always is a row permutation P such that PA has nonzero elements on it diagonal. Further, there is a row permutation P and column permutation Q such that PAQ has a nonzero diagonal and the block triangular form (7.8.4). Using this structure a linear system $Ax = b$ or $PAQy = c$, where $y = Q^T x$, $c = Pb$, reduces to

$$A_{ii}y_i = c_i - \sum_{j=i+1}^{n} A_{ij}x_j, \quad j = n : -1 : 1. \tag{7.8.5}$$

Hence, we only need to factorize the diagonal blocks. This block back-substitution can lead to significant savings.

If we require that the diagonal blocks are irreducible, then the block triangular form (7.8.4) can be shown to be essentially unique. Any one block triangular form can be obtained from any other by applying row permutations that involve the rows of a single block row, column permutations that involve the columns of a single block column, and symmetric permutations that reorder the blocks. A square matrix

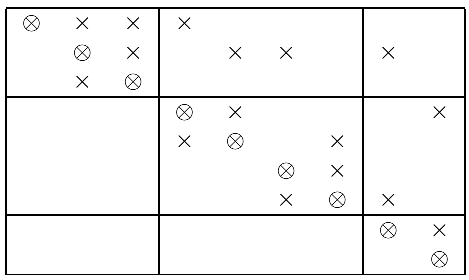

Figure 7.8.3. *The block triangular decomposition of A.*

which can be permuted to the form (7.8.4), with $t > 1$, is said to be **reducible**; otherwise it is called **irreducible**.

In the symmetric positive definite case a similar reduction to block upper triangular form can be considered, where $Q = P^T$. Some authors reserve the term reducible for this case, and use the terms bi-reducible and bi-irreducible for the general case.

An arbitrary rectangular matrix $A \in \mathbf{R}^{m \times n}$ has a block triangular form called the **Dulmage–Mendelsohn form**. If A is square and nonsingular this is the form (7.8.4). The general case is based on a canonical decomposition of bipartite graphs discovered by Dulmage and Mendelsohn. In the general case the first diagonal block may have more columns than rows, the last diagonal block more rows than column. All the other diagonal blocks are square and nonzero diagonal entries.This block form can be used for solving least squares problems by a method analogous to back-substitution.

The **bipartite graph** associated with A is denoted by $G(A) = \{R, C, E\}$, where $R = (r_1, \dots, r_m)$ is a set of vertices corresponding to the rows of A and $C = (c_1, \dots, c_m)$ a set of vertices corresponding to the columns of A. E is the set of edges, and $\{r_i, c_j\} \in E$ if and only if $a_{ij} \neq 0$. A **matching** in $G(A)$ is a subset of its edges with no common end points. In the matrix A this corresponds to a subset of nonzeros, no two of which belong to the same row or column. A **maximum matching** is a matching with a maximum number $r(A)$ of edges. The **structural rank** of A equals $r(A)$. Note that the mathematical rank is always less than or equal to its structural rank. For example, the matrix

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

has structural rank 2 but numerical rank 1.

For the case when A is a structurally nonsingular matrix there is a two-stage algorithm for permuting A to block upper triangular form. In the first stage a maximum matching in the bipartite graph $G(A)$ with row set R and column set C is found. In the second stage the block upper triangular form of each submatrix determined from the strongly connected components in the graph $G(A)$, with edges directed from columns to rows.

If A has structural rank n but is *numerically* rank deficient it will not be possible to factorize all the diagonal blocks in (7.8.4). In this case the block triangular structure given by the Dulmage–Mendelsohn form cannot be preserved, or some blocks may become severely ill-conditioned.

Note that for some applications, e.g., for matrices arising from discretizations of partial differential equations, it may be known a priori that the matrix is irreducible. In other applications the block triangular decomposition may be known in advance from the underlying physical structure. In both these cases the algorithm discussed above is not useful.

7.8.4 LU Factorization of Sparse Matrices

Hence, the first task in solving a sparse system is to order the rows and columns so that Gaussian elimination applied to the permuted matrix PAQ does not introduce too much fill-in. To find the *optimal* ordering, which minimizes the number of nonzero in L and U is unfortunately a hard problem. This is because the number of possible orderings of rows and columns is very large, $(n!)^2$, whereas solving a linear system only takes $O(n^3)$ operations. Fortunately, there are heuristic ordering algorithms which do a god job at approximately minimizing fill-in. These orderings usually also nearly minimize the arithmetic operation count.

Example 7.8.3.

The ordering of rows and columns in Gaussian elimination may greatly affect storage and number of arithmetic operations as shown by the following example. Let

$$A = \begin{pmatrix} \times & \times & \times & \dots & \times \\ \times & \times & & & \\ \times & & \times & & \\ \vdots & & & \ddots & \\ \times & & & & \times \end{pmatrix}, \qquad PAP^T = \begin{pmatrix} \times & & & & \times \\ & \ddots & & & \vdots \\ & & \times & & \times \\ & & & \times & \times \\ \times & \dots & \times & \times & \times \end{pmatrix}.$$

Matrices, or block matrices of this structure are called **arrowhead matrices** and occur in many applications.

If the $(1,1)$ element in A is chosen as the first pivot the fill in will be total and $2n^3/3$ operations required for the LU factorization. In PAP^T the orderings of rows and columns have been reversed. Now there is no fill-in except in the last step of, when pivots are chosen in natural order. Only about $4n$ flops are required to perform the factorization.

For variable-band matrices no fill-in occurs in L and U outside the envelope. One strategy therefore is to choose P and Q to approximately minimize the envelope of PAQ. (Note that the reordered matrix PAP^T in Example 7.8.3 has a small envelope but A has a full envelope!) For symmetric matrices the reverse Cuthill–McKee ordering is often used. In the unsymmetric case one can determine a reordering of the columns by applying this algorithm to the symmetric structure of $A + A^T$.

Perhaps surprisingly, the orderings that approximately minimize the total fill-in in LU factorization tend *not* to give a small bandwidth. Typically, the factors L and U instead have their nonzeros scattered throughout their triangular parts. A simple column reordering is to sort the columns by increasing column count, i.e. by the number of nonzeros in each column. This can often give a substantial reduction of the fill-in in Gaussian elimination. In Figure 7.8.5 we show the LU factorization of the matrix W reordered after column count and its LU factors. The number of nonzeros in L and U now are 6604, which is a substantial reduction.

An ordering that often performs even better is the so-called column minimum degree ordering shown in Figure 7.8.6. The LU factors of the reordered matrix now containing 5904 nonzeros. This column ordering is obtained by using the symmetric minimum degree described in the next section on the matrix W^TW. MATLABuses an implementation of this ordering algorithm that does not actually form the matrix W^TW. For the origin and details of this code we refer to Gilbert, Moler, and Schreiber [160, 1992].

For unsymmetric systems some kind of stability check on the pivot elements must be performed during the numerical factorization. Therefore, the storage structure for L and U cannot be predicted from the structure of A alone, but must be

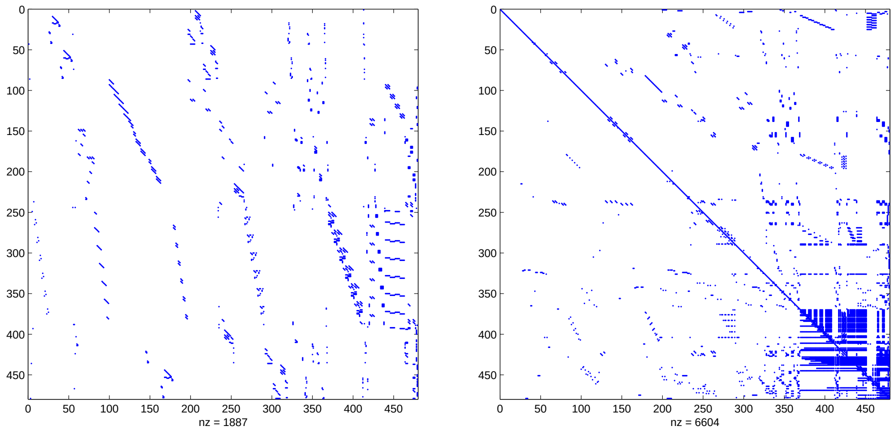

Figure 7.8.4. *Nonzero pattern of a matrix and its LU factors after reordering by increasing column count.*

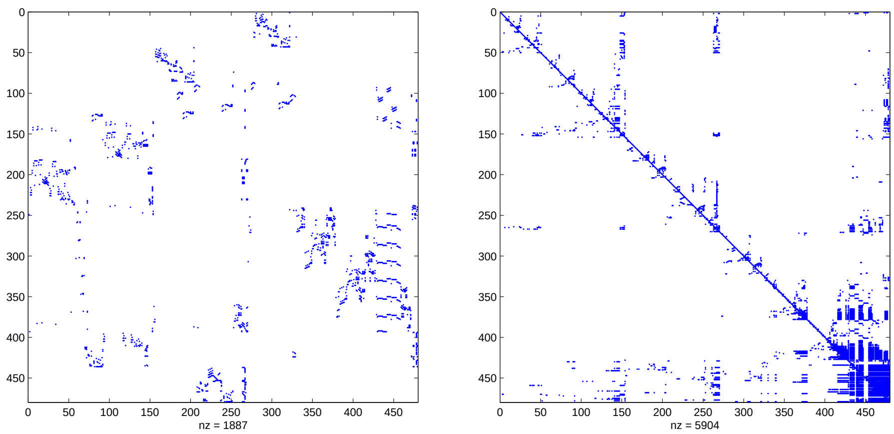

Figure 7.8.5. *Nonzero pattern of a matrix and its LU factors after minimum degree ordering.*

determined dynamically during the numerical elimination phase.

MATLABuses the column sweep method with partial pivoting due to Gilbert and Peierls [161, 1988] for computing the LU factorization a column of L and U at a time. In this the basic operation is to solve a series of sparse triangular systems involving the already computed part of L. The column-oriented storage structure is set up dynamically as the factorization progresses. Note that the size of storage needed can not be predicted in advance. The total time for this LU factorization algorithm can be shown to be proportional to the number of arithmetic operations plus the size of the result.

Other sparse LU algorithms reorders both rows and columns before the numerical factorization. One of the most used ordering algorithm is the **Markowitz algorithm**. To motivate this suppose that Gaussian elimination has proceeded through k stages and let $A^{(k)}$ be the remaining active submatrix. Denote by r_i the number of nonzero elements in the ith row and c_j is the number of nonzero elements in the jth column of $A^{(k)}$. In the Markowitz algorithm one performs a row and column interchange so that the product

$$(r_i - 1)(c_j - 1),$$

is minimized. (Some rules for tie-breaking are needed also.) This is equivalent to a *local minimization* of the fill-in at the next stage, assuming that all entries modified were zero beforehand. This choice also minimizes the number of multiplications required for this stage.

With such an unsymmetric reordering there is a conflict with ordering for sparsity and for stability. The ordering for sparsity may not give pivotal elements which are acceptable from the point of numerical stability. Usually a **threshold pivoting** scheme is used to minimize the reorderings. This means that the chosen pivot is restricted by an inequality

$$|a_{ij}^{(k)}| \geq \tau \max_r |a_{rj}^{(k)}|, \tag{7.8.6}$$

where τ, $0 < \tau \leq 1$, is a predetermined threshold value. A value of $\tau = 0.1$ is usually recommended as a good compromise between sparsity and stability. (Note that the usual partial pivoting strategy is obtained for $\tau = 1$.) The condition (7.8.6) ensures that in any column that is modified in an elimination step the maximum element increases in size by at most a factor of $(1 + 1/\tau)$. Note that a column is only modified if the pivotal row has a nonzero element in that column. The total number of times a particular column is modified during the complete elimination is often quite small if the matrix is sparse. Furthermore, it is possible to monitor stability by, for example, computing the relative backward error, see Sec. 7.6.4.

7.8.5 Cholesky Factorization of Sparse Matrices

If A is symmetric and positive definite, then the Cholesky factorization is numerically stable for any choice of pivots along the diagonal. We need only consider symmetric permutations PAP^T, where P can be chosen with regard only to sparsity. This, leads to a substantial increase in the efficiency of the sparse Cholesky algorithm since a static storage structure can be used.

We remark that the structure predicted for R from that of P^TAP by performing the Cholesky factor symbolically, is such that $R + R^T$ will be at least as full as PAP^T. In Figure 7.8.6 we show the nonzero pattern of the matrix $S = WW^T$, where W is the matrix west0479, and its Cholesky factor.

The Cholesky factorization of a sparse symmetric positive definite matrix A can be divided into four separate steps:

1. Determine a permutation P such that P^TAP has a sparse Cholesky factor L.

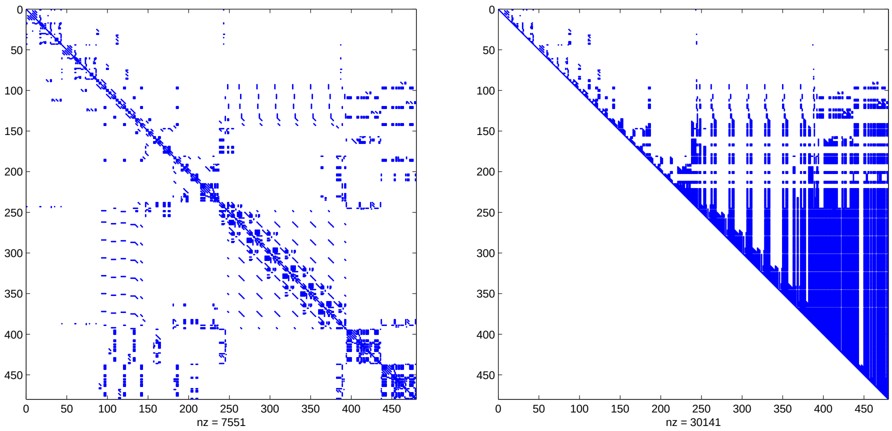

Figure 7.8.6. *Nonzero pattern of a matrix and its Cholesky factor.*

2. Perform a symbolic Cholesky factorization of PAP^T and generate a storage structure for R.

3. Form P^TAP and store in data structure for R.

4. Compute numerically the Cholesky factor R such that $P^TAP = R^TR$.

We stress that steps 1 and 2 are done symbolically, only working on the structure of A. The numerical computations take place in steps 3 and 4 where a static storage scheme can be used.

Example 7.8.4.
To illustrate the symbolic factorization we use the sparse symmetric matrix A with Cholesky factor R

$$A = \begin{pmatrix} \times & \times & & \times & \times & & \\ \times & \times & \times & & & & \\ & \times & \times & & & \times & \times \\ \times & & & \times & & & \\ \times & & & & \times & & \\ & & \times & & & \times & \\ & & \times & & & & \times \end{pmatrix}, \qquad R = \begin{pmatrix} \times & \times & & \times & \times & & \\ & \times & \times & + & + & & \\ & & \times & + & + & \times & \times \\ & & & \times & + & & \\ & & & & \times & & \\ & & & & & \times & \\ & & & & & & \times \end{pmatrix},$$

where $\times$ and $+$ denote a nonzero element. We show only the nonzero structure of A and R, not any numerical values. The five elements marked $+$ are the fill-in that occur in the Cholesky factorization.

Graph theory provides a powerful tool for the analysis and implementation of ordering algorithms. In the following we restrict ourselves to the case of a symmetric structure. Below is the ordered graph $G(A)$, of the matrix in Example 7.8.4.

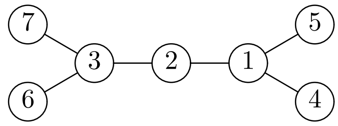

Figure 7.8.7. *The labeled graph of the matrix A.*

Example 7.8.5.

The labeled graph suggest that row and columns of the matrix in Example 7.8.5 is rearranged in order $4, 5, 7, 6, 3, 1, 2$. With this ordering the Cholesky factor of the matrix PAP^T will have no fill-in!

$$PAP^T = \begin{pmatrix} \times & & & & & \times & \\ & \times & & & & \times & \\ & & \times & & \times & & \\ & & & \times & \times & & \\ & & \times & \times & \times & & \times \\ \times & \times & & & & \times & \times \\ & & & & \times & \times & \times \end{pmatrix}, \qquad R = \begin{pmatrix} \times & & & & & \times & \\ & \times & & & & \times & \\ & & \times & & \times & & \\ & & & \times & \times & & \\ & & & & \times & & \times \\ & & & & & \times & \times \\ & & & & & & \times \end{pmatrix},$$

From the graph $G(A^TA)$ the structure of the Cholesky factor R can be predicted by using a graph model of Gaussian elimination. positive definite matrix $A \in \mathbf{R}^{n\times n}$ is due to Parter. The fill-in under the factorization process can be analyzed by considering a sequence of **elimination graphs** that can be recursively formed as follows. We take $G_0 = G(A)$, and form G_i from $G_{(i-1)}$ by removing the node i and its incident edges and adding fill edges. The fill edges in eliminating node v in the graph G are

$$\{(j, k) \mid (j, k) \in \mathrm{Adj}_G(v), j \neq k\}.$$

Thus, the fill edges correspond to the set of edges required to make the adjacent nodes of v pairwise adjacent. The filled graph $G_F(A)$ of A is a graph with n vertices and edges corresponding to all the elimination graphs G_i, $i = 0, \dots, n-1$. The filled graph bounds the structure of the Cholesky factor R,

$$G(R^T + R) \subset G_F(A). \tag{7.8.7}$$

Under a no-cancellation assumption, the relation (7.8.7) holds with equality.

The following characterization of the filled graph describes how it can be computed directly from $G(A)$.

Theorem 7.8.5. *Let $G(A) = (X, E)$ be the undirected graph of A. Then (x_i, x_j) is an edge of the filled graph $G_F(A)$ if and only if $(x_i, x_j) \in E$, or there is a path in $G(A)$ from node i to node j passing only through nodes with numbers less than $\min(i, j)$.*

Consider the structure of the Cholesky factor $R = (r_{ij}$. For each row $i \leq n$

we define $\gamma(i)$ by

$$\gamma(i) = \min\{j > i \mid r_{ij} \neq 0\}, \tag{7.8.8}$$

that is $\gamma(i)$ is the column subscript of the first off-diagonal nonzero element in row i of R. If row i has no off-diagonal nonzero, then $\gamma(i) = i$. Clearly $\gamma(n) = n$. The quantities $\gamma(i)$, $i = 1 : n$ can be used to represent the structure of the sparse Cholesky factor R. For the matrix R in Example 7.8.4 we have

i	1	2	3	4	5	6	7
$\gamma(i)$	2	3	6	4	5	6	7

We now introduce the **elimination tree** corresponding to the structure of the Cholesky factor. The tree has n nodes, labeled form 1 to n. For each i if $\gamma(i) > j$, the node $\gamma(i)$ is the **parent** of node i in the elimination tree and node j is one of possible several **child** nodes of node $\gamma(i)$. If the matrix is irreducible then n is the only node with $\gamma(n) = n$ and is the **root** of the tree. There is exactly one path from node i to the root. If node j lies on the path from node i to the root, then node j is an ancestor to node i and node j is a descendant of node i.

The most widely used algorithm for envelope reduction for symmetric matrices is the **reverse Cuthill–McKee ordering**. This works on the graph $G(A)$ as follows:

1. Determine a starting node and label this 1.
2. For $i = 1 : n - 1$ find all unnumbered nodes adjacent to the node with label i, and number them in increasing order of degree.
3. The reverse ordering is obtained by reversing the ordering just determined.

The reversal of the Cuthill–McKee ordering in step 3 was suggested by Alan George, who noticed that it often was much superior to the original ordering produced by steps 1 and 2 above. In order for the algorithm to perform well it is necessary to choose a good starting node; see George and Liu [153, Section 4.3.3]. In Figure 7.8.8 we show the structure of the matrix from Figure 7.8.1 and its Cholesky factor after reverse Cuthill–McKee reordering. The number of nonzero elements in the Cholesky factor is $23,866$.

As for unsymmetric matrices, the orderings that approximately minimize the total fill-in in the Cholesky factor tend to have their nonzeros scattered throughout the matrix. For some problems, such orderings can reduce fill-in by one or more orders of magnitude over the corresponding minimum bandwidth ordering.

In the symmetric case $r_i = c_i$. The Markowitz ordering is then equivalent to minimizing r_i, and the resulting algorithm is called the **minimum-degree algorithm**. The minimum degree ordering can be determined using a graph model of the Cholesky factorization. At the same time the nonzero structure of the Cholesky factor R can be determined and a storage structure for R generated. The minimum-degree algorithm has been subject to an extensive development and very efficient implementations now exist. For details we refer to George and Liu [153, Chapter 5] and [154, 1989].

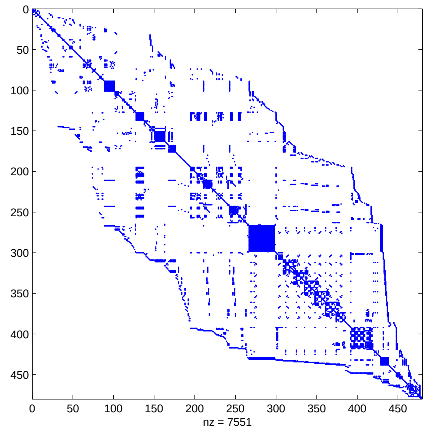

Figure 7.8.8. *Matrix and its Cholesky factor after reverse Cuthill–McKee reordering.*

Figure 7.8.9 shows the structure of the matrix from Figure 7.8.1 and its Cholesky factor after minimum-degree reordering. The number of nonzero elements in the Cholesky factor is reduced to 12,064. For nested dissection orderings, see George and Liu [153, Chapter 8].

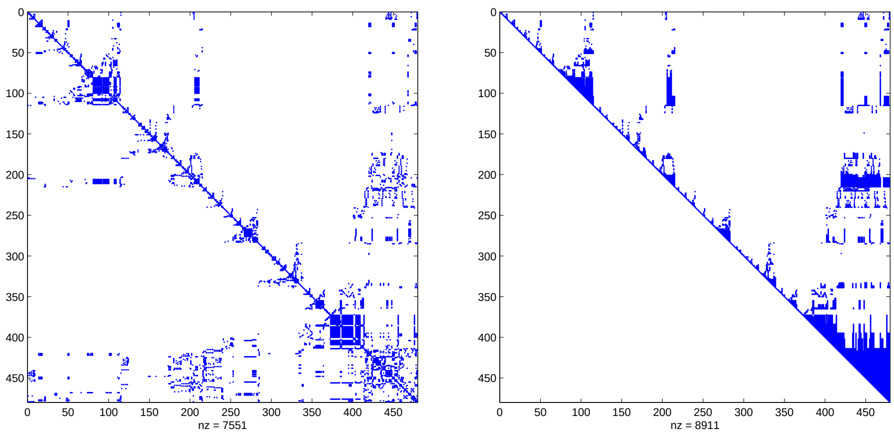

Figure 7.8.9. *Matrix and its Cholesky factor after minimum-degree reordering.*

Review Questions

8.1 Describe the coordinate form of storing a sparse matrix. Why is this not suitable for performing the numerical LU factorization?

8.2 Give an example of a sparse matrix A, which suffers extreme fill-in in Gaussian elimination..

8.3 Describe the Markowitz algorithm for ordering rows and columns of a non-symmetric matrix before factorization.

8.4 Describe threshold pivoting. Why is this used instead of partial pivoting in some schemes for LU factorization?

8.5 What does the reverse Cuthill–McKee ordering minimize?

Problems

8.1 Let $A, B \in \mathbf{R}^{n\times n}$ be sparse matrices. Show that the number of multiplications to compute the product $C = AB$ is $\sum_{i=1}^{n} \eta_i\theta_i$, where η_i denotes the number of nonzero elements in the ith *column* of A and θ_i the number of nonzeros in the ith *row* of B.

Hint: Use the outer product formulation $C = \sum_{i=1}^{n} a_{.i} b_i^T$.

8.2 (a) It is often required to add a multiple a of a sparse vector x to another sparse vector y. Show that if the vector x is held in coordinate form as nx pairs of values and indices, and y is held in a full length array this operation may be expressed thus:

$$\begin{aligned}&\text{for } k = 1 : nx\\ &\qquad y(\text{index}(k)) = a * x(k) + y(\text{index}(k));\end{aligned}$$

(b) Give an efficient algorithm for computing the inner product of two compressed vectors.

8.3 Consider a matrix with the symmetric structure

$$A = \begin{pmatrix} \times & & \times & & & \\ & \times & & \times & \times & \\ \times & & \times & \times & & \\ & \times & \times & & & \times \\ & \times & & & \times & \times \\ & & & \times & \times & \times \end{pmatrix}.$$

(a) What is the envelope of A? Where will fill-in occur during Gaussian elimination?

(b) Draw the undirected graph G, which represents the sparsity structure of A.

7.9 Structured Systems

The coefficient matrices in systems of linear equations arising from signal processing, control theory and linear prediction often have some special structure that can be taken advantage of. Several classes of such structured systems can be solved by fast methods in $O(n^2)$ operations, or by super-fast methods even in $O(n \log n)$ operations rather than $O(n^3)$ otherwise required by Gaussian elimination. This has important implications for many problems in signal restoration, acoustics, seismic exploration and many other application areas. Since the numerical stability properties of super-fast methods are generally either bad or unknown we consider only fast methods in the following.

Semiseparable matrices are treated in [372, 2007].

7.9.1 Toeplitz and Hankel Matrices

Note: The following subsections are not yet complete and will be amended.

A **Toeplitz matrix** T is a matrix whose entries are constant down each diagonal; $T = (t_{i-j})_{1 \le i,j \le n}$,

$$T_n = \begin{pmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_{-1} & t_0 & \dots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{-n+1} & t_{-n+2} & \dots & t_0 \end{pmatrix} \in \mathbf{R}^{n \times n}, \tag{7.9.1}$$

and is defined by the $2n-1$ values of $t_{-n+1}, \dots, t_0, \dots, t_{n-1}$. Toeplitz matrices are fundamental in signal processing and time series analysis. The structure reflects the invariance in time or in space. They also arise from partail differential equations with constant coefficients and from discretizations of integral equations with convolution kernels.

Toeplitz linear systems arising in applications are often large, and dimensions of $10,000$ or more are not uncommon. The original matrix T only requires $2n-1$ storage. However, the inverse of a Toeplitz matrix is not Toeplitz, and if standard factorization methods are used, at least $n(n+1)/2$ storage is needed. Special algorithms which exploit the Toeplitz structure are much faster and use less storage. Methods based on the **Levinson–Durbin recursion** (see [263, 118]) require about $2n^2$ flops. An even faster direct method has been developed for symmetric positive definite Toeplitz systems by Ammar and Gragg [4].

We now describe the Levinson–Durbin recursion for solving the linear system $T_n x = y$. We assume that all principal minors of T_n are nonsingular. Two sets of vectors are generated, called the forward vectors f_k and the backward vectors b_k. These vectors are of length k and satisfy as solutions of the linear systems

$$T_k f_k = e_1, \qquad T_k b_k = e_k, \quad k = 1 : n, \tag{7.9.2}$$

where e_1 and e_k are unit vectors of length k. The first forward and backward vectors are simply

$$f_1 = b_1 = 1/t_0.$$

Now assume that the vectors f_{k-1} and b_{k-1} have been determined. Then, since T_{k-1} is the leading principal submatrix of T_k, we have

$$T_k \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} = \begin{pmatrix} e_1 \\ \epsilon_k \end{pmatrix}, \quad \epsilon_k = T_k e_k \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix}. \tag{7.9.3}$$

Similarly, for the backward vectors we have the recursion.

$$T_k \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} = \begin{pmatrix} \delta_k \\ e_{k-1} \end{pmatrix}, \quad \delta_k = T_k^T e_k \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix}. \tag{7.9.4}$$

If we now take a linear combination of these two equations

$$T_k \left(\alpha \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} \right) = \alpha \begin{pmatrix} e_1 \\ \epsilon_k \end{pmatrix} + \beta \begin{pmatrix} \delta_k \\ e_{k-1} \end{pmatrix}.$$

and α and β are chosen so that the right-hand side becomes e_1, this will give us the forward vector f_k. Similarly, if α and β are chosen so that the right-hand side becomes e_k, this will give us the vector b_k. Denote these values by α_f, β_f , and α_b β_b, respectively. Disregarding the zero elements in the right hand side vectors, it follows that these values satisfy the 2×2 linear system

$$\begin{pmatrix} 1 & \delta_k \\ \epsilon_k & 1 \end{pmatrix} \begin{pmatrix} \alpha_f & \alpha_b \\ \beta_f & \beta_b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{7.9.5}$$

If $\delta_k \delta_k \neq 1$ this system is nonsingular and then

$$\begin{pmatrix} \alpha_f & \alpha_b \\ \beta_f & \beta_b \end{pmatrix} = \begin{pmatrix} 1 & \delta_k \\ \epsilon_k & 1 \end{pmatrix}^{-1} = \frac{1}{1 - \epsilon_k \delta_k} \begin{pmatrix} 1 & -\delta_k \\ -\epsilon_k & 1 \end{pmatrix},$$

which allows us to compute the new vectors

$$\begin{aligned} f_k &= \frac{1}{1 - \epsilon_k \delta_k} \left(\begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} - \delta_k \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} \right), \\ b_k &= \frac{1}{1 - \epsilon_k \delta_k} \left(\begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} - \epsilon_k \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} \right). \end{aligned}$$

The cost of this recursion step is about $8k$ flops.

The solution to the linear system $T_n x = y$ van be built up as follows. Assume the vector $x^{(k-1)} \in \mathbf{R}^{k-1}$ satisfies the first $k-1$ equations and write

$$T_k \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{k-1} \\ \eta_k \end{pmatrix}, \quad \eta_k = T_k^T e_k \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix}. \tag{7.9.6}$$

Then the backward vector b_k can be used to modify the last element in the right-hand side. The solution can be built up using the recusrion

$$x^{(1)} = y_1/t_0, \qquad x^{(k)} = \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} + (y_k - \eta_k) b_k, \quad k = 2 : n.$$

At any stage Only storage for the three vectors f_k, b_k, and $x^{(k)}$ are needed.

When the Toeplitz matrix is symmetric there are important simplifications. Then from (7.9.4)–(7.9.4) it follows that the backward and forward vectors are the row-reversals of each other, i.e.

$$b_k = J_k f_k, \quad J_k = (e_k, e_{k-1}, \dots, e_1).$$

Further $\epsilon_k = \delta_k$, so the auxiliary 2×2 subsystems (7.9.5) are symmetric. Taking this into account roughly halves the operation count and storage requirement.

It is imporant to note that even if the Toeplitz matrix T_n is nonsingular, its principal minors can be singular. An example is the symmetric indefinite matrix

$$T_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

for which the principal minor T_2 is singular. Likewise, if T_n is well conditioned, its principal minors can be ill-conditioned. Many of the fast algorithms for solving Toeplitz systems can only be proved to be stable for the symmetric positive definite case. The stability of the Levinson–Durbin algorithm has been analyzed by Cybenko [83]. A more general discusssion of stability of methods for solving Toeplitz systems is given by Bunch [54].

Some "superfast" algorithms for solving Toeplitz systems have been suggested. These are based on the FFT and use only $O(n \log^2 n)$. These are in general not stable except for the positive definite case. Efficient iterative methods for solving symmetric positive definite Toeplitz systems using the conjugate gradient methods preconditioned with a circulant matrix; see T. Chan [61], R. Chan and Strang [62], and Chan et al. [60].

A **Hankel matrix** is a matrix whose elements are constant along every antidiagonal, i.e.,$H = (h_{i+j-2})_{1 \le i,j \le n}$

$$H = \begin{pmatrix} h_0 & h_1 & \dots & h_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-2} & h_{n-1} & \dots & h_{2n-3} \\ h_{n-1} & h_n & \dots & h_{2n-2} \end{pmatrix} \in \mathbf{R}^{n \times n}.$$

Reversing the rows (or columns) of a Hankel matrix we get a Toeplitz matrix, i.e. HJ and JH are Toeplitz matrices, where $J = (e_n, e_{n-1}, \dots, e_1)$. Hence, methods developed for solving bf 5 Toeplitz systems apply also to Hankel systems.

7.9.2 Cauchy-Like Matrices

A **Cauchy matrix** is a matrix of the following form:

$$C = \left(\frac{1}{y_i - z_j} \right)_{1 \le i,j \le n}, \qquad a_i, b_j \in \mathbf{R}^p. \tag{7.9.7}$$

where we assume that $y_i \neq z_j$ for $1 \le i, j \le n$.

Example 7.9.1. Consider the problem of finding the coefficients of a rational function

$$r(x) = \sum_{j=1}^{n} a_j \frac{1}{x - y_j},$$

which satisfies the interpolation conditions $r(x_i) = f_i$, $i = 1, \ldots, n$. With $a = (a_1, \ldots, a_n)$, $f = (f_1, \ldots, f_n)$ this leads to the linear system $Ca = f$, where C is the Cauchy matrix in (7.9.7).

Cauchy gave in 1841 the following explicit expression for the determinant

$$\det(C) = \frac{\displaystyle\prod_{1 \le i < j \le n} (y_j - y_i)(z_j - z_i)}{\displaystyle\prod_{1 \le i \le j \le n} (y_j + z_i)}.$$

We note that any row or column permutation of a Cauchy matrix is again a Cauchy matrix. This property allows fast and stable version of Gaussian to be developed for Cauchy systems.

Many of these methods also apply in the more general case of **Loewner matrices** of the form

$$C = \left(\frac{a_i^T b_j}{y_i - z_j} \right)_{1 \le i,j \le n}, \qquad a_i, b_j \in \mathbf{R}^p. \tag{7.9.8}$$

Example 7.9.2. The most famous example of a Cauchy matrix is the **Hilbert matrix**, which is obtained by taking $y_i = z_i = i - 1/2$:

$$H_n \in \mathbf{R}^{n \times n}, \quad h_{ij} = \frac{1}{i + j - 1}.$$

For example,

$$H_4 = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{pmatrix}.$$

The Hilbert matrix is symmetric and positive definite Hankel matrix. It is also **totally positive** . The inverse of H_n is known explicitly and has integer elements. Hilbert matrices of high order are known to be very ill-conditioned; for large n it holds that $\kappa_2(H_n) \sim e^{3.5n}$.

7.9.3 Vandermonde systems

In Chapter 4 the problem of interpolating given function values $f(\alpha_i)$, $i = 1, \ldots, n$ at distinct points α_i with a polynomial of degree $\le n - 1$ was shown to lead to a linear system of equations with matrix $M = [p_j(\alpha_i)]_{i,j=1}^m$. In the case of the power basis $p_j(z) = z^{j-1}$, the matrix M equals V^T, where V is the **Vandermonde matrix**

$$V = [\alpha_j^{i-1}]_{i,j=1}^n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{pmatrix}. \tag{7.9.9}$$

Hence, the unique polynomial $P(z)$ satisfying the interpolating conditions $P(\alpha_i) = f_i$, $i = 1, \dots, n$ is given by

$$P(z) = (1, z, \dots, z^{n-1})a,$$

where a is the solution of the dual Vandermonde system.

$$V^T a = f \tag{7.9.10}$$

An efficient algorithm for solving primal and dual Vandermonde systems is the **Björck–Pereyra algorithm** described in Volume I, Sec. 3.5.4. It is related to Newton's interpolation method for determining the polynomial $P(x)$, and solves primal and dual Vandermonde systems in $\frac{1}{2}n(n+1)(3A+2M)$ operations, where A and M denotes one floating-point addition and multiplication, respectively. No extra storage is needed.

Review Questions

9.1

9.2

Problems

9.1 (a) Show that the inverse of a Toeplitz matrix is persymmetric.

(b) Show that if a matrix M is both symmetric and persymmetric all elements are defined by those in a wedge, as illustrated below for $n = 6$.

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \\ & & \times & \times & & \\ \end{pmatrix}.$$

Show that for n even the number of elements needed to define M equals $n^2/4 + n/2$.

9.2

Notes and Further References

The literature on linear algebra is very extensive. For a theoretical treatise a classical source is Gantmacher [147, 148, 1959]. Several nonstandard topics are covered in depth in two excellent volumes by Horn and Johnson [215, 1985] and [216, 1991].

An interesting survey of classical numerical methods in linear algebra can be found in Faddeev and Faddeeva [125, 1963], but many of the methods treated are now dated. A compact, lucid and modern presentation is given in Householder [220, 1964]. Bellman [27, 1960] is an original and readable complementary text.

An up to date and indispensable book for of anyone interested in computational linear algebra is Golub and Van Loan [184, 1996]. The book by Higham [211, 2002] is a wonderful and useful source book for information about the accuracy and stability of algorithms in numerical linear algebra. Other excellent textbooks on matrix computation include Stewart two volumes [346, 1998]on basic decompositions and [346, 1998] on eigensystems. For results on on perturbation theory and related topics a very complete reference book is Stewart and Sun [348, 1990]. In particular, an elegant treatise on norms and metrics is found in [348, Chapter II]. Another good text is Watkins [380] .

The Matrix Computation Toolbox by N. J. Higham is a collection of Matlab m-files containing functions for constructing test matrices, computing matrix factorizations, vizualizing matrices and other miscellaneous functions is available from the Web; see [211, Appendix D].

Section 7.2

Although the history of Gaussian elimination goes back at least to Chinese mathematicians about 250 B.C., there was no practical experience of solving large linear systems until the advent of computers in the 1940s. Gaussian elimination was the first numerical algorithm to be subjected to a rounding error analysis. In 1946 there was a mood of pessimism about the stability of Gaussian elimination. Hotelling [217, 1943] had produced bounds showing that the error in the solution would be proportional to 4^n, which suggested that it would be impossible to solve even systems of modest order. A few years later J. von Neumann and H. H. Goldstein [290, 1947] (reprinted in [353, pp. 479–557]) published more relevant error bounds. In 1948 A. M. Turing wrote a remarkable paper [364, 1948], where he formulated the LU factorization and introduced matrix condition numbers. The more or less final form of error analysis of Gaussian elimination was given by J. H. Wilkinson [386, 1961]. For a more detailed historical perspective of Gaussian elimination we refer to N. J. Higham [211, Sec. 9.13].

It is still an open (difficult!) question what the minimum exponent ω is, such that matrix multiplication can be done in $O(n^\omega)$ operations. The fastest known algorithm devised in 1987 by Don Coppersmith and Shmuel Winograd [75, 1990] has $\omega < 2.376$. Many believe that an optimal algorithm can be found which reduces the number to essentially n^2. For a review of recent efforts in this direction using group theory, see Robinson [321, 1985]. (Note that for many of the theoretically "fast" methods large constants are hidden in the O notation.)

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

The Schur complement was so named by Haynsworth in 1968 and appeared in a paper by Schur in 1917. The Banachiewicz inversion formula was introduced in 1937. For a historical account of the Schur complement see [398, 2005].

Rook pivoting for nonsymmetric matrices was introduced by Neal and Poole in [287, 1992]; see also [288, 2000]. Related pivoting strategies for symmetric indefinite matrices were introduced earlier by Fletcher [132, 1975].

Section 7.3

The idea of doing only half the elimination for symmetric systems, while preserving symmetry is probably due to Gauss, who first sketched his elimination algorithm in 1809. The Cholesky method is named after Andre-Louis Cholesky, who was a French military officer. He devised his method to solve symmetric, positive definite system arising in a geodetic survey in Crete and North Africa just before World War I.

Section 7.6

Bauer [26, 1966] in 1966 was the first to study componentwise perturbation theory. This did not catch on in English publications until Skeel took it up in two papers [334, 1979], and [335, 1980].

Section 7.8

Direct methods for sparse symmetric positive definite systems are covered in George and Liu [153, 1981], while a more general treatise given in Duff et al. [116, 1986]. A good survey is given by Duff [115]. The book by Davis [88, 2006] demonstrates a wide range of sparse matrix algorithms in a concise code. It also gives an overview of available software with links to high performance sparse LU, Cholesky, and QR factorizations codes: available at `http://www.cise.ufl.edu/research/sparse/codes`.

Section 7.9

N. Wiener and A. N. Kolmogorov independently analyzed the continuous case for linear filtering in 1941. In 1947 N. Levinson analyzed the discrete case, which yields a Toeplitz system of linear equations to solve. He gave an $O(n^2)$ algorithm, which was later improved by Durbin and others. Trench [362] has given an $O(n^2)$ algorithm for computing the inverse of a Toeplitz matrtix.

Chapter 8

Linear Least Squares Problems

Of all the principles that can be proposed, I think there is none more general, more exact, and more easy of application than that which consists of rendering the sum of squares of the errors a minimum.
—Adrien Maria Legendre, Nouvelles méthodes pour la détermination des orbites des cométes. Paris 1805

8.1 Preliminaries

8.1.1 The Least Squares Principle

A fundamental task in scientific computing is to estimate parameters in a mathematical model from collected data which are subject to errors. The influence of the errors can be reduced by using a greater number of data than the number of unknowns. If the model is linear, the resulting problem is then to "solve" an in general inconsistent linear system $Ax = b$, where $A \in \mathbf{R}^{m\times n}$ and $m \geq n$. In other words, we want to find a vector $x \in \mathbf{R}^n$ such that Ax is in some sense the "best" approximation to the known vector $b \in \mathbf{R}^m$.

There are many possible ways of defining the "best" solution to an inconsistent linear system. A choice which can often be motivated for statistical reasons (see Theorem 8.1.7) and also leads to a simple computational problem is the following: Let x be a vector which minimizes the Euclidian length of the **residual vector** $r = b - Ax$; i.e., a solution to the minimization problem

$$\min_x \|Ax - b\|_2, \tag{8.1.1}$$

where $\|\cdot\|_2$ denotes the Euclidian vector norm. Note that this problem is equivalent to minimizing the sum of squares of the residuals $\sum_{i=1}^m r_i^2$. Hence, we call (8.1.1) a **linear least squares problem** and any minimizer x a **least squares solution** of the system $Ax = b$.

Example 8.1.1. Consider a model described by a scalar function $y(t) = f(x,t)$, where $x \in \mathbf{R}^n$ is a parameter vector to be determined from measurements (y_i, t_i), $i = 1 : m$, $m > n$. In particular, let $f(x,t)$ be *linear* in x,

$$f(x,t) = \sum_{j=1}^{n} x_j \phi_j(t).$$

Then the equations $y_i = \sum_{j=1}^{n} x_j \phi_j(t_i)$, $i = 1 : m$ form an overdetermined system, which can be written in matrix form $Ax = b$, where $a_{ij} = \phi_j(t_i)$, and $b_i = y_i$.

We shall see that a least squares solution x is characterized by $r \perp \mathcal{R}(A)$, where $\mathcal{R}(A)$ the range space of A. The residual vector r is always uniquely determined and the solution x is unique if and only if $\operatorname{rank}(A) = n$, i.e., when A has linearly independent columns. If $\operatorname{rank}(A) < n$, we seek the unique least squares solution of minimum Euclidean norm.

When there are more variables than needed to match the observed data, then we have an **underdetermined problem**. In this case we can seek the **minimum norm solution** $y \in \mathbb{R}^m$ of a linear system, i.e. solve

$$\min \|y\|_2, \qquad A^T y = c, \tag{8.1.2}$$

where $c \in \mathbb{R}^n$ and $A^T y = c$ is assumed to be consistent.

We now show a necessary condition for a vector x to minimize $\|b - Ax\|_2$.

Theorem 8.1.1.

Given the matrix $A \in \mathbf{R}^{m\times n}$ and a vector $b \in \mathbf{R}^m$. The vector x minimizes $\|b - Ax\|_2$ if and only if the residual vector $r = b - Ax$ is orthogonal to $\mathcal{R}(A)$, i.e. $A^T(b - Ax) = 0$, or equivalently

$$A^T A x = A^T b \tag{8.1.3}$$

which are the **normal equations**.

Proof. Let x be a vector for which $A^T(b - Ax) = 0$. Then for any $y \in \mathbf{R}^n$ $b - Ay = (b - Ax) + A(x - y)$. Squaring this and using (8.1.3) we obtain

$$\|b - Ay\|_2^2 = \|b - Ax\|_2^2 + \|A(x - y)\|_2^2 \geq \|b - Ax\|_2^2.$$

On the other hand assume that $A^T(b - Ax) = z \neq 0$. Then if $x - y = -\epsilon z$ we have for sufficiently small $\epsilon \neq 0$,

$$\|b - Ay\|_2^2 = \|b - Ax\|_2^2 - 2\epsilon\|z\|_2^2 + \epsilon^2\|Az\|_2^2 < \|b - Ax\|_2^2$$

so x does not minimize $\|b - Ax\|_2$. $\square$

The matrix $A^T A \in \mathbf{R}^{n\times n}$ is symmetric and since

$$x^T A^T A x = \|Ax\|_2^2 \geq 0,$$

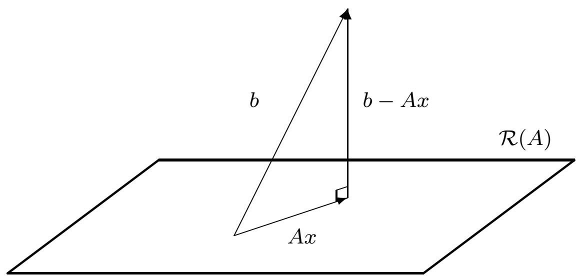

Figure 8.1.1. *Geometric characterization of the least squares solution.*

also positive semi-definite. The normal equations $A^T Ax = A^T b$ are *consistent* since

$$A^T b \in \mathcal{R}(A^T) = \mathcal{R}(A^T A),$$

and therefore a least squares solution always exists.

By Theorem 8.1.1 any least squares solution x will decompose the right hand side b into two orthogonal components

$$b = Ax + r, \quad r \perp Ax. \tag{8.1.4}$$

Here $Ax = P_{\mathcal{R}(A)} b$ is the orthogonal projection (see Sec. 8.1.3) onto $\mathcal{R}(A)$ and $r \in \mathcal{N}(A^T)$ (cf. Figure 8.1.1). Any solution to the (always consistent) normal equations (8.1.3) is a least squares solution. Note that although the least squares solution x may not be unique the decomposition in (8.1.4) always is unique.

Theorem 8.1.2.

The matrix $A^T A$ is positive definite if and only if the columns of A are linearly independent, i.e., when $\operatorname{rank}(A) = n$. *In this case the least squares solution is unique and given by*

$$x = (A^T A)^{-1} A^T b, \qquad r = (I - A(A^T A)^{-1} A^T) b. \tag{8.1.5}$$

Proof. If the columns of A are linearly independent, then $x \neq 0 \Rightarrow Ax \neq 0$. Therefore, $x \neq 0 \Rightarrow x^T A^T Ax = \|Ax\|_2^2 > 0$, and hence $A^T A$ is positive definite. On the other hand, if the columns are linearly dependent, then for some $x_0 \neq 0$ we have $Ax_0 = 0$. Then $x_0^T A^T Ax_0 = 0$, and therefore $A^T A$ is not positive definite. When $A^T A$ is positive definite it is also nonsingular and (8.1.5) follows. $\square$

For the minimum norm problem (8.1.2) let y be any solution of $A^T y = c$, and write $y = y_1 + y_2$, where $y_1 \in \mathcal{R}(A)$. $y_2 \in \mathcal{N}(A^T)$. Then $A^T y_2 = 0$ and hence y_1 is also a solution. Since $y_1 \perp y_2$ we have

$$\|y_1\|_2^2 = \|y\|_2^2 - \|y_2\|_2^2 \le \|y\|_2^2,$$

with equality only if $y_2 = 0$. Hence, the minimum norm solution lies in $\mathcal{R}(A)$ and we can write $y = Az$, for some z. Then we have $A^T y = A^T Az = c$. If A^T has linearly independent rows the inverse of A^TA exists and the minimum norm solution $y \in \mathbb{R}^m$ satisfies the normal equations of second kind

$$y = A(A^TA)^{-1}c. \tag{8.1.6}$$

The solution to the least squares problem is characterized by the two conditions

$$A^T r = 0, \qquad r = b - Ax.$$

These are $n + m$ equations for the unknowns x and the residual $\alpha y = r$, $\alpha > 0$, which we write in the form

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \tag{8.1.7}$$

We will call this the **augmented system** for the least squares problem.

The augmented system is a special case ($B = I$) of the system

$$\begin{pmatrix} B & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix} \tag{8.1.8}$$

where $B \in \mathbf{R}^{m\times m}$ is symmetric positive semi-definite. This system, often called a **saddle-point system**, represents the conditions for equilibrium of a physical system.in (8.1.8) is symmetric but in general indefinite. The system is nonsingular if and only if A has full column rank and the matrix $(B \;\; A)$ has full row rank. If B is positive definite, then r can be eliminated giving the **generalized normal equations**

$$A^T B^{-1} Ax = A^T B^{-1} b - c. \tag{8.1.9}$$

Saddle-point systems occur in numerous applications such as structural optimization and mixed formulations of finite element methods. They also form the kernel in many algorithms for constrained optimization, where they are known as KKT (Karush–Kuhn–Tucker) systems.

As shown in the following theorem, the system (8.1.8) gives a unified formulation of generalized least squares and minimum norm problems.

Theorem 8.1.3. *Assume that the matrix B is positive definite. Then the linear system* (8.1.8) *gives the first order conditions for the following two optimization problems:*

1. *Generalized linear least squares problems (GLLS)*

$$\min_x \tfrac{1}{2}\|Ax - b\|^2_{B^{-1}} + c^T x. \tag{8.1.10}$$

2. *Equality constrained quadratic optimization (ECQO)*

$$\min_r \tfrac{1}{2}\|r - b\|_B, \quad \textit{subject to} \;\; A^T r = c, \tag{8.1.11}$$

where the vector norms is defined as $\|x\|^2_{B^{-1}} = x^T B^{-1} x$ *and* $\|x\|^2_B = x^T Bx$.

Proof. If B is symmetric positive definite so is B^{-1}. The system (8.1.8) can be obtained by differentiating (8.1.10) to give

$$A^T B^{-1}(Ax - b) + c = 0,$$

and setting $r = B^{-1}(b - Ax)$. The system can also be obtained by differentiating the Lagrangian

$$L(x, r) = \frac{1}{2} r^T Br - r^T b + x^T (A^T r - c)$$

of (8.1.11), and equating to zero. Here x is the vector of Lagrange multipliers. □

The standard linear least squares problem $\min_x \|Ax - b\|_2$ is obtained by taking $B = I$ and $c = 0$. Taking $B = I$ in problem 2 this becomes

$$\min_r \frac{1}{2}\|r - b\|_2, \quad \text{subject to} \quad A^T r = c, \tag{8.1.12}$$

is to find the point r closest to b in the set of solutions to the underdetermined linear system $A^T r = c$. The solution to this problem can be written

$$r = b - Ax = P_{\mathcal{N}(A^T)} b + A(A^T A)^{-1} c. \tag{8.1.13}$$

In particular, taking $b = 0$, this is the **minimum norm solution** of the system $A^T y = c$.

8.1.2 The Gauss–Markov Theorem

Gauss claims he discovered the method of least squares in 1795. He used it for analyzing surveying data and for astronomical calculation. A famous example is when Gauss successfully predicted the orbit of the asteroid Ceres in 1801.

Gauss [150] in 1821 put the method of least squares on a sound theoretical basis. To describe his results we first need to introduce some concepts from statistics. Let the probability that random variable $y \leq x$ be equal to $F(x)$, where $F(x)$ is nondecreasing, right continuous, and satisfies

$$0 \leq F(x) \leq 1, \quad F(-\infty) = 0, \quad F(\infty) = 1.$$

Then $F(x)$ is called the **distribution function** for y.

The **expected value** and the **variance** of y are defined as the Stieltjes integrals

$$\mathcal{E}(y) = \mu = \int_{-\infty}^{\infty} y dF(y), \qquad \mathcal{E}(y - \mu)^2 = \sigma^2 = \int_{-\infty}^{\infty} (y - \mu)^2 dF(y),$$

If $y = (y_1, \dots, y_n)^T$ is a vector of random variables and $\mu = (\mu_1, \dots, \mu_n)^T$, $\mu_i = \mathcal{E}(y_i)$, then we write $\mu = \mathcal{E}(y)$. If y_i and y_j have the joint distribution $F(y_i, y_j)$ the **covariance** between y_i and y_j is

$$\begin{aligned}\sigma_{ij} &= \mathcal{E}[(y_i - \mu_i)(y_j - \mu_j)] = \int_{-\infty}^{\infty} (y_i - \mu_i)(y_j - \mu_j) dF(y_i, y_j) \\ &= \mathcal{E}(y_i y_j) - \mu_i \mu_j.\end{aligned}$$

The covariance matrix $V \in \mathbf{R}^{n\times n}$ of y is defined by

$$V = \mathcal{V}(y) = \mathcal{E}[(y-\mu)(y-\mu)^T] = \mathcal{E}(yy^T) - \mu\mu^T.$$

where the diagonal element σii is the variance of y_i.

We now prove some properties which will be useful in the following.

Lemma 8.1.4.

Let $B \in \mathbf{R}^{r\times n}$ be a matrix and y a random vector with $\mathcal{E}(y) = \mu$ and covariance matrix V. Then the expected value and covariance matrix of By is

$$\mathcal{E}(By) = B\mu, \qquad \mathcal{V}(By) = BVB^T. \tag{8.1.14}$$

In the special case that $B = b^T$ is a row vector $\mathcal{V}(b^T y) = \mu\|b\|_2^2$.

Proof. The first property follows directly from the definition of expected value. The second follows from the relation

$$\begin{aligned}\mathcal{V}(By) &= \mathcal{E}[(B(y-\mu)(y-\mu)^T B^T] \\ &= B\mathcal{E}[(y-\mu)(y-\mu)^T]B^T = BVB^T.\end{aligned}$$

□

In linear statistical models one assumes that the vector $b \in \mathbf{R}^m$ of observations is related to the unknown parameter vector $x \in \mathbf{R}^n$ by a linear relationship

$$Ax = b + \epsilon, \tag{8.1.15}$$

where $A \in \mathbf{R}^{m\times n}$, $\operatorname{rank}(A) = n$, is a known matrix and ϵ is a random vector of errors. In the **standard case** ϵ is assumed to have zero mean and covariance matrix $\sigma^2 I$,

$$\mathcal{E}(\epsilon) = 0, \qquad \mathcal{V}(\epsilon) = \sigma^2 I. \tag{8.1.16}$$

We also make the following definitions:

Definition 8.1.5.

A function g of the random vector y is called unbiased estimate of a parameter θ if $\mathcal{E}(g(y)) = 0$. When such a function exists, then θ is called an estimable parameter.

Definition 8.1.6.

The linear function $g = c^T y$, where c is a constant vector, is a minimum variance (best) unbiased estimate of the parameter θ if $\mathcal{E}(g) = \theta$, and $\mathcal{V}(g)$ is minimized over all linear estimators.

Gauss [150] in 1821 put the method of least squares on a sound theoretical basis. In his textbook from 1912 Markov [271] refers to Gauss' work and may have clarified some implicit assumptions but proves nothing new; see Placket [311, 312]

Theorem 8.1.7 (*The Gauss–Markov Theorem***).**

Consider the linear model (8.1.15)*, where $A \in \mathbf{R}^{m\times n}$ is a known matrix, and ϵ is a random vector with mean and covariance matrix given by* (8.1.16)*. Let $\hat{x}$ be the least square estimator, obtained by minimizing over x the sum of squares $\|Ax-b\|_2^2$. Then the best linear unbiased estimator of any linear functional $g = c^T x$ is $c^T\hat{x}$. Furthermore, the covariance matrix of the estimate $\hat{x}$ equals*

$$\mathcal{V}(\hat{x}) = V = \sigma^2(A^TA)^{-1} \tag{8.1.17}$$

and $\mathcal{E}(s^2) = \sigma^2$, where s^2 is the quadratic form

$$s^2 = \frac{1}{m-n}\|b - A\hat{x}\|_2^2. \tag{8.1.18}$$

In the literature Gauss–Markov theorem is sometimes stated in less general forms. It is important to note that in the theorem errors are *not* assumed to be normally distributed, nor are they assumed to be independent (but only uncorrelated—a weaker condition). They are also *not* assumed to be identically distributed, but only having zero mean and the same variance.

The residual vector $\hat{r} = \hat{b} - Ax$ of the least squares solution satisfies $A^T\hat{r} = 0$, i.e. $\hat{r}$ is orthogonal to the column space of A. This condition gives n linear relations among the m components of $\hat{r}$. It can be shown that the residuals $\hat{r}$ and therefore also s^2 are uncorrelated with $\hat{x}$, i.e.,

$$\mathcal{V}(\hat{r}, \hat{x}) = 0, \qquad \mathcal{V}(s^2, \hat{x}) = 0.$$

An estimate of the covariance of the linear functional c^Tx is given by $s^2(c^T(A^TA)^{-1}c)$. In particular, for the components $x_i = e_i^Tx$,

$$s^2(e_i^T(A^TA)^{-1}e_i) = s^2(A^TA)^{-1}_{ii}.$$

the ith diagonal element of $(A^TA)^{-1}$.

In the **general univariate linear model** the covariance matrix equals a positive semi-definite symmetric matrix $\mathcal{V}(\epsilon) = \sigma^2 V \in \mathbf{R}^{m\times m}$. If A has full column rank and V is positive definite, then the best unbiased linear estimate is given by the solution of

$$\min_x (Ax-b)^T V^{-1}(Ax-b). \tag{8.1.19}$$

The covariance matrix of the estimate $\hat{x}$ is

$$\mathcal{V}(\hat{x}) = \sigma^2(A^T V^{-1} A)^{-1} \tag{8.1.20}$$

and an unbiased estimate of σ

$$s^2 = \frac{1}{m-n}(b - A\hat{x})^T V^{-1}(b - A\hat{x}), \tag{8.1.21}$$

If the errors are uncorrelated with variances $v_{ii} > 0$, $i = 1 : m$, then V is diagonal and (8.1.19) is called a **weighted least squares** problem. Hence, if the ith equation is scaled by $1/\sqrt{v_{ii}}$, i.e. the larger the variance the smaller weight should be given to a particular equation. It is important to note that different scalings will give different solutions, unless the system is $Ax = b$ is consistent.

8.1.3 Orthogonal and Oblique Projections

We have seen that the least squares solution is characterized by the property that its residual is orthogonal to its projection onto $\mathcal{R}(A)$. In this section make a systematic study of both orthogonal and more general projection matrices.

Any square matrix $P \in \mathbf{R}^{n\times n}$ such that

$$P^2 = P. \tag{8.1.22}$$

is called **idempotent** and a **projector**. An arbitrary vector $v \in \mathbf{R}^n$ can be decomposed in a unique way as

$$v = Pv + (I - P)v = v_1 + v_2. \tag{8.1.23}$$

Here $v_1 = Pv \in S$ is a projection of v onto $\mathcal{R}(P)$, the range space of P. Since $Pv_2 = (P - P^2)v = 0$ it follows that $(I - P)$ is a projection onto $\mathcal{N}(P)$, the null space of P.

If P is symmetric, $P^T = P$, then

$$v_1^T v_2 = (Pv)^T(I - P)v = v^T P(I - P)v = v^T(P - P^2)v = 0.$$

It follows that $v_2 \perp S$, i.e., v_2 lies in the orthogonal complement $S^\perp$ of S; In this case P is the **orthogonal projector** onto S and $I - P$ the orthogonal projector onto $S^\perp$. It can be shown that the orthogonal projector P onto a given subspace S is unique, see Problem 1.

A projector P such that $P \neq P^T$ is called an **oblique projector**. If λ is an eigenvalue of a projector P then from $P^2 = P$ it follows that $\lambda^2 = \lambda$. Hence, the eigenvalues of P are either 1 or 0 and we can write the eigendecomposition

$$P = (\,U_1 \quad U_2\,)\begin{pmatrix} I_k & 0 \\ 0 & 0_{n-k}\end{pmatrix}\begin{pmatrix}\hat{Y}_1^T \\ \hat{Y}_2^T\end{pmatrix}, \quad \begin{pmatrix}\hat{Y}_1^T \\ \hat{Y}_2^T\end{pmatrix} = (U_1\, U_2)^{-1}, \tag{8.1.24}$$

where $k = \operatorname{trace}(P)$ is the rank of P. The matrices $U_1 \in \mathbf{R}^{n\times k}$ and $U_2 \in \mathbf{R}^{n\times n-k}$ can be chosen as orthogonal bases for the invariant subspaces corresponding to the eigenvalues 1 or 0, respectively and

$$\operatorname{span}(U_1) = \mathcal{R}(P), \qquad \operatorname{span}(U_2) = \mathcal{N}(P).$$

In terms of this eigendecomposition

$$P = U_1\hat{Y}_1^T, \qquad I - P = U_2\hat{Y}_2^T. \tag{8.1.25}$$

and the splitting (8.1.23) can be written

$$v = (U_1\hat{Y}_1^T)v + (U_2\hat{Y}_2^T)v = v_1 + v_2. \tag{8.1.26}$$

Here v_1 is the oblique projection of v onto $\mathcal{R}(U_1)$ along $\mathcal{R}(U_2)$. Similarly, v_2 is the oblique projection onto $\mathcal{R}(U_2)$ along $\mathcal{R}(U_1)$.

If $P^T = P$ then P is an orthogonal projector and in (8.1.24) we can take $Y = U = (\,U_1 \quad U_2\,)$ to be orthogonal. The projectors (8.1.26) then have the form

$$P = U_1U_1^T, \qquad I - P = U_2U_2^T; \tag{8.1.27}$$

For an orthogonal projector we have

$$\|Pv\|_2 = \|U_1^Tv\|_2 \le \|v\|_2 \quad \forall \quad v \in \mathbf{R}^m, \tag{8.1.28}$$

where equality holds for all vectors in $\mathcal{R}(U_1)$ and thus $\|P\|_2 = 1$. The conversion is also true; P is an orthogonal projection only if (8.1.28) holds.

From (8.1.24) we have

$$I = \begin{pmatrix} \hat{Y}_1^T \\ \hat{Y}_2^T \end{pmatrix} (U_1 \; U_2) = \begin{pmatrix} \hat{Y}_1^TU_1 & \hat{Y}_1^TU_2 \\ \hat{Y}_2^TU_1 & \hat{Y}_2^TU_2 \end{pmatrix} = \begin{pmatrix} I_k & 0 \\ 0 & I_{n-k} \end{pmatrix}. \tag{8.1.29}$$

In particular, $\hat{Y}_1^TU_2 = 0$ and $\hat{Y}_2^TU_1 = 0$. Hence, the columns of $\hat{Y}_1$ form a basis of the orthogonal complement of $\mathcal{R}(U_2)$ and, similarly, the columns of $\hat{Y}_2$ form a basis of the orthogonal complement of $\mathcal{R}(U_1)$.

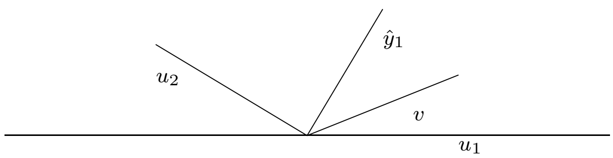

Figure 8.1.2. *The oblique projection of v on u_1 along u_2.*

Let Y_1 be an orthogonal matrix whose columns span $\mathcal{R}(\hat{Y}_1)$. Then there is a nonsingular matrix G_1 such that $\hat{Y}_1 = Y_1G$. From (8.1.29) it follows that $G^TY_1^TU_1 = I_k$, and hence $G^T = (Y_1^TU_1)^{-1}$. Similarly, $Y_2 = (Y_2^TU_2)^{-1}Y_2$ is an orthogonal matrix whose columns span $\mathcal{R}(\hat{Y}_2)$. Hence, using (8.1.25) the projectors in (8.1.25) can also be written

$$P = U_1(Y_1^TU_1)^{-1}Y_1^T, \qquad I - P = U_2(Y_2^TU_2)^{-1}Y_2^T. \tag{8.1.30}$$

Example 8.1.2.

We illustrate the case when $n = 2$ and $n_1 = 1$. Let the vectors u_1 and y_1 be normalized so that $\|u_1\|_2 = \|y_1\|_2 = 1$ and let $y_1^T u_1 = \cos\theta$, where θ is the angle between u_1 and y_1, see Figure 8.1.2. Since

$$P = u_1(y_1^T u_1)^{-1} y_1^T = \frac{1}{\cos\theta} u_1 y_1^T.$$

Hence, $\|P\|_2 = 1/\cos\theta \geq 1$, and $\|P\|_2$ becomes very large when y_1 is almost orthogonal to u_1. When $y_1 = u_1$ we have $\theta = 0$ and P is an orthogonal projection.

8.1.4 Generalized Inverses and the SVD

The SVD is a powerful tool both for analyzing and solving linear least squares problems. The reason for this is that the orthogonal matrices that transform A to diagonal form do not change the l_2-norm. We have the following fundamental result.

Theorem 8.1.8.

Let $A \in \mathbf{R}^{m\times n}$, $\operatorname{rank}(A) = r$, and consider the general linear least squares problem

$$\min_{x\in S} \|x\|_2, \qquad S = \{x \in \mathbf{R}^n \mid \|b - Ax\|_2 = \min\}. \tag{8.1.31}$$

This problem always has a unique solution, which in terms of the SVD of A can be written as

$$x = V \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T b, \tag{8.1.32}$$

Proof. Let

$$c = U^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix},$$

where $z_1, c_1 \in \mathbf{R}^r$. Using the orthogonal invariance of the l_2 norm we have

$$\begin{aligned}\|b - Ax\|_2 &= \|U^T(b - AVV^T x)\|_2 \\ &= \left\| \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} - \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} c_1 - \Sigma_1 z_1 \\ c_2 \end{pmatrix} \right\|_2.\end{aligned}$$

The residual norm will attain its minimum value equal to $\|c_2\|_2$ for $z_1 = \Sigma_1^{-1} c_1$, z_2 arbitrary. Obviously the choice $z_2 = 0$ minimizes $\|x\|_2 = \|Vz\|_2 = \|z\|_2$. $\square$

Note that problem (8.1.31) includes as special cases the solution of both overdetermined and underdetermined linear systems. We can write $x = A^\dagger b$, where

$$A^\dagger = V \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T \in \mathbf{R}^{n\times m} \tag{8.1.33}$$

is the unique **pseudo-inverse** of A and x is called the pseudo-inverse solution of $Ax = b$.

Methods for computing the SVD are described in Sec. 10.8. Note that for solving least squares problems we only need to compute the singular values, the matrix V_1 and vector $c = U_1^T b$, where we have partitioned $U = (U_1\ U_2)$ and $V = (V_1\ V_2)$ so that U_1 and V_1 have $r = \mathrm{rank}\,(A)$ columns. The pseudo-inverse solution (8.1.33) can then be written

$$x = V_1 \Sigma_1^{-1} U_1^T b = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} \cdot v_i, \quad r = \mathrm{rank}\,(A). \tag{8.1.34}$$

The matrix $A^\dagger$ is often called the **Moore–Penrose inverse**. Moore 1920 developed the concept of the general reciprocal in 1920. Penrose [1955], gave an elegant algebraic characterization and showed that $X = A^\dagger$ is uniquely determined by the four **Penrose conditions** :

$$(1) \quad AXA = A, \qquad (2) \quad XAX = X, \tag{8.1.35}$$

$$(3) \quad (AX)^T = AX, \qquad (4) \quad (XA)^T = XA. \tag{8.1.36}$$

It can be directly verified that $X = A^\dagger$ given by (8.1.33) satisfies these four conditions. In particular, this shows that $A^\dagger$ does not depend on the particular choices of U and V in the SVD. (See also Problem 2.)

The orthogonal projections onto the four fundamental subspaces of A have the following simple expressions in terms of the pseudo-inverse :

$$\begin{aligned} P_{\mathcal{R}(A)} &= AA^\dagger, \qquad & P_{\mathcal{N}(A^T)} &= I - AA^\dagger, \\ P_{\mathcal{R}(A^T)} &= A^\dagger A, \qquad & P_{\mathcal{N}(A)} &= I - A^\dagger A. \end{aligned} \tag{8.1.37}$$

These expressions are easily verified using the definition of an orthogonal projection and the Penrose conditions.

Another useful characterization of the pseudo-inverse solution is the following:

Theorem 8.1.9.

The pseudo-inverse solution $x = A^\dagger b$ is uniquely characterized by the two geometrical conditions

$$x \perp \mathcal{N}(A), \qquad Ax = P_{\mathcal{R}(A)}\, b. \tag{8.1.38}$$

Proof. These conditions are easily verified from (8.1.34). □

In the special case that $A \in \mathbf{R}^{m\times n}$ and $\mathrm{rank}\,(A) = n$ it holds that

$$A^\dagger = (A^T A)^{-1} A^T, \qquad (A^T)^\dagger = A(A^T A)^{-1} \tag{8.1.39}$$

These expressions follow from the normal equations (8.1.5) and (8.1.6). Some properties of the usual inverse can be extended to the pseudo-inverse, e.g., the relations

$$(A^\dagger)^\dagger = A, \qquad (A^T)^\dagger = (A^\dagger)^T,$$

easily follow form (8.1.33). In general $(AB)^\dagger \neq B^\dagger A^\dagger$. The following theorem gives a useful *sufficient* conditions for the relation $(AB)^\dagger = B^\dagger A^\dagger$ to hold.

Theorem 8.1.10.
If $A \in \mathbf{R}^{m\times r}$, $B \in \mathbf{R}^{r\times n}$, *and* $\operatorname{rank}(A) = \operatorname{rank}(B) = r$, *then*

$$(AB)^\dagger = B^\dagger A^\dagger = B^T(BB^T)^{-1}(A^TA)^{-1}A^T. \tag{8.1.40}$$

Proof. The last equality follows from (8.1.39). The first equality is verified by showing that the four Penrose conditions are satisfied. □

Any matrix A^- satisfying the first Penrose condition

$$AA^-A = A \tag{8.1.41}$$

is called a **generalized inverse** of A. It is also called an **inner inverse** or $\{1\}$-inverse. If it satisfies the second condition $AA^-A = A^-AA^- = A^-$ it is called an **outer inverse** or a $\{2\}$-inverse.

Let A^- be a $\{1\}$-inverse of A. Then for all b such that the system $Ax = b$ is consistent $x = A^-b$ is a solution. The general solution can be written

$$x = A^-b + (I - A^-A)y, \quad y \in \mathbf{C}^n.$$

We also have

$$(AA^-A^-)^2 = AA^-AA^- = AA^-, \qquad (A^-A)^2 = A^-AA^-A = A^-A.$$

This shows that AA^- and A^-A are idempotent and therefore (in general oblique) projectors

$$AX = P_{\mathcal{R}(A),S}, \qquad XA = P_{T,\mathcal{N}(A)},$$

where S and T are some subspaces complementary to $\mathcal{R}(A)$ and $\mathcal{N}(A)$, respectively.

Let $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$. Then $\|Ax-b\|_2$ is the minimized when x satisfies the normal equations $A^TAx = A^Tb$. Suppose now that a generalized inverse A^- satisfies

$$(AA^-)^T = AA^-. \tag{8.1.42}$$

Then AA^- is the orthogonal projector onto $\mathcal{R}(A)$ and A^- is called a **least squares inverse**. We have

$$A^T = (AA^-A)^T = A^TAA^-,$$

which shows that $x = A^-b$ satisfies the normal equations and therefore is a least squares solution. Conversely, if $A^- \in \mathbf{R}^{n\times m}$ has the property that for all b, $\|Ax-b\|_2$ is smallest when $x = A^-b$, then A^- is a a least squares inverse

The following dual result also holds. If A^- is a generalized inverse, and

$$(A^-A)^T = A^-A$$

then A^-A is the orthogonal projector orthogonal to $\mathcal{N}(A)$ and A^- is called a **minimum norm inverse**. If $Ax = b$ is consistent, then the unique solution for which $\|x\|_2$ is smallest satisfies the normal equations

$$x = A^Tz, \quad AA^Tz = b.$$

For a minimum norm inverse we have

$$A^T = (AA^-A)^T = A^-AA^T,$$

and hence $x = A^Tz = A^-(AA^T)z = A^-b$, which shows that $x = A^-b$ is the solution of smallest norm.

Conversely, if $A^- \in \mathbf{R}^{n\times m}$ is such that, whenever $Ax = b$ has a solution then $x = A^-b$ is a minimum norm solution, then A^- is a minimum norm inverse.

We first give some perturbation bounds for the pseudo-inverse. We consider a matrix $A \in \mathbf{R}^{m\times n}$ and let $B = A + E$ be the perturbed matrix. The theory is complicated by the fact that when the rank of A changes $A^\dagger$ varies discontinuously

Example 8.1.3.

When the rank changes the perturbation in $A^\dagger$ may be unbounded when the perturbation $\|E\|_2 \to 0$. A trivial example of this is obtained by taking

$$A = \begin{pmatrix} \sigma & 0 \\ 0 & 0 \end{pmatrix}, \quad E = \begin{pmatrix} 0 & 0 \\ 0 & \epsilon \end{pmatrix},$$

where $\sigma > 0$, $\epsilon \neq 0$. Then $1 = \text{rank}\,(A) \neq \text{rank}\,(A+E) = 2$,

$$A^\dagger = \begin{pmatrix} \sigma^{-1} & 0 \\ 0 & 0 \end{pmatrix}, \quad (A+E)^\dagger = \begin{pmatrix} \sigma^{-1} & 0 \\ 0 & \epsilon^{-1} \end{pmatrix},$$

and $\|(A+E)^\dagger - A^\dagger\|_2 = |\epsilon|^{-1} = 1/\|E\|_2$. □

This example shows that formulas derived by operating formally with pseudo-inverses may have no meaning numerically..

The perturbations for which the pseudo-inverse is well behaved can be characterized by the condition

$$\text{rank}\,(A) = \text{rank}\,(B) = \text{rank}\,(P_{\mathcal{R}(A)}BP_{\mathcal{R}(A^T)}); \tag{8.1.43}$$

The matrix B is said to be an **acute perturbation** of A if this condition holds; see Stewart [342, 1977]. In particular, we have the following result.

Theorem 8.1.11.

If rank $(A+E) =$ rank $(A) = r$, *and* $\eta = \|A^\dagger\|_2\|E\|_2 < 1$, *then*

$$\|(A+E)^\dagger\|_2 \le \frac{1}{1-\eta}\|A^\dagger\|_2. \tag{8.1.44}$$

Proof. From the assumption and Theorem 1.2.7 it follows that

$$1/\|(A+E)^\dagger\|_2 = \sigma_r(A+E) \ge \sigma_r(A) - \|E\|_2 = 1/\|A^\dagger\|_2 - \|E\|_2 > 0,$$

which implies (8.1.44). □

Let $A, B \in \mathbf{R}^{m\times n}$, and $E = B - A$. If A and $B = A + E$ are square nonsingular matrices, then we have the well-known identity

$$B^{-1} - A^{-1} = -B^{-1}EA^{-1}.$$

In the general case **Wedin's pseudo-inverse identity** (see [383]) holds

$$B^\dagger - A^\dagger = -B^\dagger EA^\dagger + (B^TB)^\dagger E^T P_{\mathcal{N}(A^T)} + P_{\mathcal{N}(B)}E^T(AA^T)^\dagger, \tag{8.1.45}$$

This identity can be proved by expressing the projections in terms of pseudo-inverses using the relations in (8.1.37).

Let $A = A(\alpha)$ be a matrix, where α is a scalar parameter. Under the assumption that $A(\alpha)$ has local constant rank the following formula for the derivative of the pseudo-inverse $A^\dagger(\alpha)$ follows from (8.1.45):

$$\frac{dA^\dagger}{d\alpha} = -A^\dagger \frac{dA}{d\alpha} A^\dagger + (A^TA)^\dagger \frac{dA^T}{d\alpha} P_{N(A)} + P_{N(A^T)} \frac{dA^T}{d\alpha}(AA^T)^\dagger. \tag{8.1.46}$$

This formula is due to Wedin [383, p 21]. We observe that if A has full column rank then the second term vanishes; if A has full row rank then it is the third term that vanishes. The variable projection algorithm for separable nonlinear least squares is based on a related formula for the derivative of the orthogonal projection matrix $P_{\mathcal{R}(A)}$; see Sec. 11.2.5.

For the case when $\operatorname{rank}(B) = \operatorname{rank}(A)$ the following theorem applies.

Theorem 8.1.12. *If $B = A + E$ and $\operatorname{rank}(B) = \operatorname{rank}(A)$, then*

$$\|B^\dagger - A^\dagger\| \le \mu \|B^\dagger\|\, \|A^\dagger\|\, \|E\| \tag{8.1.47}$$

where $\mu = 1$ for the Frobenius norm $\|\cdot\|_F$, and for the spectral norm $\|\cdot\|_2$,

$$\mu = \begin{cases} \frac{1}{2}(1+\sqrt{5}) & \text{if } \operatorname{rank}(A) < \min(m,n), \\ \sqrt{2} & \text{if } \operatorname{rank}(A) = \min(m,n). \end{cases}$$

Proof. For the $\|\cdot\|_2$ norm, see Wedin [384]. The result that $\mu = 1$ for the Frobenius norm is due to van der Sluis and Veltkamp [366]. $\square$

8.1.5 Matrix Approximation and the SVD

The singular values decomposition (SVD) plays a very important role in a number of least squares matrix approximation problems. In this section we have collected a number of results that will be used extensively in the following.

The singular values have the following extremal property, the **minimax characterization**.

Theorem 8.1.13.

Let $A \in \mathbf{R}^{m\times n}$ have singular values $\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_p \ge 0$, $p = \min(m,n)$, and S be a linear subspace of $\mathbf{R}^n$ of dimension $\dim(S)$. *Then*

$$\sigma_i = \min_{dim(S)=n-i+1} \max_{\substack{x\in S\\ x\neq 0}} \frac{\|Ax\|_2}{\|x\|_2}. \tag{8.1.48}$$

Proof. The result is established in almost the same way as for the corresponding eigenvalue theorem, Theorem 10.3.9 (Fischer's theorem). □

The minimax characterization of the singular values may be used to establish the following relations between the singular values of two matrices A and B.

Theorem 8.1.14.

Let $A, B \in \mathbf{R}^{m\times n}$ have singular values $\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_p$ and $\tau_1 \ge \tau_2 \ge \ldots \ge \tau_p$ respectively, where $p = \min(m,n)$. Then

$$\max_i |\sigma_i - \tau_i| \le \|A-B\|_2, \tag{8.1.49}$$

$$\sum_{i=1}^p |\sigma_i - \tau_i|^2 \le \|A-B\|_F^2. \tag{8.1.50}$$

Proof. See Stewart [340, pp. 321–322]. □

Hence, perturbations of the elements of a matrix A result in perturbations of the same, or smaller, magnitude in the singular values. This result is important for the use of the SVD to determine the "numerical rank" of a matrix; see below.

The eigenvalues of the leading principal minor of order $n-1$ of a Hermitian matrix C can be shown to interlace the eigenvalues of C, see Theorem 10.3.8. From the relation (7.1.48) corresponding results can be derived for the singular values of a matrix A.

Theorem 8.1.15.

Let

$$\hat{A} = (A, u) \in \mathbf{R}^{m\times n}, \quad m \ge n, \quad u \in \mathbf{R}^m.$$

Then the ordered singular values σ_i of A interlace the ordered singular values $\hat{\sigma}_i$ of $\hat{A}$ as follows

$$\hat{\sigma}_1 \ge \sigma_1 \ge \hat{\sigma}_2 \ge \sigma_2 \ldots \ge \hat{\sigma}_{n-1} \ge \sigma_{n-1} \ge \hat{\sigma}_n.$$

Similarly, if A is bordered by a row,

$$\hat{A} = \begin{pmatrix} A \\ v^* \end{pmatrix} \in \mathbf{R}^{m\times n}, \quad m > n, \quad v \in \mathbf{R}^n,$$

then

$$\hat{\sigma}_1 \ge \sigma_1 \ge \hat{\sigma}_2 \ge \sigma_2 \ldots \ge \hat{\sigma}_{n-1} \ge \sigma_{n-1} \ge \hat{\sigma}_n \ge \sigma_n.$$

The best approximation of a matrix A by another matrix of lower rank can be expressed in terms of the SVD of A.

Theorem 8.1.16.
Let $\mathcal{M}_k^{m\times n}$ denote the set of matrices in $\mathbf{R}^{m\times n}$ of rank k. Assume that $A \in \mathcal{M}_r^{m\times n}$ and consider the problem

$$\min_{X\in\mathcal{M}_k^{m\times n}} \|A - X\|, \quad k < r.$$

Then the SVD expansion of A truncated to k terms $X = B = \sum_{i=1}^k \sigma_i u_i v_i^T$, solves this problem both for the l_2 norm and the Frobenius norm. Further, the minimum distance is given by

$$\|A - B\|_2 = \sigma_{k+1}, \qquad \|A - B\|_F = (\sigma_{k+1}^2 + \ldots + \sigma_r^2)^{1/2}.$$

The solution is unique for the Frobenius norm but not always for the l_2 norm.

Proof. Eckhard and Young [119] proved it for the Frobenius norm. Mirsky [280] generalized it to unitarily invariant norms, which includes the l_2-norm. □

Let $A \in \mathbf{C}^{m\times n}$, be a matrix of rank n with the "thin" SVD $A = U_1\Sigma V^H$. Since $A = U_1\Sigma V^H = U_1\Sigma U_1^H U_1 V^H$, we have

$$A = PH, \quad P = U_1 V^H, \quad H = V\Sigma V^H, \tag{8.1.51}$$

where P is unitary, $P^H P = I$, and $H \in \mathbf{C}^{n\times n}$ is Hermitian positive semi-definite. The decomposition (8.1.51) is called the **polar decomposition** of A. If rank $(A) = n$, then H is positive definite and the polar decomposition is unique. If the polar decomposition $A = PH$ is given, then from a spectral decomposition $H = V\Sigma V^H$ one can construct the singular value decomposition $A = (PV)\Sigma V^H$.

The unitary factor in the polar decomposition can be written in the form

$$P = e^{iF},$$

where F is a Hermitian matrix; see Gantmacher [147, p. 278]. The decomposition $A = e^{iF}H$ can be regarded as a generalization to matrices of the complex number representation $z = re^{i\theta}$, $r \geq 0$!

The polar decomposition is also related to the matrix square root and sign functions; see Sec. 9.5.5. The significance of the factor P in the polar decomposition is that it is the unitary matrix closest to A.

Theorem 8.1.17.
Let $A \in \mathbf{C}^{m\times n}$ be a given matrix and $A = UH$ its polar decomposition., Then for any unitary matrix $U \in \mathcal{M}_{m\times n}$,

$$\|A - U\|_F \geq \|A - P\|_F.$$

Proof. This theorem was proved for $m = n$ and general unitarily invariant norms by Fan and Hoffman [126]. The generalization to $m > n$ follows from the additive property of the Frobenius norm. □

Less well know is that the optimal properties of the Hermitian polar factor H. Let $A \in \mathbf{C}^{n\times n}$ be a Hermitian matrix with at least one negative eigenvalue. Consider the problem of finding a perturbation E such that $A+E$ is positive semi-definite.

Theorem 8.1.18.
Let $A \in \mathbf{C}^{m\times n}$ be Hermitian and $A = UH$ its polar decomposition. Set

$$B = A + E = \frac{1}{2}(H + A), \qquad E = \frac{1}{2}(H - A).$$

Then for any positive semi-definite Hermitian matrix X it holds that

$$\|A - B\|_2 \leq \|A - X\|_2.$$

Proof. See Higham [208]. □

Any real orthogonal matrix Q, with $det(Q) = +1$, can be written as $Q = e^K$, where K is skew-symmetric since

$$Q^T = e^{K^T} = e^{-K} = Q^{-1}.$$

The eigenvalues of a skew-symmetric matrix K lie on the imaginary axis and are mapped onto eigenvalues for Q on the unit circle by the mapping $Q = e^K$

Example 8.1.4.
Let $A \in \mathbf{R}^{3\times 3}$ have the polar decomposition $A = QS,$, where $Q = e^K$ and

$$K = \begin{pmatrix} 0 & k_{12} & k_{13} \\ -k_{12} & 0 & k_{23} \\ -k_{13} & -k_{23} & 0, \end{pmatrix}$$

is skew-symmetric.

8.1.6 Elementary Orthogonal Matrices

When solving linear equations we made use of elementary elimination matrices of the form $L_j = I + l_j e_j^T$, see (7.2.16). These were used to describe the elementary steps in Gaussian elimination and LU factorization.

We now introduce **elementary orthogonal matrices**, which are equal to the unit matrix modified by a matrix of rank one. Such matrices are flexible and useful tools for constructing algorithms for a variety of problems in linear algebra. The great attraction of using such transformations stems from the fact that they preserve both the Euclidean and Frobenius norm. Hence, their use leads to numerically stable algorithms.

Symmetric matrices of the form

$$P = I - 2uu^T, \quad \|u\|_2 = 1, \tag{8.1.52}$$

are of fundamental importance in matrix computations. Since

$$P^T P = P^2 = I - 4uu^T + 4u(u^T u)u^T = I,$$

it follows that P is orthogonal and $P^2 = I$. Hence the inverse is readily available, $P^{-1} = P$. Note that $Pa \in \text{span}[a, u]$, and $Pu = -u$, i.e., P reverses u. Further, for any $a \perp u$ we have $Pa = a$. The effect of the transformation Pa for a general vector a is to reflect a in the $(m-1)$ dimensional hyper plane with normal vector u; see Figure 8.1.3. This is equivalent to subtracting *twice* the orthogonal projection of a onto u. The normal u is parallel to the difference $(a - Pa)$. The

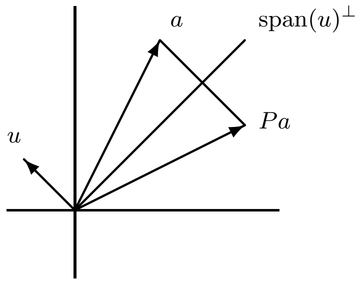

Figure 8.1.3. *A Householder reflection of the vector a.*

use of elementary reflectors in numerical linear algebra was made popular in matrix computation by Householder [218]. Matrices of the form (8.1.52) are therefore often called **Householder reflectors** and the vector u is called a **Householder vector**.

If a matrix $A = (a_1, \dots, a_n) \in \mathbf{R}^{m\times n}$ is *premultiplied* by P the product is $PA = (Pa_1, \dots, Pa_n$, where

$$Pa_j = a_j - \beta(u^T a_j)u. \tag{8.1.53}$$

An analogous formula, exists for *postmultiplying* A with P, where P now acts on the *rows* of A. Hence such products can be computed without explicitly forming P itself. The products

$$PA = A - \beta u(u^T A), \qquad AP = A - \beta(Au)u^T,$$

can be computed in in $4mn$ flops using one matrix–vector product followed by a rank one update.

An important use of Householder reflectors is the computation of the QR factorization of a matrix $A \in \mathbf{R}^{m\times n}$ for solving least squares problems. Another is the reduction of a matrix by an orthogonal similarity transformation to condensed form, when solving eigenvalue problems. For these and other applications we need to consider the following construction. Let $a \in \mathbf{R}^m$, be a given nonzero vector. We want to construct a plane reflection such that *multiplication by P zeros all components except the first in a*, i.e.,

$$Pa = \pm\sigma e_1, \quad \sigma = \|a\|_2. \tag{8.1.54}$$

Multiplying (8.1.54) from the left by P and using $P^2 = I$ it follows that $y = Ue_1$ satisfies $U^T y = e_1$ or

$$Pe_1 = \pm a/\sigma.$$

Hence, (8.1.54) is equivalent to finding a square orthogonal matrix P with its first column proportional to $\pm a/\sigma$. It is easily seen that (8.1.54) is satisfied if we take

$$u = a \mp \sigma e_1 = \begin{pmatrix} \alpha_1 \mp \sigma \\ a_2 \end{pmatrix}, \quad a = \begin{pmatrix} \alpha_1 \\ a_2 \end{pmatrix}. \tag{8.1.55}$$

Note that u differs from a only in its first component. A short calculation shows that

$$1/\beta = \tfrac{1}{2}u^T u = \tfrac{1}{2}(a \mp \sigma e_1)^T (a \mp \sigma e_1) = \tfrac{1}{2}(\sigma^2 \mp 2\sigma\alpha_1 + \sigma^2) = \sigma(\sigma \mp \alpha_1).$$

If a is close to a multiple of e_1, then $\sigma \approx |\alpha_1|$ and cancellation may lead to a large relative error in β. To avoid this we take

$$u = a + \operatorname{sign}(\alpha_1)\sigma e_1, \qquad 1/\beta = \sigma(\sigma + |\alpha_1|), \tag{8.1.56}$$

which gives

$$Pa = -\operatorname{sign}(\alpha_1)\sigma e_1 = \hat{\sigma} e_1.$$

Note that with this choice of sign the vector $a = e_1$ will be mapped onto $-e_1$. (It is possible to rewrite the formula in (8.1.56) for β so that the other choice of sign does not give rise to numerical cancellation; see Parlett [305, pp. 91].)

The Householder reflection in (8.1.52) does not depend on the scaling of u. It is often more convenient to scale u so that its first component equals 1. If we write

$$P = I - \beta u u^T, \quad u = \begin{pmatrix} 1 \\ u_2 \end{pmatrix},$$

then

$$\beta = 1 + |\alpha_1|/\sigma. \qquad u_2 = \rho a_2, \quad \rho = \operatorname{sign}(\alpha_1)/(\sigma + |\alpha_1|), \tag{8.1.57}$$

This has the advantage that we can stably reconstruct β from u_2 using

$$\beta = 2/(u^T u) = 2/(1 + u_2^T u_2).$$

Algorithm 8.1.
The following algorithm constructs a Householder reflection $P = I - \beta uu^T$, where $u^T e_1 = 1$, such that $Pa = -sign(\alpha_1)\sigma e_1$.

```
function [u,beta,sigma] = house(a)
  % HOUSE computes a Householder reflection
  % P = I - beta*u*u' where u = [1; u2'] such that
  % P*a = -sign(a_1)*norm(a)*e_1;
  n = length(a);
  alpha = a(1);
  sigma = -sign(alpha)*norm(a);
  beta = 1 + abs(alpha/sigma);
  rho = sigma*beta;
  u = [1;a(2:n)/rho];
```

Householder reflection can be generalized to the complex case as shown by Wilkinson [387, pp. 49–50]. Consider a *unitary transformations* of the form

$$P = I - \frac{1}{\gamma}uu^H, \quad \gamma = \frac{1}{2}u^H u, \quad u \in \mathbf{C}^n. \tag{8.1.58}$$

It is easy to check that P is Hermitian, $P^H = P$, and unitary, $P^{-1} = P$. Let $x \in \mathbf{C}^n$ and u be such that $Px = ke_1$. Then $|k| = \|x\|_2$, but it is in general not possible to have k real. Since P is Hermitian $x^H Px = kx^H e_1$ must be real. If we denote the first component of x by $x_1 = e^{i\theta_1}|x_1|$, then u we can take

$$k = \|x\|_2 e^{i\theta_1}, \tag{8.1.59}$$

Then in (8.1.58) $u = x + ke_1$ and γ is given by

$$\gamma = \frac{1}{2}(\|x\|_2^2 + 2|k||x_1| + |k|^2) = \|x\|_2(\|x\|_2 + |x_1|). \tag{8.1.60}$$

Note that u differs from x only in its first component.

Another useful class of elementary orthogonal transformations are **plane rotations**. A rotation clockwise through an angle θ in $\mathbf{R}^2$ is represented by the matrix

$$G(\theta) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \qquad c = \cos\theta, \quad s = \sin\theta. \tag{8.1.61}$$

Note that $G^{-1}(\theta) = G(-\theta)$, and $\det G(\theta) = +1$. Such transformations are also known as **Givens rotations** after Wallace Givens, who used them to reduce matrices to simpler form in [167].

In $\mathbf{R}^n$ the matrix representing a rotation in the plane spanned by the unit vectors e_i and e_j, $i < j$, is the following rank two modification of the unit matrix

I_n

$$G_{ij}(\theta) = \begin{array}{cc} & \begin{array}{ccccccc} & & i & & j & & \end{array} \\ \begin{array}{c} \\ \\ i \\ \\ j \\ \\ \\ \end{array} & \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & & s & & \\ & & & \ddots & & & \\ & & -s & & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \end{array}. \qquad (8.1.62)$$

Premultiplying a vector $a = (\alpha_1, \dots, \alpha_n)^T$ by $G_{ij}(\theta)$ we get

$$G_{ij}(\theta)a = (\tilde{\alpha}_1, \dots, \tilde{\alpha}_n)^T, \qquad \tilde{\alpha}_k = \begin{cases} \alpha_k, & k \neq i, j; \\ c\alpha_i + s\alpha_j, & k = i; \\ -s\alpha_i + c\alpha_j, & k = j. \end{cases} \qquad (8.1.63)$$

Thus, a plane rotation may be multiplied into a vector at a cost of two additions and four multiplications. We can determine the rotation $G_{ij}(\theta)$ so that $\tilde{\alpha}_j$ becomes zero by taking

$$c = \alpha_i/\sigma, \qquad s = \alpha_j/\sigma, \qquad \sigma = (\alpha_i^2 + \alpha_j^2)^{1/2} \neq 0. \qquad (8.1.64)$$

To guard against possible overflow, the Givens rotation should be computed as in the following procedure:

Algorithm 8.2.

```
function [c,s,r] = givens(a,b)
% GIVENS computes c and s in a Givens rotation
% Given scalars a and b computes c and s in
% a Givens rotation such that
% 0 = -s*a + c*b, and r = c*a + s*b
if b == 0
   c = 1.0; s = 0.0; r = a;
else if abs(b) > abs(a)
   t = a/b; tt = sqrt(1+t*t);
   s = 1/tt; c = t*s; r = tt*b;
else
   t = b/a; tt = sqrt(1+t*t);
   c = 1/tt; s = t*c; r = tt*a;
end
```

No trigonometric functions are involved in constructing a Givens rotation, only a square root. Note that $-G(\theta)$ also zeros $\tilde{\alpha}_j$ so that c and s are only determined up to a common factor ± 1.

Example 8.1.5.

The polar representation of a complex number $z = x + i\,y$ can be computed by a function call `[c,s,r] = givens(x,y)`. This gives $z = |r|e^{i\theta}$, where $e^{i\theta} = z/|r|$, and

$$z = \begin{cases} r(c + i\,s) & \text{if } r \geq 0, \\ |r|(-c + i\,(-s)) & \text{if } r < 0. \end{cases}$$

Premultiplication of a matrix $A \in R^{m\times n}$ with a Givens rotation G_{ij} will only affect the two *rows* i and j in A, which are transformed according to

$$a_{ik} := ca_{ik} + sa_{jk}, \tag{8.1.65}$$

$$a_{jk} := -sa_{ik} + ca_{jk}, \quad k = 1 : n. \tag{8.1.66}$$

The product requires $4n$ multiplications and $2n$ additions. An analogous algorithm, which only affects *columns* i and j, exists for postmultiplying A with G_{ij}.

Givens rotations can be used in several different ways to construct an orthogonal matrix U such that $Ua = \pm\sigma e_1$. Let G_{1k}, $k = 2 : m$ be a sequence of Givens rotations, where G_{1k} is determined to zero the kth component in the vector a,

$$G_{1m} \ldots G_{13}G_{12}a = \sigma e_1.$$

Note that G_{1k} will not destroy previously introduced zeros. Another possible sequence is $G_{k-1,k}$, $k = m : -1 : 2$, where $G_{k-1,k}$ is chosen to zero the kth component. This demonstrates the flexibility of Givens rotations compared to reflectors.

Given's rotation are ubiquitous in matrix algorithms and used to transform a matrix to a more compact form. To illustrate the rotation pattern it is convenient to use a schematic diagram introduced by J. H. Wilkinson and which we call a **Wilkinson diagram**. The diagram below shows the zeroing of the (4,2) element in A by a rotation of rows 2 and 4.

$$\begin{array}{c} \\ \rightarrow \\ \\ \rightarrow \\ \\ \\ \end{array} \begin{pmatrix} \times & \times & \times & \times \\ \otimes & \times & \times & \times \\ \otimes & \times & \times & \times \\ \otimes & \otimes & \times & \times \\ \otimes & \times & \times & \times \\ \otimes & \times & \times & \times \end{pmatrix}.$$

In a Wilkinson diagram $\times$ stands for a (potential) nonzero element and $\otimes$ for a nonzero element that has been zeroed out. The arrows points to the rows that took part in the last rotation.

It is essential to note that the matrix G_{ij} is never explicitly formed, but represented by (i, j) and the two numbers c and s. When a large number of rotations need to be stored it is more economical to store just a single number, from which c and s can be retrieved in a numerically stable way. Since the formula $\sqrt{1 - x^2}$ is poor if $|x|$ is close to unity a slightly more complicated method than storing just c

or s is needed. In a scheme devised by Stewart [341] one stores the number c or s of smallest magnitude. To distinguish between the two cases one stores the reciprocal of c. More precisely, if $c \neq 0$ we store

$$\rho = \begin{cases} s, & \text{if } |s| < |c|; \\ 1/c, & \text{if } |c| \le |s| \end{cases}.$$

In case $c = 0$ we put $\rho = 1$, a value that cannot appear otherwise.

To reconstruct the Givens rotation, if $\rho = 1$, we take $s = 1$, $c = 0$, and

$$\rho = \begin{cases} sF = \rho, & c = \sqrt{1 - s^2}, \quad \text{if } |\rho| < 1; \\ c = 1/\rho, & s = \sqrt{1 - c^2}, \quad \text{if } |\rho| > 1; \end{cases}$$

It is possible to rearrange the Givens rotations so that it uses only two instead of four multiplications per element and no square root. These modified transformations called "fast" Givens transformations, and are described in Golub and Van Loan [184, 1996, Sec. 5.1.13].

For complex matrices we need **unitary** Givens rotations, which are matrices of the form

$$G = \begin{pmatrix} \bar{c} & \bar{s} \\ -s & c \end{pmatrix}, \qquad c = e^{i\gamma} \cos\theta, \qquad s = e^{i\delta} \sin\theta. \tag{8.1.67}$$

From $\bar{c}c + \bar{s}s = \cos^2\theta + \sin^2\theta = 1$ it follows that $G^H G = I$, i.e., G is unitary. Given a complex vector $(x_1 \ x_2)^T \in \mathbf{C}^2$ we want to choose c and s so that

$$G \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \bar{c}z_1 + \bar{s}z_2 \\ -sz_1 + cz_2 \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix}, \qquad \sigma^2 = |z_1|^2 + |z_2|^2. \tag{8.1.68}$$

This holds provided that

$$c = z_1/\sigma, \qquad s = z_2/\sigma.$$

8.1.7 Angles Between Subspaces and the CS Decomposition

In many applications the relationship between two given subspaces needs to be investigated. For example, in statistical models canonical correlations measure how "close" two set of observations are.

This and similar questions can be answered by computing angles between subspaces. Let F and G be subspaces of $\mathbf{C}^n$ and assume that

$$p = \dim(F) \ge \dim(G) = q \ge 1.$$

The smallest angle $\theta_1(F, g) \in [0, \pi/2]$ between F and G is defined by

$$\cos\theta_1 = \max_{u \in F} \max_{v \in G} u^H v, \quad \|u\|_2 = \|v\|_2 = 1.$$

Assume that the maximum is attained for $u = u_1$ and $v = v_1$. Then $\theta_2(F, g)$ is defined as the smallest angle between the orthogonal complement of F with respect to u_1 and that of G with respect to v_1. Continuing in this way until one of the subspaces is empty, we are led to the following definition:

Definition 8.1.19.

The **principal angles** $\theta_k \in [0, \pi/2]$ *between two subspaces of* $\mathbf{C}^n$ *are recursively defined for* $k = 1 : q$, *by*

$$\cos\theta_k = \max_{u \in F} \max_{v \in G} u^H v = u_k^H v_k, \quad \|u\|_2 = \|v\|_2 = 1, \tag{8.1.69}$$

subject to the constraints

$$u^H u_j = 0, \qquad v^H v_j = 0, \quad j = 1 : k-1.$$

The vectors u_k *and* v_k, $k = 1 : q$, *are called* **principal vectors** *of the pair of spaces.*

The principal vectors are not always uniquely defined, but the principal angles are. The vectors $V = (v_1, \dots, v_q)$ form a unitary basis for G and the vectors $U = (u_1, \dots, u_q)$ can be complemented with $(p - q)$ unitary vectors so that $(u_1, \dots, u_p)$ form a unitary basis for F. It will be shown that it also holds that

$$u_j^H v_k = 0, \quad j \neq k, \quad j = 1 : p, \quad k = 1 : q.$$

In the following we assume that the subspaces F and G are defined as the range of two unitary matrices $Q_A \in \mathbf{C}^{n\times p}$ and $Q_B \in \mathbf{C}^{n\times q}$. The following theorem shows the relation between the SVD of the matrix $Q_A^H Q_B$ and the angles between the subspaces.

Theorem 8.1.20.

Assume that the columns of $Q_A \in \mathbf{C}^{n\times p}$ *and* $Q_B \in \mathbf{C}^{n\times q}$, $p \geq q$, *form unitary bases for two subspaces of* $\mathbf{C}^n$. *Let the thin SVD of the matrix* $M = Q_A^H Q_B \in \mathbf{C}^{p\times q}$ *be*

$$M = YCZ^H, \quad C = \operatorname{diag}(\sigma_1, \dots, \sigma_q), \tag{8.1.70}$$

where $y^H Y = Z^H Z = ZZ^H = I_q$ *and* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$. *Then the principal angles and principal vectors associated with this pair of subspaces are given by*

$$\cos\theta_k = \sigma_k, \qquad U = Q_A Y, \qquad V = Q_B Z. \tag{8.1.71}$$

Proof. The singular values and vectors of M can be characterized by the property

$$\sigma_k = \max_{\|y\|_2 = \|z\|_2 = 1} y^H M z = y_k^H M z_k, \tag{8.1.72}$$

subject to $y^H yj = z^H z_j = 0$, $j = 1 : k$. If we put $u = Q_A y \in F$ and $v = Q_B z \in G$ then it follows that $\|u\|_2 = \|y\|_2$, $\|v\|_2 = \|z\|_2$, and

$$u^H u_j = y^H y_j, \qquad v^H v_j = z^H z_j.$$

Since $y^H M z = y^H Q_A^H Q_B z = u^H v$, (8.1.72) is equivalent to

$$\sigma_k = \max_{\|u\|_2 = \|v\|_2 = 1} u_k^H v_k,$$

subject to $u^H u_j = 0$, $v^H v_j = 0$, $j = 1 : k-1$. Now (8.1.71) follows directly from definition 8.1.19. $\square$

In principle, a unitary basis for the *intersection of two subspaces* is obtained by taking the vectors u_k that corresponding to $\theta_k = 0$ or $\sigma_k = 1$. However, numerically small angles θ_k are well defined from $\sin\theta_k$ but not from $\cos\theta_k$. We now show how to compute $\sin\theta_k$.

We now change the notations slightly and write the SVD in (8.1.70) and the principal vectors as

$$M = Y_A C Y_B^H, \qquad U_A = Q_A Y_A, \quad U_B = Q_B Y_B.$$

Since Q_A is unitary it follows that $P_A = Q_A Q_A^H$ is the orthogonal projector onto F. Then we have

$$P_A Q_B = Q_A Q_A^H Q_B = Q_A M = U_A C Y_B. \tag{8.1.73}$$

Squaring $Q_B = P_A Q_B + (I - P_A) Q_B$, using (8.1.73) and $P_A(I - P_A) = 0$ gives

$$Q_B^H (I - P_A)^2 Q_B = Y_B (I - C^2) Y_B^H,$$

which shows that the SVD of $(I - P_A) Q_B = Q_B - Q_A M$ can be written

$$(I - P_A) Q_B = W_A S Y_B^H, \quad S^2 = I - C^2,$$

and thus $S = \pm\mathrm{diag}\,(\sin\theta_k)$.

We assume for convenience in the following that $p + q \le n$. Then the matrix $W_A \in \mathbf{R}^{n\times q}$ can be chosen so that $W_A^H U_A = 0$.

$$(I - P_B) Q_A = Q_A - Q_B M = W_B S Y_A^H. \tag{8.1.74}$$

Combining this with $P_A Q_B = U_A C Y_B^H$ we can write

$$U_B = Q_B Y_A = (U_A C + W_A S) = (U_A \ W_A) \begin{pmatrix} C \\ S \end{pmatrix}.$$

If we put

$$P_{A,B} = U_B U_A^H = (U_A \ W_A) \begin{pmatrix} C \\ S \end{pmatrix} U_A^H$$

then the transformation $y = P_{A,B} x$, rotates a vector $x \in R(A)$ into a vector $y \in R(B)$, and $\|y\|_2 = \|x\|_2$.

By analogy we also have the decomposition

$$(I - P_A) Q_B = Q_B - Q_A M = W_A S Y_B^H. \tag{8.1.75}$$

The CS decomposition is a special case a decomposition of a partitioned orthogonal matrix related to the SVD.

Theorem 8.1.21 (*Thin CS Decomposition*).

Let $Q_1 \in \mathbf{R}^{(m\times n)}$ have orthonormal columns, that is $Q_1^T Q_1 = I$, and be partitioned as

$$Q_1 = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} \begin{matrix} \}m_1 \\ \}m_2 \end{matrix}, \tag{8.1.76}$$

where $m_1 \geq n$, and $m_2 \geq n$. Then there are orthogonal matrices $U_1 \in \mathbf{R}^{m_1\times m_1}$, $U_2 \in \mathbf{R}^{m_2\times m_2}$, and $V_1 \in \mathbf{R}^{n\times n}$, and square nonnegative diagonal matrices

$$C = \operatorname{diag}(c_1, \dots, c_n), \quad S = \operatorname{diag}(s_1, \dots, s_n), \tag{8.1.77}$$

satisfying $C^2 + S^2 = I_n$ such that

$$\begin{pmatrix} U_1 & 0 \\ 0 & U_2 \end{pmatrix}^T \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} V_1 = \begin{pmatrix} U_1^T Q_{11} V_1 \\ U_2^T Q_{21} V_1 \end{pmatrix} = \begin{pmatrix} C \\ S \end{pmatrix} \begin{matrix} \}m \\ \}p \end{matrix} \tag{8.1.78}$$

The diagonal elements in C and S are

$$c_i = \cos(\theta_i), \quad s_i = \sin(\theta_i), \quad i = 1 : n,$$

where without loss of generality, we may assume that

$$0 \leq \theta_1 \leq \theta_2 \leq \cdots \leq \theta_n \leq \pi/2.$$

Proof. To construct U_1, V_1, and C, note that since U_1 and V_1 are orthogonal and C is a nonnegative diagonal matrix, $Q_{11} = U_1 C V_1^T$ is the SVD of Q_{11}. Hence, the elements c_i are the singular values of Q_{11}, and since $\|Q_{11}\|_2 \leq \|Q\|_2 = 1$, we have $c_i \in [0, 1]$.

If we put $\tilde{Q}_{21} = Q_{21} V_1$, then the matrix

$$\begin{pmatrix} C \\ 0 \\ \tilde{Q}_{21} \end{pmatrix} = \begin{pmatrix} U_1^T & 0 \\ 0 & I_{m_2} \end{pmatrix} \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} V_1$$

has orthonormal columns. Thus, $C^2 + \tilde{Q}_{21}^T \tilde{Q}_{21} = I_n$, which implies that $\tilde{Q}_{21}^T \tilde{Q}_{21} = I_n - C^2$ is diagonal and hence the matrix $\tilde{Q}_{21} = (\tilde{q}_1^{(2)}, \dots, \tilde{q}_n^{(2)})$ has orthogonal columns.

We assume that the singular values $c_i = \cos(\theta_i)$ of Q_{11} have been ordered according to (8.1.21) and that $c_r < c_{r+1} = 1$. Then the matrix $U_2 = (u_1^{(2)}, \dots, u_p^{(2)})$ is constructed as follows. Since $\|\tilde{q}_j^{(2)}\|_2^2 = 1 - c_j^2 \neq 0, \quad j \leq r$ we take

$$u_j^{(2)} = \tilde{q}_j^{(2)} / \|\tilde{q}_j^{(2)}\|_2, \qquad j = 1, \dots, r,$$

and fill the possibly remaining columns of U_2 with orthonormal vectors in the orthogonal complement of $\mathcal{R}(\tilde{Q}_{21})$. From the construction it follows that $U_2 \in \mathbf{R}^{m_2\times m_2}$ is orthogonal and that

$$U_2^T \tilde{Q}_{21} = U_2 Q_{21} V_1 = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}, \qquad S = \operatorname{diag}(s_1, \dots, s_q)$$

with $s_j = (1 - c_j^2)^{1/2} > 0$, if $j = 1 : r$, and $s_j = 0$, if $j = r + 1 : n$. □

In the theorem above we assumed that $n \le m/2m$. The general case gives rise to four different forms corresponding to cases where Q_{11} and/or Q_{21} have too few rows to accommodate a full diagonal matrix of order n.

The proof of the CS decomposition is constructive. In particular, U_1, V_1, and C can be computed by a standard SVD algorithm. However, the above algorithm for computing U_2 is unstable when some singular values c_i are close to 1. and needs to be modified.

Using the same technique the following CS decomposition of a square partitioned orthogonal matrix can be shown.

Theorem 8.1.22 (*Full CS Decomposition*).

Let

$$Q = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \mathbf{R}^{m\times m}. \tag{8.1.79}$$

be an arbitrary partitioning of the orthogonal matrix Q. Then there are orthogonal matrices

$$\begin{pmatrix} U_1 & 0 \\ 0 & U_2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} V_1 & 0 \\ 0 & V_2 \end{pmatrix}$$

such that

$$U^T Q V = \left(\begin{array}{c|c} U_1^T Q_{11} V_1 & U_1^T Q_{12} V_2 \\ \hline U_2^T Q_{21} V_1 & U_2^T Q_{22} V_2 \end{array}\right) = \left(\begin{array}{ccc|ccc} I & 0 & 0 & 0 & 0 & 0 \\ 0 & C & 0 & 0 & S & 0 \\ 0 & 0 & 0 & 0 & 0 & I \\ \hline 0 & 0 & 0 & I & 0 & 0 \\ 0 & S & 0 & 0 & -C & 0 \\ 0 & 0 & I & 0 & 0 & 0 \end{array}\right) \tag{8.1.80}$$

where $C = \operatorname{diag}(c_, \dots, c_q)$ and $S = \operatorname{diag}(s_, \dots, s_q)$ are diagonal matrices and

$$c_i = \cos(\theta_i), \quad s_i = \sin(\theta_i), \quad i = 1 : q,$$

Proof. For a proof, see Paige and Saunders [301]. □

Review Questions

1.1 State the Gauss–Markov theorem.

1.2 Assume that A has full column rank. Show that the matrix $P = A(A^TA)^{-1}A^T$ is symmetric and satisfies the condition $P^2 = P$.

1.3 (a) Give conditions for a matrix P to be the orthogonal projector onto a subspace $S \in \mathbf{R}^n$.

(b) Define the orthogonal complement of S in $\mathbf{R}^n$.

1.4 (a) Which are the four fundamental subspaces of a matrix? Which relations hold between them? Express the orthogonal projections onto the fundamental subspaces in terms of the SVD.

(b) Give two geometric conditions which are necessary and sufficient conditions for x to be the pseudo-inverse solution of $Ax = b$.

1.5 Which of the following relations are universally correct?

(a) $\mathcal{N}(B) \subseteq \mathcal{N}(AB)$. (b) $\mathcal{N}(A) \subseteq \mathcal{N}(AB)$. (c) $\mathcal{N}(AB) \subseteq \mathcal{N}(A)$.

(d) $\mathcal{R}(AB) \subseteq \mathcal{R}(B)$. (e) $\mathcal{R}(AB) \subseteq \mathcal{R}(A)$. (f) $\mathcal{R}(B) \subseteq \mathcal{R}(AB)$.

1.6 (a) What are the four Penrose conditions for X to be the pseudo-inverse of A?

(b)A matrix X is said to be a **left-inverse** if $XA = I$. Show that a left-inverse is an $\{1, 2, 3\}$-inverse, i.e. satisfies the Penrose conditions (1), (2), and (3). Similarly, show that a **right-inverse** is an $\{1, 2, 4\}$-inverse.

1.7 Let the singular values of $A \in \mathbf{R}^{m\times n}$ be $\sigma_1 \geq \cdots \geq \sigma_n$. What relations are satisfied between these and the singular values of

$$\tilde{A} = (A, u), \qquad \hat{A} = \begin{pmatrix} A \\ v^T \end{pmatrix}?$$

1.8 (a) Show that $A^\dagger = A^{-1}$ when A is a nonsingular matrix.

(b) Construct an example where $G \neq A^\dagger$ despite the fact that $GA = I$.

Problems

1.1 (a) Compute the pseudo-inverse $x^\dagger$ of a column vector x.

(b) Take $A = (\,1 \quad 0\,)$, $B = (\,1 \quad 1\,)^T$, and show that $1 = (AB)^\dagger \neq B^\dagger A^\dagger = 1/2$.

1.2 (a) Verify that the Penrose conditions uniquely defines the matrix X. Do it first for $A = \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, and then transform the result to a general matrix A.

1.3 (a) Show that if $w \in \mathbf{R}^n$ and $w^T w = 1$, then the matrix $P(w) = I - 2ww^T$ is both symmetric and orthogonal.

(b) Given two vectors $x, y \in \mathbf{R}^n$, $x \neq y$, $\|x\|_2 = \|y\|_2$, then

$$P(w)x = y, \qquad w = (y - x)/\|y - x\|_2.$$

1.4 Let $S \subseteq \mathbf{R}^n$ be a subspace, P_1 and P_2 be orthogonal projections onto $S = \mathcal{R}(P_1) = \mathcal{R}(P_2)$. Show that $P_1 = P_2$, i.e., the orthogonal projection onto S is unique.

Hint: Show that for any $z \in \mathbf{R}^n$

$$\|(P_1 - P_2)z\|_2^2 = (P_1 z)^T (I - P_2)z + (P_2 z)^T (I - P_1)z = 0.$$

1.5 (R. E. Cline) Let A and B be any matrices for which the product AB is defined, and set

$$B_1 = A^\dagger AB, \qquad A_1 = AB_1 B_1^\dagger.$$

Show that $AB = AB_1 = A_1 B_1$ and that $(AB)^\dagger = B_1^\dagger A_1^\dagger$.

Hint: Use the Penrose conditions.

1.6 (a) Show that the matrix $A \in \mathbf{R}^{m\times n}$ has a **left inverse** $A^L \in \mathbf{R}^{n\times m}$, i.e., $A^L A = I$, if and only if $\operatorname{rank}(A) = n$. Although in this case $Ax = b \in \mathcal{R}(A)$ has a unique solution, the left inverse is not unique. Find the general form of Σ^L and generalize the result to A^L.

(b) Discuss the **right inverse** A^R in a similar way.

1.7 Show that $A^\dagger$ minimizes $\|AX - I\|_F$.

1.8 Prove *Bjerhammar's characterization* : Let A have full column rank and let B be any matrix such that $A^T B = 0$ and $(A \quad B)$ is nonsingular. Then $A^\dagger = X^T$ where

$$\begin{pmatrix} X^T \\ Y^T \end{pmatrix} = (A \quad B)^{-1}.$$

8.2 The Method of Normal Equations

8.2.1 Forming and Solving the Normal Equations

Consider the linear model

$$Ax = b + \epsilon, \qquad A \in \mathbf{R}^{m\times n}, \tag{8.2.1}$$

where ϵ has zero mean and variance-covariance matrix equal to $\sigma^2 I$. By the Gauss–Markov theorem the least squares estimate satisfies the normal equations $A^T Ax = A^T b$. After forming $A^T A$ and $A^T b$ the normal equations can be solved by symmetric Gaussian elimination (which Gauss did), or by computing the Cholesky factorization (due to [30])

$$A^T A = R^T R,$$

where R is upper triangular. We now discuss some details in the numerical implementation of this method. We defer treatment of rank deficient problems to later and assume throughout this section that the numerical rank of A equals n.

The first step is to compute the elements of the symmetric matrix $C = A^T A$ and the vector $d = A^T b$. If $A = (a_1, a_2, \dots, a_n)$ has been partitioned by columns, we can use the inner product formulation

$$c_{jk} = (A^T A)_{jk} = a_j^T a_k, \qquad d_j = (A^T b)_j = a_j^T b, \quad 1 \le j \le k \le n. \tag{8.2.2}$$

Since C is symmetric it is only necessary to compute and store its lower (or upper) triangular which requires $\frac{1}{2}mn(n+1)$ multiplications. Note that if $m \gg n$, then the number of elements $\frac{1}{2}n(n+1)$ in the upper triangular part of A^TA is much smaller than the number mn of elements in A. Hence, in this case the formation of A^TA and A^Tb can be viewed as a *data compression*!

The inner product formulation (8.2.2) accesses the data A and b column-wise. This may not always be suitable. For example, for large problems, where the matrix A is held in secondary storage, each column needs to be accessed many times. In an alternative row oriented algorithm outer product of the rows are accumulated Denoting by $\tilde{a}_i^T$, the ith row of A, $i = 1 : m$, we get

$$C = A^TA = \sum_{i=1}^{m} \tilde{a}_i\tilde{a}_i^T, \qquad d = A^Tb = \sum_{i=1}^{m} b_i\tilde{a}_i. \tag{8.2.3}$$

This only needs *one pass* through the data (A, b). Here A^TA is expressed as the sum of m matrices of rank one and A^Tb as a linear combination of the transposed rows of A. No more storage is needed than that for A^TA and A^Tb. This outer product form is also preferable if the matrix A is sparse; see the hint to Problem 7.6.1. Note that both formulas can be combined if we adjoin b to A and form

$$(A, b)^T(A, b) = \begin{pmatrix} A^TA & A^Tb \\ b^TA & b^Tb \end{pmatrix}.$$

The matrix $C = A^TA$ is symmetric, and if $\operatorname{rank}(A) = n$ also positive definite. Gauss solved the normal equations by symmetric Gaussian elimination, but computing the Cholesky factorization

$$C = A^TA = R^TR, \quad R \in \mathbf{R}^{n\times n}, \tag{8.2.4}$$

is now the standard approach. The Cholesky factor R is upper triangular and nonsingular and can be computed by one of the algorithms given in Sec. 7.4.2. The least squares solution is then obtained by solving the two triangular systems

$$R^Tz = d, \qquad Rx = z. \tag{8.2.5}$$

Forming and solving the normal equations requires (neglecting lower order terms) about $\frac{1}{2}mn^2 + \frac{1}{6}n^3$ flops. If we have several right hand sides b_i, $i = 1 : p$, then the Cholesky factorization need only be computed once. To solve for each new right hand side then only needs $mn + n^2$ additional flops.

Example 8.2.1.

Linear regression is the problem of fitting a linear model $y = \alpha + \beta x$ to a set of given points (x_i, y_i), $i = 1 : m$. This leads to a overdetermined linear system

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

Forming the normal equations we get

$$\begin{pmatrix} m & \sum_{i=1}^m x_i \\ \sum_{i=1}^m x_i & \sum_{i=1}^m x_i^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m y_i x_i \end{pmatrix}. \tag{8.2.6}$$

Eliminating α we obtain the "classical" formulas

$$\beta = \Big(\sum_{i=1}^m y_i x_i - m\bar{y}\bar{x} \Big) \Big/ \Big(\sum_{i=1}^m x_i^2 - m\bar{x}^2 \Big),$$

where

$$\bar{y} = \frac{1}{m} \sum_{i=1}^m y_i, \qquad \bar{x} = \frac{1}{m} \sum_{i=1}^m x_i. \tag{8.2.7}$$

are the mean values. The first equation in (8.2.6) gives

$$\bar{y} = \alpha + \beta\bar{x}. \tag{8.2.8}$$

which shows that $(\bar{y}, \bar{x})$ lies on the fitted line. This determines $\alpha = \bar{y} - \beta\bar{x}$.

A more accurate formula for β is obtained by first subtracting out the mean values from the data. We have

$$(y - \bar{y}) = \beta(x - \bar{x})$$

In the new variables the matrix of normal equation is *diagonal*. and we find

$$\beta = \sum_{i=1}^m (y_i - \bar{y})(x_i - \bar{x})_i \Big/ \sum_{i=1}^m (x_i - \bar{x})^2. \tag{8.2.9}$$

A drawback of this formula is that it requires two passes through the data.

Computing the Variance-Covariance Matrix.

From the Gauss–Markov Theorem the covariance matrix of the solution x is $V_x = \sigma^2 C_x$, where

$$C_x = (A^T A)^{-1} = (R^T R)^{-1} = R^{-1} R^{-T}, \tag{8.2.10}$$

and R is the Cholesky factor of $A^T A$. To compute C_x we compute the upper triangular matrix $S = R^{-1}$, which satisfies the triangular matrix equation $RS = I$. It can be computed in $n^3/3$ flops by the algorithm given in (7.2.40). We then form $C_x = SS^T$, which is symmetric. Therefore, only its upper triangular part is needed. This takes $n^3/3$ flops and can be sequenced so that the elements of C_x overwrite those of S.

An unbiased estimate of σ^2 is given by

$$s^2 = \|b - Ax\|_2^2/(m - n).. \tag{8.2.11}$$

In order to assess the accuracy of the computed estimate of x it is often required to compute the matrix C_x or part of it. The least squares residual vector $r = b - A\hat{x}$ has variance-covariance matrix equal to $\sigma^2 V_r$, where

$$V_r = A(A^T A)^{-1} A^T = P_{\mathcal{R}(A)}. \tag{8.2.12}$$

The **normalized residuals**

$$\tilde{r} = \frac{1}{s}(\operatorname{diag} V_r)^{-1/2}\hat{r}$$

are often used to detect and identify single or multiple bad data, which is assumed to correspond to large components in $\tilde{r}$.

In many situations the matrix C_x only occurs as an intermediate quantity in a formula. For example, the variance of a linear functional $\varphi = f^T\hat{x}$ is equal to

$$\mathcal{V}(\varphi) = f^T V_x f = \sigma^2 f^T R^{-1}R^{-T} f = \sigma^2 z^T z, \tag{8.2.13}$$

where $z = R^{-T}f$. Thus, the variance may be computed by solving by forward substitution the lower triangular system $R^T z = f$ and forming $\sigma^2 z^T z = \sigma^2\|z\|_2^2$. This is a more stable and efficient approach than using the expression involving V_x. In particular, the variance of the component $x_i = e_i^T x$ is obtained by solving

$$R^T z = e_i.$$

Note that since R^T is lower triangular, z will have $i-1$ leading zeros. For $i = n$ only the last component is nonzero and equals r_{nn}^{-1}.

There is an alternative way of computing C_x without inverting R. We have from (8.2.13), multiplying by R from the left,

$$RC_x = R^{-T}. \tag{8.2.14}$$

The diagonal elements of R^{-T} are simply r_{kk}^{-1}, $k = n,\ldots,1$, and since R^{-T} is lower triangular it has $\frac{1}{2}n(n-1)$ zero elements. Hence, $\frac{1}{2}n(n+1)$ elements of R^{-T} are known and the corresponding equations in (8.2.14) suffice to determine the elements in the upper triangular part of the symmetric matrix C_x.

To compute the elements in the last column c_n of C_x we solve the system

$$Rc_n = r_{nn}^{-1}e_n, \quad e_n = (0,\ldots,0,1)^T$$

by back-substitution. This gives

$$c_{nn} = r_{nn}^{-2}, \quad c_{in} = -r_{ii}^{-1}\sum_{j=i+1}^{n} r_{ij}c_{jn}, \quad i = n-1,\ldots,1. \tag{8.2.15}$$

By symmetry $c_{ni} = c_{in}, \quad i = n-1,\ldots,1$, so we also know the last row of C_x. Now assume that we have computed the elements $c_{ij} = c_{ji}$, $j = n,\ldots,k+1$, $i \le j$. We next determine the elements c_{ik}, $i \le k$. We have

$$c_{kk}r_{kk} + \sum_{j=k+1}^{n} r_{kj}c_{jk} = r_{kk}^{-1},$$

and since the elements $c_{kj} = c_{jk}$, $j = k+1:n$, have already been computed,

$$c_{kk} = r_{kk}^{-1}\Bigl(r_{kk}^{-1} - \sum_{j=k+1}^{n} r_{kj}c_{kj}\Bigr). \tag{8.2.16}$$

Similarly, for $i = k-1 : (-1) : 1$,

$$c_{ik} = -r_{ii}^{-1}\left(\sum_{j=i+1}^{k} r_{ij}c_{jk} + \sum_{j=k+1}^{n} r_{ij}c_{kj}\right). \tag{8.2.17}$$

Using the formulas (8.2.15)–(8.2.17) all the elements of C_x can be computed in about $2n^3/3$ flops.

When the matrix R is sparse, Golub and Plemmons [168] have shown that the same algorithm can be used very efficiently to compute *all elements in* C_x, *associated with nonzero elements in* R. Since R has a nonzero diagonal this includes the diagonal elements of C_x giving the variances of the components x_i, $i = 1 : n$. If R has bandwidth w, then the corresponding elements in C_x can be computed in only $2nw^2$ flops; see Björck [40, Sec 6.7.4].

Example 8.2.2.

The French astronomer Bouvard[24] collected 126 observations of the movements of Jupiter and Saturn. These were used to estimate the mass of Jupiter and gave the normal equations

$$A^TA = \begin{pmatrix} 795938 & -12729398 & 6788.2 & -1959.0 & 696.13 & 2602 \\ -12729398 & 424865729 & -153106.5 & -39749.1 & -5459 & 5722 \\ 6788.2 & -153106.5 & 71.8720 & -3.2252 & 1.2484 & 1.3371 \\ -1959.0 & -153106.5 & -3.2252 & 57.1911 & 3.6213 & 1.1128 \\ 696.13 & -5459 & 1.2484 & 3.6213 & 21.543 & 46.310 \\ 2602 & 5722 & 1.3371 & 1.1128 & 46.310 & 129 \end{pmatrix},$$

$$A^Tb = \begin{pmatrix} 7212.600 \\ -738297.800 \\ 237.782 \\ -40.335 \\ -343.455 \\ -1002.900 \end{pmatrix}.$$

In these equations the mass of Uranus is $(1 + x_1)/19504$, the mass of Jupiter $(1 + x_2)/1067.09$. Laplace [253] 1820, working from these normal equations, computed the least squares estimate for x_2 and its variance.

In many least squares problems the matrix A has the property that in each row all nonzero elements in A are contained in a narrow band. For banded rectangular matrix A we define:

Definition 8.2.1.

For $A \in \mathbf{R}^{m\times n}$ let f_i and l_i be the column subscripts of the first and last nonzero in the ith row of A, i.e.,

$$f_i = \min\{j \mid a_{ij} \neq 0\}, \qquad l_i = \max\{j \mid a_{ij} \neq 0\}. \tag{8.2.18}$$

[24] Alexis Bouvard (1767–1843) French astronomer and director of the Paris Observatory.

Then the matrix A is said to have row bandwidth w, where

$$w = \max_{1\le i\le m} w_i, \qquad w_i = (l_i - f_i + 1). \tag{8.2.19}$$

Alternatively w is the smallest number for which it holds that

$$a_{ij}a_{ik} = 0, \quad \text{if} \quad |j-k| \ge w. \tag{8.2.20}$$

For this structure to have practical significance we need to have $w \ll n$. Matrices of small row bandwidth often occur naturally, since they correspond to a situation where only variables "close" to each other are coupled by observations. We now prove a relation between the row bandwidth of the matrix A and the bandwidth of the corresponding matrix of normal equations A^TA.

Theorem 8.2.2.
Assume that the matrix $A \in \mathbf{R}^{m\times n}$ has row bandwidth w. Then the symmetric matrix A^TA has bandwidth $r \le w-1$.

Proof. From the Definition 8.2.1 it follows that $a_{ij}a_{ik} \ne 0 \Rightarrow |j-k| < w$. Hence,

$$|j-k| \ge w \Rightarrow (A^TA)_{jk} = \sum_{i=1}^m a_{ij}a_{ik} = 0.$$

□

If the matrix A also has full column rank it follows that we can use the band Cholesky Algorithm 7.3.2 to solve the normal equations.

8.2.2 Recursive Least Squares.

In various least squares problems the solution has to be updated when data is added or deleted. Such modifications are usually referred to as **updating** when (new) data is added and **down-dating** when (old) data is removed. In time-series problems a "sliding window method" is often used. At each time step a new data row is added, and then the oldest data row deleted. In signal processing applications often require real-time solutions so efficiency is critical. Another instance when a data row has to be removed is when it has somehow been identified as faulty.

The solution to the least squares problem $\min_x \|Ax-b\|_2$ satisfies the normal equations $A^TAx = A^Tb$. If the equation $w^Tx = \beta$ is added, then the updated solution $\tilde{x}$ satisfies the modified normal equations

$$(A^TA + ww^T)\tilde{x} = A^Tb + \beta w, \tag{8.2.21}$$

where we assume that $\operatorname{rank}(A) = n$. We would like to avoid computing the Cholesky factorization of the modified problem from scratch. A straightforward method for computing $\tilde{x}$ is based on updating the matrix $C = (A^T A)^{-1} = R^{-1}R^{-T}$. Since $\tilde{C}^{-1} = C^{-1} + ww^T$, we have by the Sherman–Morrison formula (7.1.25)

$$\tilde{C} = C - \frac{1}{1 + w^T u} uu^T, \quad u = Cw. \tag{8.2.22}$$

Adding the term $ww^T x$ to both sides of the unmodified normal equations and subtracting from (8.2.21) gives

$$(A^T A + ww^T)(\tilde{x} - x) = (\beta - w^T x)w.$$

Solving for the updated solution gives the following basic formula:

$$\tilde{x} = x + (\beta - w^T x)\tilde{u}, \quad \tilde{u} = \tilde{C}w. \tag{8.2.23}$$

Since the matrix C is the scaled covariance method matrix this is called a **covariance matrix method**.

Equations (8.2.22)–(8.2.23) define a **recursive least squares** algorithm associated with the Kalman filter. The vector $\tilde{u} = \tilde{C}w$, which weights the predicted residual $\beta - w^T x$ of the new observation, is called the **Kalman gain vector**.

The equations (8.2.22)–(8.2.23) can, with slight modifications, be used also for *deleting* an observation $w^T x = \beta$. We have

$$\tilde{C} = C + \frac{1}{1 - w^T u} uu^T, \qquad \tilde{x} = x - (\beta - w^T x)\tilde{u}, \tag{8.2.24}$$

provided that $1 - w^T u \neq 0$.

The simplicity and recursive nature of this updating algorithm has made it popular for many applications. The main disadvantage of the algorithm is its serious sensitivity to roundoff errors. The updating algorithms based on orthogonal transformations developed in Sec. 8.4.1 are therefore generally to be preferred.

The method can also be used together with updating schemes for the Cholesky factor factor R or its inverse R^{-1}, where $A^T A = R^T R$. The Kalman gain vector can then be computed from

$$z = \tilde{R}^{-T} w, \qquad p = \tilde{R}^{-1} z.$$

Such methods are often referred to as "square root methods" in the signal processing literature. Schemes for updating R^{-1} are described in [40, Sec. 3.3]. Since no backsubstitutions is needed in these schemes, they are easier to parallelize.

8.2.3 Perturbation Bounds for Least Squares Problems

We now consider the effect of perturbations of A and b on the least squares solution x. In this analysis the condition number of the matrix $A \in \mathbf{R}^{m \times n}$ will play a significant role. The following definition generalizes the condition number (6.6.3) of a square nonsingular matrix.

Definition 8.2.3.

Let $A \in \mathbf{R}^{m\times n}$ have rank $r > 0$ and singular values equal to $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. Then the condition number of A is

$$\kappa(A) = \|A\|_2 \|A^\dagger\|_2 = \sigma_1/\sigma_r,$$

where the last equality follows from the relations $\|A\|_2 = \sigma_1$, $\|A^\dagger\|_2 = \sigma_r^{-1}$.

Using the singular value decomposition $A = U\Sigma V^T$ we obtain

$$A^TA = V\Sigma^T(U^TU)\Sigma V^T = V \begin{pmatrix} \Sigma_r^2 & 0 \\ 0 & 0 \end{pmatrix} V^T. \tag{8.2.25}$$

Hence, $\sigma_i(A^TA) = \sigma_i^2(A)$, and it follows that

$$\kappa(A^TA) = \kappa^2(A).$$

This shows that *the matrix of the normal equations has a condition number which is the square of the condition number of A.*

We now give a first order perturbation analysis for the least squares problem when $\operatorname{rank}(A) = n$. Denote the perturbed data $A + \delta A$ and $b + \delta b$ and assume that δA sufficiently small so that $\operatorname{rank}(A + \delta A) = n$. Let the perturbed solution be $x + \delta x$ and $r + \delta r$, where $r = b - Ax$ is the residual vector. The perturbed solution satisfies the augmented system

$$\begin{pmatrix} I & A + \delta A \\ (A + \delta A)^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{s} \\ \tilde{x} \end{pmatrix} = \begin{pmatrix} b + \delta b \\ 0 \end{pmatrix}. \tag{8.2.26}$$

Subtracting the unperturbed equations and neglecting second order quantities the perturbations $\delta s = \delta r$ and δx satisfy the linear system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \delta s \\ \delta x \end{pmatrix} = \begin{pmatrix} \delta b - \delta A x \\ -\delta A^T s \end{pmatrix}. \tag{8.2.27}$$

From the Schur–Banachiewicz formula (see Sec. 7.1.5) it follows that the inverse of the matrix in this system equals

$$\begin{aligned} \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}^{-1} &= \begin{pmatrix} (I - A(A^TA)^{-1}A^T) & A(A^TA)^{-1} \\ (A^TA)^{-1}A^T & -(A^TA)^{-1} \end{pmatrix} \\ &= \begin{pmatrix} P_{\mathcal{N}(A^T)} & -(A^\dagger)^T \\ A^\dagger & -(A^TA)^{-1} \end{pmatrix}. \end{aligned} \tag{8.2.28}$$

We find that the perturbed solution satisfies

$$\delta x = A^\dagger(\delta b - \delta A\, x) + (A^TA)^{-1}\delta A^T r, \tag{8.2.29}$$

$$\delta r = P_{\mathcal{N}(A^T)}(\delta b - \delta A x) - (A^\dagger)^T \delta A^T\, r. \tag{8.2.30}$$

Using (8.1.33) and (8.2.25) it follows that

$$\|A^\dagger\|_2 = \|(A^\dagger)^T\|_2 = 1/\sigma_n, \qquad \|(A^TA)^{-1}\|_2 = 1/\sigma_n^2, \qquad \|P_{\mathcal{N}(A^T)}\|_2 = 1.$$

Hence, taking norms in (8.2.29) and (8.2.30) we obtain

$$\|\delta x\|_2 \lessapprox \frac{1}{\sigma_n}\|\delta b\|_2 + \frac{1}{\sigma_n}\|\delta A\|_2 \left(\|x\|_2 + \frac{1}{\sigma_n}\|r\|_2\right), \tag{8.2.31}$$

$$\|\delta r\|_2 \lessapprox \|\delta b\|_2 + \|\delta A\|_2 \left(\|x\|_2 + \frac{1}{\sigma_n}\|r\|_2\right), \tag{8.2.32}$$

Note that if the system $Ax = b$ is consistent, then $r = 0$ and the bound is identical to that obtained for a square nonsingular linear system. Otherwise, there is a second term present in the perturbation bound. A more refined perturbation analysis (see Wedin [384]) shows that if

$$\eta = \|A^\dagger\|_2\|\delta A\|_2 \ll 1.$$

then $\operatorname{rank}(A + \delta A) = n$, and there are perturbations δA and δb such that these upper bounds are almost attained.

Assuming that $x \neq 0$ and setting $\delta b = 0$, we get an upper bound for the normwise relative perturbation

$$\frac{\|\delta x\|_2}{\|x\|_2} \leq \kappa_{LS}\frac{\|\delta A\|_2}{\|A\|_2}, \quad \kappa_{LS} = \kappa(A)\Big(1 + \frac{\|r\|_2}{\sigma_n\|x\|_2}\Big) \tag{8.2.33}$$

Hence, κ_{LS} is the condition number for the least squares problem. The following two important facts should be noted:

- κ_{LS} depends not only on A but also on r and therefore on b;
- If $\|r\|_2 \ll \sigma_n\|x\|_2$ then $\kappa_{LS} \approx \kappa(A)$, but if $\|r\|_2 > \sigma_n\|x\|_2$ the second term in (8.2.33) will dominate,

Example 8.2.3. The following simple example illustrates the perturbation analysis above. Consider a least squares problem with

$$A = \begin{pmatrix} 1 & 0 \\ 0 & \delta \\ 0 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 0 \\ \alpha \end{pmatrix}, \qquad \delta A = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \delta/2 \end{pmatrix}.$$

and $\kappa(A) = 1/\delta \gg 1$. If $\alpha = 1$ then

$$x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \delta x = \frac{2}{5\delta}\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad r = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \delta r = -\frac{1}{5}\begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}.$$

For this right hand side $\|x\|_2 = \|r\|_2$ and $\kappa_{LS} = 1/\delta + 1/\delta^2 \approx \kappa^2(A)$. This is reflected in the size of δx.

If instead we take $\alpha = \delta$, then a short calculation shows that $\|r\|_2/\|x\|_2 = \delta$ and $\kappa_{LS} = 2/\delta$. The same perturbation δA now gives

$$\delta x = \frac{2}{5}\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad \delta r = -\frac{\delta}{5}\begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}.$$

It should be stressed that in order for the perturbation analysis above to be useful, the matrix A and vector b should be scaled so that perturbations are "well defined" by bounds on $\|\delta A\|_2$ and $\|b\|_2$. It is not uncommon that the columns in $A = (a_1, a_2, \dots, a_n)$ have widely differing norms. Then a much better estimate may often be obtained by applying (8.2.33) to the scaled problem $\min_{\tilde{x}} \|\tilde{A}\tilde{x} - b\|_2$, chosen so that $\tilde{A}$ has columns of unit length, i.e.,

$$\tilde{A} = AD^{-1}, \qquad \tilde{x} = Dx, \qquad D = \mathrm{diag}(\|a_1\|_2, \dots, \|a_n\|_2).$$

By Theorem 8.2.4 this column scaling approximately minimizes $\kappa(AD^{-1})$ over $D > 0$. Note that scaling the columns also changes the norm in which the error in the original variables x is measured.

If the *rows* in A differ widely in norm, then (8.2.33) may also considerably overestimate the perturbation in x. As remarked above, we cannot scale the rows in A without changing the least squares solution.

Perturbation bounds with better scaling properties can be obtained by considering component-wise perturbations; see Sec. 8.2.4.

8.2.4 Stability and Accuracy with Normal Equations

We now turn to a discussion of the accuracy of the method of normal equations for least squares problems. First we consider rounding errors in the formation of the system of normal equations. Using the standard model for floating point computation we get for the elements $\bar{c}_{ij}$ in the computed matrix $\bar{C} = fl(A^TA)$

$$\bar{c}_{ij} = fl\left(\sum_{k=1}^{m} a_{ik}a_{jk}\right) = \sum_{k=1}^{m} a_{ik}a_{jk}(1 + \delta_k),$$

where (see (2.4.4)) $|\delta_k| < 1.06(m + 2 - k)u$ (u is the machine unit). It follows that the computed matrix satisfies

$$\bar{C} = A^TA + E, \qquad |e_{ij}| < 1.06um\sum_{k=1}^{m} |a_{ik}||a_{jk}|. \tag{8.2.34}$$

A similar estimate holds for the rounding errors in the computed vector A^Tb. Note that it is *not* possible to show that $\bar{C} = (A + E)^T(A + E)$ for some small error matrix E, i.e., the rounding errors in forming the matrix A^TA are not in general equivalent to small perturbations of the initial data matrix A. From this we can deduce that *the method of normal equations is not backwards stable.* The following example illustrates that when A^TA is ill-conditioned, *it might be necessary to use double precision in forming and solving the normal equations in order to avoid loss of significant information.*

Example 8.2.4.

Läuchli [256]: Consider the system $Ax = b$, where

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \epsilon & & \\ & \epsilon & \\ & & \epsilon \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad |\epsilon| \ll 1.$$

We have, exactly

$$A^TA = \begin{pmatrix} 1+\epsilon^2 & 1 & 1 \\ 1 & 1+\epsilon^2 & 1 \\ 1 & 1 & 1+\epsilon^2 \end{pmatrix}, \qquad A^Tb = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$$x = \frac{1}{3+\epsilon^2} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^T, \qquad r = \frac{1}{3+\epsilon^2} \begin{pmatrix} \epsilon^2 & -1 & -1 & -1 \end{pmatrix}^T.$$

Now assume that $\epsilon = 10^{-4}$, and that we use eight-digit decimal floating point arithmetic. Then $1 + \epsilon^2 = 1.00000001$ rounds to 1, and the computed matrix A^TA will be singular. We have lost all information contained in the last three rows of A! Note that the residual in the first equation is $O(\epsilon^2)$ but $O(1)$ in the others.

Least squares problems of this form occur when the error in some equations (here $x_1 + x_2 + x_3 = 1$) have a much smaller variance than in the others; see Sec. 8.6.1.

To assess the error in the least squares solution $\bar{x}$ computed by the method of normal equations, we must also account for rounding errors in the Cholesky factorization and in solving the triangular systems. Using Theorem 6.6.6 and the perturbation bound in Theorem 6.6.2 it can be shown that provided that $2n^{3/2}u\kappa(A^TA) < 0.1$, the error in the computed solution $\bar{x}$ satisfies

$$\|\bar{x} - x\|_2 \le 2.5n^{3/2}u\kappa(A^TA)\|x\|_2. \tag{8.2.35}$$

As seen in Sec. 8.2.4, for "small" residual least squares problem the true condition number is approximately $\kappa(A) = \kappa^{1/2}(A^TA)$. In this case *the system of normal equations can be much worse conditioned than the least squares problem from which it originated.*

Sometimes ill-conditioning is caused by an unsuitable formulation of the problem. Then a different choice of parameterization can significantly reduce the condition number. For example, in approximation problems one should try to use orthogonal, or nearly orthogonal, base functions. In case the elements in A and b are the original data the ill-conditioning cannot be avoided in this way.

In statistics the linear least squares problem $\min_x \|b - Ax\|_2$ derives from a **multiple linear regression** problem, where the vector b is a response variable and the columns of A contain the values of the explanatory variables.

In Secs. 8.3 and 8.4 we consider methods for solving least squares problems based on orthogonalization. These methods work directly with A and b and are backwards stable.

In Sec. 7.7.7 we discussed how the scaling of rows and columns of a linear system $Ax = b$ influenced the solution computed by Gaussian elimination. For a least squares problem $\min_x \|Ax - b\|_2$ a row scaling of $(A,\ b)$ is not allowed since such a scaling would change the exact solution. However, we can scale the columns of A. If we take $x = Dx'$, the normal equations will change into

$$(AD)^T(AD)x' = D(A^TA)Dx' = DA^Tb.$$

Hence, this corresponds to a *symmetric scaling* of rows and columns in A^TA. It is important to note that if the Cholesky algorithm is carried out without pivoting the computed solution is *not* affected by such a scaling, cf. Theorem 7.5.6. This means that even if no explicit scaling is carried out, the rounding error estimate (8.2.35) for the computed solution $\bar{x}$ holds for *all* D,

$$\|D(\bar{x} - x)\|_2 \leq 2.5n^{3/2}u\kappa(DA^TAD)\|Dx\|_2.$$

(Note, however, that scaling the columns changes the norm in which the error in x is measured.)

Denote the *minimum* condition number under a symmetric scaling with a positive diagonal matrix by

$$\kappa'(A^TA) = \min_{D>0} \kappa(DA^TAD). \tag{8.2.36}$$

The following result by van der Sluis [1969] shows the scaling where D is chosen so that in AD all column norms are equal, i.e. $D = \mathrm{diag}(\|a_1\|_2, \ldots, \|a_n\|_2)^{-1}$, comes within a factor of n of the minimum value.

Theorem 8.2.4. *Let $C \in \mathbf{R}^{n\times n}$ be a symmetric and positive definite matrix, and denote by $\mathcal{D}$ the set of $n\times n$ nonsingular diagonal matrices. Then if in C all diagonal elements are equal, and C has at most q nonzero elements in any row, it holds that*

$$\kappa(C) \leq q \min_{D\in\mathcal{D}} \kappa(DCD).$$

As the following example shows, this scaling can reduce the condition number considerably. In cases where the method of normal equations gives surprisingly accurate solution to a seemingly very ill-conditioned problem, the explanation often is that the condition number of the scaled problem is quite small!

Example 8.2.5. The matrix $A \in R^{21\times 6}$ with elements

$$a_{ij} = (i-1)^{j-1}, \qquad 1 \leq i \leq 21, \quad 1 \leq j \leq 6$$

arises when fitting a fifth degree polynomial $p(t) = x_0 + x_1t + x_2t^2 + \ldots + x_5t^5$ to observations at points $x_i = 0, 1, \ldots, 20$. The condition numbers are

$$\kappa(A^TA) = 4.10 \cdot 10^{13}, \qquad \kappa(DA^TAD) = 4.93 \cdot 10^6.$$

where D is the column scaling in Theorem 8.2.4. Thus, the condition number of the matrix of normal equations is reduced by about seven orders of magnitude by this scaling!

8.2.5 Backward Error Analysis

An algorithm for solving the linear least squares problem is said to numerically stable if for any data A and b, there exist small perturbation matrices and vectors δA and δb, such that *the computed solution $\bar{x}$ is the exact solution to*

$$\min_x \|(A + \delta A)x - (b + \delta b)\|_2, \tag{8.2.37}$$

where $\|\delta A\| \leq \tau \|A\|$, $\|\delta b\| \leq \tau \|b\|$, with τ being a small multiple of the unit round-off u. We shall see that methods in which the normal equations are explicitly formed cannot be backward stable. On the other hand, many methods based on orthogonal factorizations have been proved to be backward stable.

Any computed solution $\bar{x}$ is called a stable solution if it satisfies (8.2.37). This does not mean that $\bar{x}$ is close to the exact solution x. If the least squares problem is ill-conditioned then a stable solution can be very different from x. For a stable solution the error $\|x - \bar{x}\|$ can be estimated using the perturbation results given in Section 8.2.4.

Many special fast methods exist for solving structured least squares problems, e.g., where A is a Toeplitz matrix. These methods cannot be proved to be backward stable, which is one reason why a solution to the following problem is of interest:

For a consistent linear system we derived in Sec. 7.6.4 a simple posteriori bounds for the the smallest backward error of a computed solution $\bar{x}$. The situation is more difficult for the least squares problem.

Given an alleged solution $\tilde{x}$, we want to find a perturbation δA of smallest norm such that $\tilde{x}$ is the exact solution to the perturbed problem

$$\min_x \|(b + \delta b) - (A + \delta A)x\|_2. \tag{8.2.38}$$

If we could find the backward error of smallest norm, this could be used to verify numerically the stability properties of an algorithm. There is not much loss in assuming that $\delta b = 0$ in (8.2.39). Then the optimal backward error in the Frobenius norm is

$$\eta_F(\tilde{x}) = \min\{\|\delta A\|_F \mid \tilde{x} \text{ solves } \min_x \|b - (A + \delta A)x\|_2\}. \tag{8.2.39}$$

This the optimal backward error can be found by characterizing the set of all backward perturbations and then finding an optimal bound, which minimizes the Frobenius norm.

Theorem 8.2.5. *Let $\tilde{x}$ be an alleged solution and $\tilde{r} = b - A\tilde{x} \neq 0$. The optimal backward error in the Frobenius norm is*

$$\eta_F(\tilde{x}) = \begin{cases} \|A^T\tilde{r}\|_2/\|\tilde{r}\|_2, & \textit{if } \tilde{x} = 0, \\ \min\{\eta, \sigma_{min}(\,[A \quad C]\,)\} & \textit{otherwise.} \end{cases} \tag{8.2.40}$$

where

$$\eta = \|\tilde{r}\|_2/\|\tilde{x}\|_2, \qquad C = I - (\tilde{r}\tilde{r}^T)/\|\tilde{r}\|_2^2$$

and $\sigma_{min}(\,[A \quad C])$ *denotes the smallest (nonzero) singular value of the matrix* $[A \quad C] \in \mathbb{R}^{m\times(n+m)}$.

The task of computing $\eta_F(\tilde{x})$ is thus reduced to that of computing $\sigma_{min}(\,[A \quad C]\,)$. Since this is expensive, approximations that are accurate and less costly have been derived. If a QR factorization of A is available lower and upper bounds for $\eta_F(\tilde{x})$ can be computed in only $\mathcal{O}(mn)$ operations. Let $r_1 = P_{\mathcal{R}(A)}\tilde{r}$ be the orthogonal projection of $\tilde{r}$ onto the range of A. If $\|r_1\|_2 \le \alpha\|r\|_2$ it holds that

$$\frac{\sqrt{5}-1}{2}\,\tilde{\sigma}_1 \le \eta_F(\tilde{x}) \le \sqrt{1+\alpha^2}\,\tilde{\sigma}_1, \tag{8.2.41}$$

where

$$\tilde{\sigma}_1 = \left\|(A^TA+\eta I)^{-1/2}A^T\tilde{r}\right\|_2/\|\tilde{x}\|_2. \tag{8.2.42}$$

Since $\alpha \to 0$ for small perturbations $\tilde{\sigma}_1$ is an asymptotic upper bound.

8.2.6 The Peters–Wilkinson method

Standard algorithms for solving nonsymmetric linear systems $Ax = b$ are usually based on LU factorization with partial pivoting. Therefore it seems natural to consider such factorizations in particular, for least squares problems which are only mildly over- or under-determined, i.e. where $|m-n| \ll n$.

A rectangular matrix $A \in \mathbf{R}^{m\times n}$, $m \ge n$, can be reduced by Gaussian elimination to an upper trapezoidal form U. In general, column interchanges are needed to ensure numerical stability. Usually it will be sufficient to use partial pivoting with a linear independence check. Let $\tilde{a}_{q,p+1}$ be the element of largest magnitude in column $p+1$. If $|\tilde{a}_{q,p+1}| < tol$, column $p+1$ is considered to be linearly dependent and is placed last. We then look for a pivot element in column $p+2$, etc.

In the full column rank case, $\operatorname{rank}(A) = n$, the resulting LDU factorization becomes

$$\Pi_1 A \Pi_2 = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = LDU = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} DU, \tag{8.2.43}$$

where $L_1 \in \mathbf{R}^{n\times n}$ is unit lower triangular, D diagonal, and $U \in \mathbf{R}^{n\times n}$ is unit upper triangular and nonsingular. Thus, the matrix L has the same dimensions as A and a lower trapezoidal structure. Computing this factorization requires $\frac{1}{2}n^2(m-\frac{1}{3}n)$ flops.

Using the LU factorization (8.2.43) and setting $\tilde{x} = \Pi_2^T x$, $\tilde{b} = \Pi_1 b$, the least squares problem $\min_x \|Ax-b\|_2$ is reduced to

$$\min_y \|Ly-\tilde{b}\|_2, \qquad DU\tilde{x} = y. \tag{8.2.44}$$

If partial pivoting by rows is used in the factorization (8.2.43), then L is usually a well-conditioned matrix. In this case the solution to the least squares problem (8.2.44) can be computed from the normal equations

$$L^TLy = L^T\tilde{b},$$

without substantial loss of accuracy. This is the approach taken by Peters and Wilkinson [309, 1970].

Forming the symmetric matrix L^TL requires $\frac{1}{2}n^2(m-\frac{2}{3}n)$ flops, and computing its Cholesky factorization takes $n^3/6$ flops. Hence, neglecting terms of order n^2, the total number of flops to compute the least squares solution by the Peters–Wilkinson method is $n^2(m-\frac{1}{3}n)$. Although this is always more expensive than the standard method of normal equations, it is a more stable method as the following example shows. It is particularly suitable for weighted least squares problems; see Sec. 8.6.1.

Example 8.2.6. (Noble [291, 1976])

Consider the matrix A and its pseudo-inverse

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1+\epsilon^{-1} \\ 1 & 1-\epsilon^{-1} \end{pmatrix}, \quad A^\dagger = \frac{1}{6}\begin{pmatrix} 2 & 2-3\epsilon^{-1} & 2+3\epsilon^{-1} \\ 0 & 3\epsilon^{-1} & -3\epsilon^{-1} \end{pmatrix}.$$

The (exact) matrix of normal equations is

$$A^TA = \begin{pmatrix} 3 & 3 \\ 3 & 3+2\epsilon^2 \end{pmatrix}.$$

If $\epsilon \le \sqrt{u}$, then in floating point computation $fl(3+2\epsilon^2) = 3$, and the computed matrix $fl(A^TA)$ has rank one. The LU factorization is

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

where L and U are well-conditioned. The correct pseudo-inverse is now obtained from

$$A^\dagger = U^{-1}D^{-1}(L^TL)^{-1}L^T = \begin{pmatrix} 1 & -\epsilon \\ 0 & \epsilon \end{pmatrix} \begin{pmatrix} 1/3 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \end{pmatrix}.$$

and there is no cancellation.

As seen in Example 8.2.4 weighted least squares problems of the form

$$\min \left\| \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} x - \begin{pmatrix} \gamma b_1 \\ b_2 \end{pmatrix} \right\|, \tag{8.2.45}$$

where $\gamma \gg 1$, are not suited to a direct application of the method of normal equations. If $p = \operatorname{rank}(A_1)$ steps of Gaussian elimination with pivoting are applied to the resulting factorization can be written

$$\Pi_1 \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} \Pi_2 = LDU, \tag{8.2.46}$$

where Π_1 and Π_2 are permutation matrices, and

$$L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \in \mathbf{R}^{m\times n}, \quad U = \begin{pmatrix} U_{11} & U_{12} \\ & I \end{pmatrix} \in \mathbf{R}^{n\times n}.$$

Here $L_{11} \in \mathbf{R}^{p\times p}$ is unit lower triangular, and $U_{11} \in \mathbf{R}^{p\times p}$ is unit upper triangular. Assuming that A has full rank, D is nonsingular. Then (4.4.1) is equivalent to

$$\min_y \|Ly - \Pi_1 b\|_2, \qquad DU\Pi_2^T x = y.$$

The least squares problem in y is usually well-conditioned, since any ill-conditioning from the weights is usually reflected in D. Therefore, it can be solved by forming the normal equations; see Problem .

Review Questions

2.1 Give a necessary and sufficient condition for x to be a solution to $\min_x \|Ax - b\|_2$, and interpret this geometrically. When is is the least squares solution x unique? When is $r = b - Ax$ unique?

2.2 What are the advantages and drawbacks with the method of normal equations for computing the least squares solution of $Ax = b$? Give a simple example, which shows that loss of information can occur in forming the normal equations.

2.3 Discuss how the accuracy of the method of normal equations can be improved by (a) scaling the columns of A, (b) iterative refinement.

2.4 Show that the more accurate formula in Example 8.2.1 can be interpreted as a special case of the method (8.5.3)–(8.5.4) for partitioned least squares problems.

2.5 (a) Let $A \in \mathbf{R}^{m\times n}$ with $m < n$. Show that A^TA is singular.

(b) Show, using the SVD, that $\operatorname{rank}(A^TA) = \operatorname{rank}(AA^T) = \operatorname{rank}(A)$.

2.6 Define the condition number $\kappa(A)$ of a rectangular matrix A. What terms in the perturbation of a least squares solution depend on κ and κ^2, respectively?

Problems

2.1 In order to estimate the height above sea level for three points, A,B, and C, the difference in altitude was measured between these points and points D,E, and F at sea level. The measurements obtained form a linear system in the heights x_A, x_B, and x_C of A,B, and C,

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ x_B \\ x_C \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 1 \end{pmatrix}.$$

Show that the least squares solution and residual vector are

$$x = \frac{1}{4}(5, 7, 12)^T, \qquad r = \frac{1}{4}(-1, 1, 0, 2, 3, -3)^T.$$

and verify that the residual vector is orthogonal to all columns in A.

2.2 (a) Consider the linear regression problem of fitting $y(t) = \alpha + \beta(t - c)$ by the method of least squares to the data

t	1	3	4	6	7
$f(t)$	−2.1	−0.9	−0.6	0.6	0.9

With the (unsuitable) choice $c = 1,000$ the normal equations

$$\begin{pmatrix} 5 & 4979 \\ 4979 & 4958111 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} -2.1 \\ -2097.3 \end{pmatrix}$$

become very ill-conditioned. Show that if the element 4958111 is rounded to $4958 \cdot 10^3$ then β is perturbed from its correct value 0.5053 to −0.1306!

(b) As shown in Example 8.2.1, a much better choice of base functions is shifting with the mean value of t, i.e., taking $c = 4.2$. However, it is not necessary to shift with the *exact* mean; Show that shifting with 4, the midpoint of the interval $(1, 7)$, leads to a very well-conditioned system of normal equations.

2.3 Denote by x_V the solution to the weighted least squares problem with covariance matrix V. Let x be the solution to the corresponding unweighted problem $(V = I)$. Using the normal equations show that

$$x_V - x = (A^T V^{-1} A)^{-1} A^T (V^{-1} - I)(b - Ax). \tag{8.2.47}$$

Conclude that weighting the rows affects the solution if $b \notin \mathcal{R}(A)$.

2.4 Assume that $\operatorname{rank}(A) = n$, and put $\bar{A} = (A, b) \in \mathbf{R}^{m\times(n+1)}$. Let the corresponding cross product matrix, and its Cholesky factor be

$$\bar{C} = \bar{A}^T \bar{A} = \begin{pmatrix} C & d \\ d^T & b^T b \end{pmatrix}, \qquad \bar{R} = \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}.$$

Show that the solution x and the residual norm ρ to the linear least squares problem $\min_x \|b - Ax\|_2$ is given by

$$Rx = z, \qquad \|b - Ax\|_2 = \rho.$$

2.5 Let $A \in \mathbf{R}^{m\times n}$ and $\operatorname{rank}(A) = n$. Show that the minimum norm solution of the underdetermined system $A^T y = c$ can be computed as follows:

(i) Form the matrix A^TA, and compute its Cholesky factorization $A^TA = R^TR$.

(ii) Solve the two triangular systems $R^Tz = c$, $Rx = z$, and compute $y = Ax$.

2.6 (S. M. Stiegler [350].) In 1793 the French decided to base the new metric system upon a unit, the meter, equal to one 10,000,000th part of the distance from the the north pole to the equator along a meridian arc through Paris. The following famous data obtained in a 1795 survey consist of four measured subsections of an arc from Dunkirk to Barcelona. For each subsection the length of the arc S (in modules), the degrees d of latitude and the latitude L of the midpoint (determined by the astronomical observations) are given.

Segment	Arc length S	latitude d	Midpoint L
Dunkirk to Pantheon	62472.59	2.18910°	49° 56′ 30″
Pantheon to Evaux	76145.74	2.66868°	47° 30′ 46″
Evaux to Carcassone	84424.55	2.96336°	44° 41′ 48″
Carcassone to Barcelona	52749.48	1.85266°	42° 17′ 20″

If the earth is ellipsoidal, then to a good approximation it holds

$$z + y\sin^2(L) = S/d,$$

where z and y are unknown parameters. The meridian quadrant then equals $M = 90(z + y/2)$ and the eccentricity is e is found from $1/e = 3(z/y + 1/2)$. Use least squares to determine z and y and then M and $1/e$.

2.7 Consider the least squares problem $\min_x \|Ax - b\|_2^2$, where A has full column rank. Partition the problem as

$$\min_{x_1,x_2} \left\| (A_1 \; A_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - b \right\|_2^2.$$

By a geometric argument show that the solution can be obtained as follows. First compute x_2 as solution to the problem

$$\min_{x_2} \|P_{A_1}^{\perp}(A_2x_2 - b)\|_2^2,$$

where $P_{A_1}^{\perp} = I - P_{A_1}$ is the orthogonal projector onto $\mathcal{N}(A_1^T)$. Then compute x_2 as solution to the problem

$$\min_{x_1} \|A_1x_1 - (b - A_2x_2)\|_2^2.$$

2.8 Show that if $A, B \in \mathbf{R}^{m\times n}$ and rank $(B) \neq$ rank (A) then it is not possible to bound the difference between $A^\dagger$ and $B^\dagger$ in terms of the difference $B - A$. *Hint*: Use the following example. Let $\epsilon \neq 0$, $\sigma \neq 0$, take

$$A = \begin{pmatrix} \sigma & 0 \\ 0 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} \sigma & \epsilon \\ \epsilon & 0 \end{pmatrix},$$

and show that $\|B - A\|_2 = \epsilon$, $\|B^\dagger - A^\dagger\|_2 > 1/\epsilon$.

2.9 Show that for any matrix A it holds

$$A^{\dagger} = \lim_{\mu \to 0} (A^TA + \mu^2 I)^{-1} A^T = \lim_{\mu \to 0} A^T (AA^T + \mu^2 I)^{-1}. \tag{8.2.48}$$

2.10 (a) Let $A = (a_1, a_2)$, where $a_1^T a_2 = \cos\gamma$, $\|a_1\|_2 = \|a_2\|_2 = 1$. Hence, γ is the angle between the vectors a_1 and a_2. Determine the singular values and right singular vectors v_1, v_2 of A by solving the eigenvalue problem for

$$A^TA = \begin{pmatrix} 1 & \cos\gamma \\ \cos\gamma & 1 \end{pmatrix}.$$

Then determine the left singular vectors u_1, u_2 from (7.1.33).
(b) Show that if $\gamma \ll 1$, then $\sigma_1 \approx \sqrt{2}$ and $\sigma_2 \approx \gamma/\sqrt{2}$ and

$$u_1 \approx (a_1 + a_2)/2, \qquad u_2 \approx (a_1 - a_2)/\gamma.$$

2.11 The least squares problem $\min_x \|Ax - b\|_2$, where

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \epsilon & & \\ & \epsilon & \\ & & \epsilon \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

is of the form (8.2.45). Compute the factorization $A = LDU$ that is obtained after one step of Gaussian elimination and show that L and U are well-conditioned. Compute the solution from

$$L^TLy = L^Tb, \qquad Ux = D^{-1}y.$$

by solving the system of normal equations for y and solving for x by back-substitution.

8.3 Orthogonal Factorizations

8.3.1 Householder QR Factorization

Orthogonality plays a key role in least squares problems; see Theorem 8.1.2. By using methods directly based on orthogonality the squaring of the condition number that results from forming the normal equations can be avoided.

We first show that any matrix $A \in \mathbf{R}^{m\times n}$ $(m \geq n)$ can be factored into the product of a *square* orthogonal matrix $Q \in \mathbf{R}^{m\times m}$ and an upper triangular matrix $R \in \mathbf{R}^{m\times n}$ with positive diagonal elements.

Theorem 8.3.1. *The Full QR Factorization*

Let $A \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(A) = n$. *Then there is a square orthogonal matrix $Q \in \mathbf{R}^{m\times m}$ and an upper triangular matrix R with positive diagonal elements such that*

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}. \tag{8.3.1}$$

Proof. The proof is by induction on n. Let A be partitioned in the form $A = (a_1, A_2)$, $a_1 \in \mathbf{R}^m$, where $\rho = \|a_1\|_2 > 0$. Put $y = a_1/\rho$, and let $U = (y, U_1)$ be an orthogonal matrix. Then since $U_1^T a_1 = 0$ the matrix $U^T A$ must have the form

$$U^T A = \begin{pmatrix} \rho & y^T A_2 \\ 0 & U_1^T A_2 \end{pmatrix} = \begin{pmatrix} \rho & r^T \\ 0 & B \end{pmatrix},$$

where $B \in \mathbf{R}^{(m-1)\times(n-1)}$. For $n = 1$, A_2 is empty and the theorem holds with $Q = U$ and $R = \rho$, a scalar. If $n > 1$ then $\operatorname{rank}(B) = n - 1 > 0$, and by the induction hypothesis there is an orthogonal matrix $\tilde{Q}$ such that $\tilde{Q}^T B = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$. (8.3.1) will hold if we define

$$Q = U \begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q} \end{pmatrix}, \qquad R = \begin{pmatrix} \rho & r^T \\ 0 & \tilde{R} \end{pmatrix}.$$

□

The proof of this theorem gives a way to compute Q and R, provided we can construct an orthogonal matrix $U = (y,\ U_1)$ given its first column. Several ways to perform this construction using elementary orthogonal transformations were given in Sec. .

Note that from the form of the decomposition (8.3.1) it follows that R has the same singular values and right singular vectors as A. A relationship between the Cholesky factorization of $A^T A$ and the QR decomposition of A is given next.

Lemma 8.3.2.

Let $A \in \mathbf{R}^{m\times n}$ have rank n. Then if the R factor in the QR factorization of A has positive diagonal elements it equals the Cholesky factor of $A^T A$.

Proof. If $\operatorname{rank}(A) = n$ then the matrix $A^T A$ is nonsingular. Then its Cholesky factor R_C is uniquely determined, provided that R_C is normalized to have a positive diagonal. From (8.3.33) we have $A^T A = R^T Q_1^T Q_1 R = R^T R$, and hence $R = R_C$. Since $Q_1 = AR^{-1}$ the matrix Q_1 is also uniquely determined. □

The QR factorization can be written

$$A = (\, Q_1 \quad Q_2 \,) \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1 R. \tag{8.3.2}$$

where the square orthogonal matrix $Qin\mathbf{R}^{m\times m}$ has been partitioned as

$$Q = (Q_1, Q_2), \quad Q_1 \in \mathbf{R}^{m\times n}, \quad Q_2 \in \mathbf{R}^{m\times(m-n)}.$$

Here $A = Q_1 R$ is called the **thin QR factorization**. From (8.3.2) it follows that the columns of Q_1 and Q_2 form orthonormal bases for the range space of A and its orthogonal complement,

$$\mathcal{R}(A) = \mathcal{R}(Q_1), \qquad \mathcal{N}(A^T) = \mathcal{R}(Q_2), \tag{8.3.3}$$

and the corresponding orthogonal projections are

$$P_{\mathcal{R}(A)} = Q_1 Q_1^T, \qquad P_{\mathcal{N}(A^T)} = Q_2 Q_2^T. \tag{8.3.4}$$

Note that although the matrix Q_1 in (8.3.2) is uniquely determined, Q_2 can be any orthogonal matrix with range $\mathcal{N}(A^T)$. The matrix Q is *implicitly* defined as a product of Householder or Givens matrices.

The QR factorization of a matrix $A \in \mathbf{R}^{m\times n}$ of rank n can be computed using a sequence of n Householder reflectors. Let $A = (a_1, a_2, \dots, a_n)$, $\sigma_1 = \|a_1\|_2$, and choose $P_1 = I - \beta_1 u_1 u_1^T$, so that

$$P_1 a_1 = P_1 \begin{pmatrix} \alpha_1 \\ \hat{a}_1 \end{pmatrix} = \begin{pmatrix} r_{11} \\ 0 \end{pmatrix}, \qquad r_{11} = -\text{sign}\,(\alpha_1)\sigma_1.$$

By (8.1.56) we achieve this by choosing $\beta_1 = 1 + |\alpha_1|/\sigma_1$,

$$u_1 = \begin{pmatrix} 1 \\ \hat{u}_1 \end{pmatrix}, \qquad \hat{u}_1 = \text{sign}\,(\alpha_1)\hat{a}_1/\rho_1, \qquad \rho_1 = \sigma_1\beta_1.$$

P_1 is then applied to the remaining columns $a_2, \dots, a_n$, giving

$$A^{(2)} = P_1 A = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & \tilde{a}_{22} & \dots & \tilde{a}_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{n2} & \dots & \tilde{a}_{nn} \end{pmatrix}.$$

Here the first column has the desired form and, as indicated by the notation, the first row is the final first row in R. In the next step the $(m-1)\times(n-1)$ block in the lower right corner is transformed. All remaining steps, $k = 2 : n$ are similar to the first. Before the kth step we have computed a matrix of the form

$$A^{(k)} = \begin{array}{c} \\ {\scriptstyle k-1} \\ \\ \end{array} \!\!\begin{array}{c} \begin{array}{cc} {\scriptstyle k-1} & \end{array} \\ \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & \hat{A}^{(k)} \end{pmatrix} \end{array}, \tag{8.3.5}$$

where the first $k-1$ rows of $A^{(k)}$ are rows in the final matrix R, and $R_{11}^{(k)}$ is upper triangular. In step k the matrix $a^{(k)}$ is transformed,

$$A^{(k+1)} = P_k A^{(k)}, \qquad P_k = \begin{pmatrix} I_k & 0 \\ 0 & \tilde{P}_k \end{pmatrix}. \tag{8.3.6}$$

Here $\tilde{P}_k = I - \beta_k u_k u_k^T$ is chosen to zero the elements below the main diagonal in the first column of the submatrix

$$\hat{A}^{(k)} = (a_k^{(k)}, \dots, a_n^{(k)}) \in \mathbf{R}^{(m-k+1)\times(n-k+1)},$$

i.e. $\tilde{P}_k a_k^{(k)} = r_{kk} e_1$. With $\sigma_k = \|a_k^{(k)}\|_2$, using (8.1.55), we get $r_{kk} = -\text{sign}\,\big(a_{kk}^{(k)}\big)\sigma_k$, and

$$\hat{u}_k = \text{sign}\,(\alpha_k^{(k)})\hat{a}_k^{(k)}/\rho_k, \qquad \beta_k = 1 + |a_{kk}|/\sigma_k. \tag{8.3.7}$$

where $\rho_k = \sigma_k \beta_k$. After n steps we have obtained the QR factorization of A, where

$$R = R_{11}^{(n+1)}, \qquad Q = P_1 P_2 \cdots P_n. \tag{8.3.8}$$

Note that the diagonal elements r_{kk} will be positive if $a_k^{(kk)}$ is negative and negative otherwise. Negative diagonal elements may be removed by multiplying the corresponding rows of R and columns of Q by -1.

Algorithm 8.3.
Householder QR Factorization. Given a matrix $A^{(1)} = A \in \mathbf{R}^{m\times n}$ of rank n, the following algorithm computes R and Householder matrices:

$$P_k = \operatorname{diag}(I_{k-1}, \tilde{P}_k), \qquad \tilde{P}_k = I - \beta_k u_k u_k^T, \quad k = 1:n, \tag{8.3.9}$$

so that $Q = P_1 P_2 \cdots P_n$.

$$
\begin{aligned}
&\text{for } k = 1:n\\
&\quad [u_k, \beta_k, r_{kk}] = \text{house}(a_k^{(k)});\\
&\quad \text{for } j = k+1, \dots, n\\
&\qquad \gamma_{jk} = \beta_k u_k^T a_j^{(k)};\\
&\qquad r_{kj} = a_{kj}^{(k)} - \gamma_{jk};\\
&\qquad a_j^{(k+1)} = \hat{a}_j^{(k)} - \gamma_{jk}\hat{u}_k;\\
&\quad \text{end}\\
&\text{end}
\end{aligned}
$$

If $m = n$ the last step can be skipped. The vectors $\hat{u}_k$ can overwrite the elements in the strictly lower trapezoidal part of A. Thus, all information associated with the factors Q and R can be overwritten A. The vector $(\beta_1, \dots, \beta_n)$ of length n can be recomputed from

$$\beta_k = \tfrac{1}{2}(1 + \|\hat{u}_k\|_2^2)^{1/2},$$

and therefore need not be saved. In step k the application of the Householder transformation to the active part of the matrix requires $4(m-k+1)(n-k)$ flops. Hence the total flop count is

$$4\sum_{k=1}^{n-1}(m-k+1)(n-k) = 4\sum_{p=1}^{n-1}((m-n)p + p(p+1)) = 2(mn^2 - n^3/3).$$

If $m = n$ this is $4n^3/3$ flops.

Theorem 8.3.3.
*Let $\bar{R}$ denote the upper triangular matrix R computed by the Householder QR algorithm. Then there exists an exactly orthogonal matrix $\hat{Q} \in \mathbf{R}^{m\times m}$ (*not *the*

matrix corresponding to exact computation throughout) such that

$$A + E = \hat{Q} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \qquad \|e_j\|_2 \leq \bar{\gamma}_n \|a_j\|_2, \quad j = 1 : n. \tag{8.3.10}$$

where $\bar{\gamma}_n$ *is defined in* (7.1.82).

As have been stressed before it is usually not advisable to compute the matrix Q in the QR factorization explicitly, even when it is to be used in later computing matrix-vector products. In case that

$$Q = Q^{(0)} = P_1 P_2 \cdots P_n$$

from the Householder algorithm is explicitly required it can be accumulated using the backward recurrence

$$Q^{(n)} = I_m, \qquad Q^{(k-1)} = P_k Q^{(k)}, \quad k = n : -1 : 1. \tag{8.3.11}$$

which requires $4(mn(m-n) + n^3/3)$ flops. (Note that this is more efficient than the corresponding forward recurrence. By setting

$$Q^{(n)} = \begin{pmatrix} I_n \\ 0 \end{pmatrix}, \quad \text{or} \quad Q^{(n)} = \begin{pmatrix} 0 \\ I_{m-n} \end{pmatrix},$$

For $m = n$ this becomes $4n^3/3$ flops. The matrices Q_1 and Q_2, whose columns span the range space and nullspace of A, respectively, can be similarly computed in $2(mn^2 - n^3/3)$ and $2m^2n - 3mn^2 + n^3$ flops, respectively; see Problem 6 (b).

An algorithm similar to Algorithm 8.3.1, but using Givens rotations, can easily be developed. The greater flexibility of Givens rotations can be taken advantage of when the matrix A is structured or sparse. An important exmaple is the QR factorization of a Hessenberg matrix

$$H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,n-1} & h_{1,n} \\ h_{21} & h_{22} & \cdots & h_{2,n-1} & h_{2,n} \\ & h_{32} & \cdots & \vdots & \vdots \\ & & \ddots & h_{n-1,n-1} & h_{n-1,n} \\ & & & h_{n,n-1} & h_{n,n} \end{pmatrix}.$$

using Givens transformations. Applications of this arise e.g., in linear regression and modified least squares problems; see Sec. 8.4.1.

We illustrate below the first two steps of the algorithm for $n = 5$ in a Wilkinson diagram. In the first step a rotation G_{12} in rows (1,2) is applied to zero out the element h_{21}; in the second step a rotation G_{23} in rows (2,3) is applied to zero out the next subdiagonal element h_{32}, etc.

$$\begin{array}{c} \rightarrow \\ \rightarrow \\ \\ \\ \\ \end{array} \begin{pmatrix} \times & \times & \times & \times & \times \\ \otimes & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{pmatrix} \qquad \begin{array}{c} \\ \rightarrow \\ \rightarrow \\ \\ \\ \end{array} \begin{pmatrix} \times & \times & \times & \times & \times \\ \otimes & \times & \times & \times & \times \\ & \otimes & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{pmatrix}.$$

The arrows points to the rows that took part in the last rotation. Note that no new nonzero elements are introduced by the rotations. After $n-1$ steps all subdiagonal elements have been zeroed out and we have obtained the QR factorization

$$H = QR, \quad Q = G_{12}^T G_{23}^T \cdots G_{n-1,n}^T. \tag{8.3.12}$$

The first step takes $6n$ flops and the total work of this QR factorization is only about $3n$ flops.

It is often advisable to use **column pivoting** in Householder QR factorization. The standard procedure is choose pivot column to maximize the diagonal element r_{kk} in the kth step. Assume that after $k-1$ steps we have computed the partial QR factorization

$$A^{(k)} = (P_{k-1}\cdots P_1)A(\Pi_1\cdots\Pi_{k-1}) = \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & \tilde{A}^{(k)} \end{pmatrix}, \tag{8.3.13}$$

Then the pivot column in the next step is chosen as a column of largest norm in the submatrix

$$\tilde{A}^{(k)} = (\tilde{a}_k^{(k)}, \dots, \tilde{a}_n^{(k)}),$$

If $\tilde{A}^{(k)} = 0$ the algorithm terminates. Otherwise, let

$$s_j^{(k)} = \|\tilde{a}_j^{(k)}\|_2^2, \quad j = k:n. \tag{8.3.14}$$

and interchange columns p and k, where p is the smallest index such that $s_p^{(k)} = \max_{j=k}^n s_j^{(k)}$, This pivoting strategy ensures that the computed triangular factor has the property stated in Theorem 8.3.4. Since the column lengths are invariant under orthogonal transformations the quantities $s_j^{(k)}$ can be updated

$$s_j^{(k+1)} = s_j^{(k)} - r_{jk}^2, \quad j = k+1:n. \tag{8.3.15}$$

We remark that the pivoting rule (8.3.36) is equivalent to maximizing the diagonal element r_{kk}, $k = 1:r$. Therefore, (in exact arithmetic it computes the Cholesky factor that corresponds to using the pivoting (7.3.9) in the Cholesky factorization.

If the column norms in $\tilde{a}^{(k)}$ were recomputed at each stage, then column pivoting would increase the operation count by 50%. Instead the norms of the columns of A can be computed initially, and recursively updated as the factorization proceeds. This reduces the overhead of column pivoting to $0(mn)$ operations. This pivoting strategy can also be implemented in the Cholesky and modified Gram–Schmidt algorithms.

Since column norms are preserved by orthogonal transformations the factor R has the following important property:

Theorem 8.3.4.

Suppose that R is computed by QR factorization with column pivoting. Then the elements in R satisfy the inequalities

$$r_{kk}^2 \geq \sum_{i=k}^{j} r_{ij}^2, \quad j = k+1, \dots, n. \tag{8.3.16}$$

In particular, $|r_{kk}| \geq |r_{kj}|$, $j > k$, *and the diagonal elements form a non-increasing sequence*

$$|r_{11}| \geq |r_{22}| \geq \cdots \geq |r_{nn}|. \tag{8.3.17}$$

For any QR factorization it holds that

$$\sigma_1 = \max_{\|x\|=1} \|Rx\|_2 \geq \|Re_1\|_2 = |r_{11}|,$$

and thus $|r_{11}|$ is a lower bound for the largest singular value σ_1 of A. and singular values $1/\sigma_k(A)$. Similarly,

$$\sigma_n = \min_{\|x\|=1} \|R^T x\|_2 \leq \|R^T e_n\|_2 = |r_{nn}|,$$

which gives an upper bound for σ_n.

For a triangular matrix satisfying (8.3.16) we also have the upper bound

$$\sigma_1(R) = \|R\|_2 \leq \|R\|_F = \Big(\sum_{i\leq j} r_{ij}^2\Big)^{1/2} \leq \sqrt{n} r_{11}.$$

$\sigma_1 \leq n^{1/2} r_{11}$. Using the interlacing property of singular values (Theorem 8.1.15), a similar argument gives the upper bounds

$$\sigma_k(R) \leq (n-k+1)^{1/2} |r_{k,k}|, \quad 1 \leq k \leq n. \tag{8.3.18}$$

If after k steps in the pivoted QR factorization it holds that

$$|r_{k,k}| \leq (n-k+1)^{-1/2}\delta,$$

then $\sigma_k(A) = \sigma_k(R) \leq \delta$, and A has numerical rank at most equal to $k-1$, and we should terminate the algorithm. Unfortunately, the converse is not true, i.e., the rank is not always revealed by a small element $|r_{kk}|$, $k \leq n$. Let R be an upper triangular matrix whose elements satisfy (8.3.16). The best known *lower* bound for the smallest singular value is

$$\sigma_n \geq 3|r_{nn}|/\sqrt{4^n + 6n - 1} \geq 2^{1-n}|r_{nn}|. \tag{8.3.19}$$

(For a proof of this see Lawson and Hanson [257, Ch. 6].)

The lower bound in (8.3.19) can almost be attained as shown in the example below due to Kahan. Then the pivoted QR factorization may not reveal the rank of A.

Example 8.3.1. Consider the upper triangular matrix

$$R_n = \operatorname{diag}(1, s, s^2, \ldots, s^{n-1}) \begin{pmatrix} 1 & -c & -c & \ldots & -c \\ & 1 & -c & \ldots & -c \\ & & 1 & & \vdots \\ & & & \ddots & -c \\ & & & & 1 \end{pmatrix}, \qquad s^2 + c^2 = 1.$$

It can be verified that the elements in R_n satisfies the inequalities in (8.3.19), and that R_n is invariant under QR factorization with column pivoting. For $n = 100$, $c = 0.2$ the last diagonal element of R is $r_{nn} = s^{n-1} = 0.820$. This can be compared with the smallest singular value which is $\sigma_n = 0.368 \cdot 10^{-8}$. If the columns are reordered as $(n, 1, 2, \dots, n-1)$ and the rank is revealed from the pivoted QR factorization!

The above example has inspired research into alternative column permutation strategies. The following theorem, which we state without proof, shows that a column permutation Π can always be found so that the numerical rank of A is revealed by the QR factorization of $A\Pi$.

Theorem 8.3.5. (H. P. Hong and C. T. Pan [1992].)

Let $A \in \mathbf{R}^{m\times n}$, $(m \geq n)$, and r be a given integer $0 < r < n$. Then there exists a permutation matrix Π_r, such that the QR factorization has the form

$$Q^T A\Pi_r = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \tag{8.3.20}$$

with $R_{11} \in \mathbf{R}^{r\times r}$ upper triangular, and

$$\sigma_{\min}(R_{11}) \geq \frac{1}{c}\sigma_r(A), \qquad \sigma_{\max}(R_{22}) \leq c\sigma_{r+1}(A), \tag{8.3.21}$$

where $c = \sqrt{r(n-r) + \min(r, n-r)})$.

Note that the bounds in this theorem are much better than those in (8.3.19).

From the interlacing properties of singular values (Theorem 8.1.15) it follows by induction that for any factorization of the form (8.3.20) we have the inequalities

$$\sigma_{\min}(R_{11}) \leq \sigma_r(A), \qquad \sigma_{\max}(R_{22}) \geq \sigma_{r+1}(A). \tag{8.3.22}$$

Hence, to achieve (8.3.21) we want to choose the permutation Π to maximize $\sigma_{\min}(R_{11})$ and simultaneously minimize $\sigma_{\max}(R_{22})$. These two problems are in a certain sense dual; cf. Problem 2.

Assume now that A has a well defined numerical rank $r < n$, i.e.,

$$\sigma_1 \geq \ldots \geq \sigma_r \gg \delta \geq \sigma_{r+1} \geq \ldots \geq \sigma_n.$$

Then the above theorem says that if the ratio σ_k/σ_{k+1} is sufficiently large then there is a permutation of the columns of A such that the rank of A is revealed by the QR factorization. Unfortunately, to find such a permutation may be a hard problem. The naive solution, to try all possible permutations, is not feasible since the cost prohibitive—it is exponential in the dimension n.

8.3.2 Least Squares Problems by QR Factorization

We now show how to use the QR factorization to solve the linear least squares problem (8.1.1).

Theorem 8.3.6.

Let the QR factorization of $A \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(A) = n \le m$ *be given by* (8.3.1). *Then the unique solution x to $\min_x ||Ax - b||_2$ and for the corresponding residual vector r are given by*

$$x = R^{-1}c_1, \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = Q^T b, \quad r = Q\begin{pmatrix} 0 \\ c_2 \end{pmatrix}, \tag{8.3.23}$$

and hence $\|r\|_2 = \|c_2\|_2$.

Proof. Since Q is orthogonal we have

$$\|Ax - b\|_2^2 = \left\|Q^T(Ax - b)\right\|_2^2 = \left\|\begin{pmatrix} Rx \\ 0 \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}\right\|_2^2 = \|Rx - c_1\|_2^2 + \|c_2\|_2^2.$$

Obviously the minimum residual norm $\|c_2\|_2$ is obtained by taking $x = R^{-1}c_1$. With c defined by (8.3.23) and using the orthogonality of Q we have

$$b = QQ^T b = Q_1c_1 + Q_2c_2 = Ax + r$$

which shows the formula for r. $\square$

By Theorem 8.3.6, when R and the Householder reflections $P_1, P_2, \dots, P_n$ have been computed by Algorithm 8.3.1 the least squares solution x and residual r can be computed as follows:

$$\begin{aligned} c &= \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = P_n \cdots P_2 P_1 b, \\ Rx &= c_1, \qquad r = P_1 \cdots P_{n-1}P_n \begin{pmatrix} 0 \\ c_2 \end{pmatrix}, \end{aligned} \tag{8.3.24}$$

and $\|r\|_2 = \|c_2\|_2$.

When the matrix A has full row rank, $\operatorname{rank}(A) = m \le n$, the QR factorization of A^T (which is equivalent to the LQ factorization of A) can be used to compute the minimum norm solution (8.1.2).

Theorem 8.3.7.

Let $A \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(A) = m$, *have the LQ factorization*

$$A = \begin{pmatrix} L & 0 \end{pmatrix}\begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix}, \quad Q_1 \in \mathbf{R}^{n\times m},$$

Then the general solution to the underdetermined system $Ax = b$ is

$$x = Q_1y_1 + Q_2y_2, \qquad y_1 = L^{-1}b \tag{8.3.25}$$

where y_2 is arbitrary. The minimum norm solution $x = Q_1L^{-1}b$ is obtained by taking $y_2 = 0$.

Proof. Since $A = (\,L \quad 0\,)\,Q^T$ the system $Ax = b$ can be written

$$(\,L \quad 0\,)\,y = b, \quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = Q^T x.$$

L is nonsingular, and thus y_1 is determined by $Ly_1 = b$. The vector y_2 can be chosen arbitrarily. Further, since $\|x\|_2 = \|Qy\|_2 = \|y\|_2$ the minimum norm solution is obtained by taking $y_2 = 0$. $\square$

The operation count $mn^2 - n^3/3$ for the QR method can be compared with that for the method of normal equations, which requires $\frac{1}{2}(mn^2 + n^3/3)$ multiplications. Hence, for $m = n$ both methods require the same work but for $m \gg n$ the QR method is twice as expensive. To compute c by (8.3.24) requires $(2mn - n^2)$ multiplications, and thus to compute the solution for each new right hand side takes only $(2mn - n^2/2)$ multiplications.

The Householder QR algorithm, and the resulting method for solving the least squares problem are backwards stable, both for x and r, and the following result holds.

Theorem 8.3.8.

*Let $\bar{R}$ denote the computed R. Then there exists an exactly orthogonal matrix $\tilde{Q} \in \mathbf{R}^{m\times m}$ (*not* the matrix corresponding to exact computation throughout) such that*

$$A + \Delta A = \tilde{Q} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \quad \|\Delta A\|_F \le c\gamma_{mn}\|A\|_F, \tag{8.3.26}$$

where c is a small constant. Further, the computed solution $\bar{x}$ is the exact solution of a slightly perturbed least squares problem

$$\min_x \|(A + \delta A)x - (b + \delta b)\|_2,$$

where the perturbation can be bounded in norm by

$$\|\delta A\|_F \le c\gamma_{mn}\|A\|_F, \qquad \|\delta b\|_2 \le c\gamma_{mn}\|b\|_2, \tag{8.3.27}$$

Proof. See Higham [211, Theorem 19.5]. $\square$

The column pivoting strategy suggested earlier may not be appropriate, when we are given one vector b, which is to be approximated by a linear combination of as few columns of the matrix A as possible. This occurs, e.g., in regression analysis, where each column in A corresponds to one factor. Even when the given b is parallel to one column in A it could happen that we had to complete the full QR factorization of A before this fact was discovered. Instead we would like at each stage to select the column that maximally reduces the sum of squares of the residuals. An elegant way to achieve this is to apply the QR factorization with standard pivoting to the augmented matrix $(\,b \quad A\,)$ giving

$$Q^T(\,b \quad A\,) = \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} \beta_1 e_1 & H \\ 0 & 0 \end{pmatrix}. \tag{8.3.28}$$

Here $\beta = \|b\|_2$ and H is an $(n+1) \times n$ Hessenberg matrix. The least squares problem now reads $\min_x \|\beta e_1 - Hx\|_2$. This can be solved by computing the QR factorization of the the matrix H, whic as shown can be done in $3n^2$ flops. We return to this approach later in Sec. 8.4.6.

A method combining LU factorization and orthogonalization can be developed by solving the least squares problem in (8.2.44) by an orthogonal reduction of L to lower triangular form. The solution is then obtained by solving $\tilde{L}y = c_1$ by forward substitution, where

$$\begin{pmatrix} L_1 \\ L_2 \end{pmatrix} = Q \begin{pmatrix} \tilde{L} \\ 0 \end{pmatrix}, \qquad Q^T \Pi_1 b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}.$$

In the method of **Cline method** [70, 1973] a sequence of Householder transformations are used to perform this reduction. In the kth step, $k = n : (-1) : 1$, we pre-multiply by a Householder transformation P_k, which is chosen to transform only the rows $k : m$ and zero elements in column k below row n. Below we show the reduction to lower triangular form of L for $m = 6$, $n = 4$, after two steps $k = n, n-1$:

$$\begin{pmatrix} \times & & & \\ \times & \times & & \\ \times & \times & \times & \\ \times & \times & \times & \times \\ \times & \times & \otimes & \otimes \\ \times & \times & \otimes & \otimes \end{pmatrix}.$$

The next step will use rows 2,5,6 to zero out the (5,2) and (6,2). These transformations require $4n^2(m-n)$ flops. Note that the ordering of the transformations is such that no nonzero elements are introduced in L_1.

For slightly overdetermined least squares problems, the elimination method combined with Householder transformations is very efficient. The total number of flops required for computing the least squares solution x by Cline's method is about $2n^2(3m - \frac{7}{3}n)$ flops. Since the method of normal equations using the Cholesky factorization on A^TA requires $2n^2(\frac{1}{2}m + \frac{1}{6}n)$ flops, Cline's method uses fewer operations if $m \le \frac{4}{3}n$.

A version solving (8.2.44) with the MGS method has been analyzed by Plemmons [313, 1974]. If the lower triangular structure of L is taken advantage of then this method requires $4n^2(\frac{3}{2}m - \frac{5}{6}n)$ flops, which is slightly more than Cline's variant. Similar methods for the underdetermined case $(m < n)$ based on the LU decomposition of A are described by Cline and Plemmons [72, 1976].

8.3.3 Gram–Schmidt QR Factorization

In Householder QR factorization the matrix A is premultiplied by a sequence of elementary orthogonal transformations to obtain the R factor. In Gram–Schmidt orthogonalization elementary column operations are instead used to transform the matrix A into an orthogonal matrix Q.[25]

We start by considering the case of orthogonalizing *two* linearly independent vectors a_1 and a_2 in $\mathbf{R}^n$. More precisely, we want to compute two vectors q_1 and q_2 so that

$$\|q_1\|_2 = \|q_2\|_2 = 1, \quad q_2^T q_1 = 0.$$

and span $[a_1, a_2]$ = span $[q_1, q_2]$. We first normalize the vector a_1 and set $q_1 = a_1/r_{11}$, where $r_{11} = \|a_1\|_2$ and then compute

$$\hat{q}_2 = (I - q_1 q_1^T) a_2 = a_2 - r_{12} q_1, \quad r_{12} = q_1^T a_2, \tag{8.3.29}$$

which is equivalent to projecting a_2 onto the orthogonal complement of q_1. It is easily verfied that $q_1^T \hat{q}_2 = q_1^T a_2 - r_{12} q_1^T q_1 = 0$, i.e., $\hat{q}_2 \neq 0$ is orthogonal to q_1. Further, $\hat{q}_2 \neq 0$ since otherwise a_2 would be parallel to a_1, contrary to our assumption. It only remains to normalize $\hat{q}_2$, and set

$$q_2 = \hat{q}_2/r_{22}, \quad r_{22} = \|\hat{q}_2\|_2.$$

Since q_1 and q_2 both are linear combinations of a_1 and a_2, they span the same subspace of $\mathbf{R}^n$. Further, we have the relation

$$(\, a_1 \quad a_2 \,) = (\, q_1 \quad q_2 \,) \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}.$$

The above algorithm can be extended to orthogonalizing any sequence of linearly independent vectors $a_1, a_2, \ldots, a_n$ in $\mathbf{R}^m$ $(m \geq n)$. Elementary orthogonal projections are used to compute orthonormal vectors $q_1, q_2, \ldots, q_n$. such that

$$\text{span}\,[a_1, \ldots, a_k] = \text{span}\,[q_1, \ldots, q_k], \quad k = 1 : n. \tag{8.3.30}$$

Algorithm 8.4. *Classical Gram–Schmidt.*

Set $r_{11} = \|a_1\|_2$, $q_1 = a_1/r_{11}$. For $k = 2 : n$, orthogonalize a_k against $q_1, \ldots, q_{k-1}$:

$$\hat{q}_k = a_k - \sum_{i=1}^{k-1} r_{ik} q_i, \quad r_{ik} = q_i^T a_k, \quad i = 1 : k-1; \tag{8.3.31}$$

and normalize

$$r_{kk} = \|\hat{q}_k\|_2, \qquad q_k = \hat{q}_k / r_{kk}. \tag{8.3.32}$$

Note that $\hat{q}_k \neq 0$, since otherwise a_k is a linear combination of the vectors $a_1, \ldots, a_{k-1}$, which contradicts the assumption. The algorithm requires approximately $2mn^2$ flop. This is $2n^3/3$ flops more than Householder QR factorization, provided the matrix Q is kept in product form.

In matrix terms Gram–Schmidt orthogonalization computes the thin QR Factorization.

[25]The difference between the Householder and Gram–Schmidt QR algorithms has been aptly summarized by Trefethen, who calls Gram–Schmidt triangular orthogonalization as opposed to Householder which is orthogonal triangularization.

Theorem 8.3.9.

Let the matrix $A = (a_1, a_2, \dots, a_n) \in \mathbf{R}^{m\times n}$ have linearly independent columns. Then the Gram–Schmidt algorithm computes $Q_1 \in \mathbf{R}^{m\times n}$ with orthonormal columns and an upper triangular $R \in \mathbf{R}^{n\times n}$ with positive diagonal elements, such that

$$A = (a_1, a_2, \dots, a_n) = (q_1, q_2, \dots, q_n) \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{pmatrix} \equiv Q_1 R. \tag{8.3.33}$$

Proof. Combining (8.3.31) and (8.3.32) we obtain

$$a_k = r_{kk} q_k + \sum_{i=1}^{k-1} r_{ik} q_i = \sum_{i=1}^{k} r_{ik} q_i, \quad k = 1 : n,$$

which is equivalent with (8.3.33). Since the vectors q_k are mutually orthogonal by construction the theorem follows. ☐

For the *numerical* GS factorization of a matrix A a small reordering of the above algorithm gives the **modified Gram–Schmidt** method (MGS). Although mathematically equivalent to the classical algorithm MGS has greatly superior numerical properties, and is therefore usually to be preferred.

The modified Gram–Schmidt (MGS) algorithm employs a sequence of elementary orthogonal projections. At the beginning of step k, we have computed

$$(q_1, \dots, q_{k-1}, a_k^{(k)}, \dots, a_n^{(k)}),$$

where we have put $a_j = a_j^{(1)}$, $j = 1 : n$. Here $a_k^{(k)}, \dots, a_n^{(k)}$ have already been made orthogonal to $q_1, \dots, q_{k-1}$, which are final columns in Q_1. In the kth step q_k is obtained by normalizing the vector $a_k^{(k)}$,

$$\tilde{q}_k = a_k^{(k)}, \qquad r_{kk} = \|\tilde{q}_k\|_2, \qquad q_k = \tilde{q}_k / r_{kk}, \tag{8.3.34}$$

and then $a_{k+1}^{(k)}, \dots, a_n^{(k)}$ are orthogonalized against q_k

$$a_j^{(k+1)} = (I_m - q_k q_k^T) a_j^{(k)} = a_j^{(k)} - r_{kj} q_k, \quad r_{kj} = q_k^T a_j^{(k)}, \quad j = k+1 : n. \tag{8.3.35}$$

After n steps we have obtained the factorization (8.3.33). Note that for $n = 2$ MGS and CGS are identical.

Algorithm 8.5. *Modified Gram–Schmidt.*

Given $A \in R^{m\times n}$ with $\operatorname{rank}(A) = n$ the following algorithm computes the factorization $A = Q_1 R$:

for $k = 1 : n$
  $\hat{q}_k = a_k^{(k)}; \quad r_{kk} = \|\hat{q}_k\|_2;$
  $q_k = \hat{q}_k / r_{kk};$
  for $j = k+1 : n$
    $r_{kj} = q_k^T a_j^{(k)};$
    $a_j^{(k+1)} = a_j^{(k)} - r_{kj} q_k;$
  end
end

The operations in Algorithm 8.3.3 can be sequenced so that the elements in R are computed in a column-wise fashion. However, the row-wise version given above is more suitable if column pivoting is to be performed; see below. The unnormalized vector $\tilde{q}_k$ is just the orthogonal projection of a_k onto the complement of $\text{span}[a_1, a_2, \ldots, a_{k-1}] = \text{span}[q_1, q_2, \ldots, q_{k-1}]$.

In CGS the orthogonalization of a_k can be written

$$\hat{q}_k = (I - Q_{k-1} Q_{k-1}^T) a_k, \quad Q_{k-1} = (q_1, \ldots, q_{k-1}).$$

In MGS *the projections $r_{ik} q_i$ are subtracted from a_k as soon as they are computed*, which corresponds to computing

$$\hat{q}_k = (I - q_{k-1} q_{k-1}^T) \cdots (I - q_1 q_1^T) a_k.$$

For $k > 2$ these two expressions are identical only if the $q_1, \ldots, q_{k-1}$ are accurately orthogonal.

If the columns $a_1, \ldots, a_k - 1$ are linearly independent, then after $k-1$ steps in the Gram–Schmidt orthogonalization we have computed orthogonal vectors q_j such that

$$a_j = r_{jj} q_j + \cdots r_{1j} q_1, \quad r_{jj} \neq 0, \quad j = 1 : k-1.$$

It follows that q_{k-1} is a linear combination of $a_1, \ldots, a_{k-1}$. Assume now that a_k is a linear combination of $a_1, \ldots, a_{k-1}$. Then $a_k^{(k)} = 0$ and the orthogonalization process breaks down. However, if $\text{rank}\,(A) > k$ there must be a vector a_j, $j = k : n$, which is linearly independent of $a_1, \ldots, a_{k-1}$ and for which $a_j^{(k)} \neq 0$. We can then interchange columns k and j and proceed until all remaining columns are linearly dependent on the computed q vectors.

This suggest that we augment the MGS method with **column pivoting**, Let

$$s_j^{(k)} = \|a_j^{(k)}\|_2^2, \quad j = k : n. \tag{8.3.36}$$

Then in step k we choose a column p which maximizes $s_j(k)$ for $j = k : n$ and interchange columns k and p. After the kth step we can update

$$s_j^{(k+1)} = s_j^{(k)} - r_{kj}^2, \quad j = k+1 : n.$$

With column pivoting MGS can be used also when the matrix A has linearly dependent columns. If rank $(A) = r$ it will compute a factorization of the form

$$A\Pi = QR, \quad Q \in \mathbf{R}^{m\times r}, \quad R = (\, R_{11} \quad R_{12}\,) \in \mathbf{R}^{r\times n}, \tag{8.3.37}$$

where Π is a permutation matrix, $Q^TQ = I$, and R_{11} is upper triangular and nonsingular. Indeed, MGS with column pivoting is a good algorithm for determining the rank of a given matrix A.

8.3.4 Loss of Orthogonality in GS and MGS

Loss of orthogonality will occur in orthogonalization whenever cancellation takes place in subtracting the orthogonal projection on q_i from $a_k^{(i)}$, that is when

$$a_j^{(k+1)} = (I - q_kq_k^T)a_j^{(k)}, \qquad \|a_k^{(i+1)}\|_2 \ll \alpha\|a_k^{(i)}\|_2. \tag{8.3.38}$$

We use the standard model for floating point computation, and the basic results in Sec. 2.3.2 to analyze the rounding errors. For the *computed* scalar product $\bar r_{12} = fl(q_1^Ta_2)$ we get

$$|\bar r_{12} - r_{12}| < \gamma_m\|a_2\|_2, \quad \gamma_m = \frac{mu}{1 - mu/2},$$

where u is the unit roundoff. Using $|r_{12}| \le \|a_2\|_2$ we obtain for $\bar q_2 = fl(a_2 - fl(\bar r_{12}q_1))$

$$\|\bar q_2 - \hat q_2\|_2 < \gamma_{m+2}\|a_2\|_2.$$

Since $q_1^T\hat q_2 = 0$, it follows that $|q_1^T\bar q_2| < \gamma_{m+2}\|a_2\|_2$ and the loss of orthogonality

$$\frac{|q_1^T\bar q_2|}{\|\bar q_2\|_2} \approx \frac{|q_1^T\bar q_2|}{\|\hat q_2\|_2} < \gamma_{m+2}\frac{\|a_2\|_2}{\|\tilde q_2\|_2} = \frac{\gamma_{m+2}}{\sin\phi(q_1, a_2)}, \tag{8.3.39}$$

is proportional to $\phi(q_1, a_2)$, the angle between q_1 and a_2.

Example 8.3.2. As an illustration consider the matrix

$$A = (a_1, a_2) = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix}.$$

Using the Gram–Schmidt algorithm and IEEE double precision we get

$$q_1 = \begin{pmatrix} 0.98640009002732 \\ 0.16436198585466 \end{pmatrix},$$

$r_{12} = q_1^Ta_2 = 0.87672336001729,$

$$\hat q_2 = a_2 - r_{12}q_1 = \begin{pmatrix} -0.12501091273265 \\ 0.75023914025696 \end{pmatrix} 10^{-8},$$

$$q_2 = \begin{pmatrix} -0.16436196071471 \\ 0.98640009421635 \end{pmatrix},$$

and

$$R = \begin{pmatrix} 1.31478090189963 & 0.87672336001729 \\ 0 & 0.00000000760583 \end{pmatrix}.$$

Severe cancellation has taken place when computing $\hat{q}_2$, which leads to a serious loss of orthogonality between q_1 and q_2:

$$q_1^T q_2 = 2.5486557 \cdot 10^{-8},$$

which should be compared with the unit roundoff $1.11 \cdot 10^{-16}$. We note that the loss of orthogonality is roughly equal to a factor 10^{-8}.

Due to round-off there will be a gradual (sometimes catastrophic) loss of orthogonality in Gram–Schmidt orthogonalization. Suprisingly, in this respect CGS and MGS behave very differently for $n > 2$. (Note that for $n = 2$ MGS and CGS are the same.) For MGS the loss of orthogonality occurs in a predictable manner and can be bounded in terms of the condition number $\kappa(A)$. It has been shown that if $c_2\kappa u < 1$, then

$$\|I - \bar{Q}_1^T \bar{Q}_1\|_2 \le \frac{c_1}{1 - c_2\kappa u}\kappa u.$$

where c_1 and c_2 denote constants depending on m, n, and the details of the arithmetic. In contrast, the computed vectors q_k from CGS may depart from orthogonality to an almost arbitrary extent. The more gradual loss of orthogonality in the computed vectors q_i for MGS is illustrated in the example below; see also Problem 1.

Example 8.3.3. A matrix $A \in \mathbf{R}^{50\times 10}$ was generated by computing

$$A = U\,DV^T, \quad D = \mathrm{diag}\,(1, 10^{-1}, \ldots, 10^{-9})$$

with U and V orthonormal matrices. Hence, A has singular values $\sigma_i = 10^{-i+1}$, $i = 1 : 10$, and $\kappa(A) = 10^9$. Table 8.3.1 shows the condition number of $A_k = (a_1, \ldots, a_k)$ and the loss of orthogonality in CGS and MGS after k steps as measured by $\|I_k - Q_k^T Q_k\|_2$.

For MGS the loss of orthogonality is more gradual and proportional to $\kappa(A_k)$, whereas for CGS the loss of orthogonality is roughly proportional to $\kappa^2(A_k)$,

It can be shown that for MGS the computed $\bar{Q}_1$ and $\bar{R}$ satisfy

$$A + E = \bar{Q}_1\bar{R}, \quad \|E\|_2 \le c_0 u\|A\|_2. \tag{8.3.40}$$

that is $\bar{Q}_1\bar{R}$ accurately represents A.

Theorem 8.3.10.

Let $\bar{Q}$ and $\bar{R}$ denote the factors computed by the MGS algorithm. Then it holds that

$$A + E = \hat{Q}\bar{R}, \qquad \|e_j\|_2 \le \bar{\gamma}_n\|a_j\|_2, \quad j = 1 : n, \tag{8.3.41}$$

and

$$\|I - \bar{Q}_1^T \bar{Q}_1\|_F \le \bar{\gamma}_n\kappa_2(A). \tag{8.3.42}$$

Table 8.3.1. *Loss of orthogonality and CGS and MGS.*

k	$\kappa(A_k)$	$\|I_k - Q_C^T Q_C\|_2$	$\|I_k - Q_M^T Q_M\|_2$
1	1.000e+00	1.110e-16	1.110e-16
2	1.335e+01	2.880e-16	2.880e-16
3	1.676e+02	7.295e-15	8.108e-15
4	1.126e+03	2.835e-13	4.411e-14
5	4.853e+05	1.973e-09	2.911e-11
6	5.070e+05	5.951e-08	3.087e-11
7	1.713e+06	2.002e-07	1.084e-10
8	1.158e+07	1.682e-04	6.367e-10
9	1.013e+08	3.330e-02	8.779e-09
10	1.000e+09	5.446e-01	4.563e-08

where $\bar{\gamma}_n$ *is defined in* (7.1.82).

In some applications it is important that the computed $\bar{Q}_1$ is accurately orthogonal.This can be achieved by **reorthogonalizing** the computed vectors in the Gram–Schmidt algorithm, This is to be carried out whenever (8.3.38) is satisfied for some suitable parameter $\alpha < 1$ typically chosen in the range $[0.1, 1/\sqrt{2}]$. For example, reorthogonalizing the computed vector $a_2^{(2)}$ in Example 8.3.2 against q_1 gives

$$q_1^T q_2 = 2.5486557 \cdot 10^{-8}, \quad \tilde{q}_2 = \begin{pmatrix} -0.16436198585466 \\ 0.98640009002732 \end{pmatrix}.$$

The vector $\tilde{q}_2$ is exactly orthogonal to q_1.

For the case $n = 2$ it has been shown that *one step of reorthogonalization will always suffice*. (This is made more precise in Parlett [305, Sec. 6.9].) If reorthogonalization is carried whenever

$$\|\bar{q}_2\|_2 < \alpha \|a_2\|_2. \tag{8.3.43}$$

Then the following algorithm computes a vector $\tilde{q}_2$, which satisfies

$$\|\tilde{q}_2 - q_2\|_2 \le cu(1+\alpha)\|a_2\|_2, \qquad \|a_1^T \tilde{q}_2\| \le cu\alpha^{-1}\|\bar{q}_2\|_2 \|a_1\|_2, \tag{8.3.44}$$

where q_2 is the exact complement of a_2 orthogonal to a_1. The first inequality implies that $\bar{q}_2$ is close to a linear combination of a_1 and a_2. The second says that $\bar{q}_2$ is nearly orthogonal to a_1 unless α is small.

When α is large, say $\alpha \ge 1/\sqrt{2}$, then the bounds in (8.3.44) are very good but reorthogonalization will occur more frequently. If α is small, reorthogonalization will be rarer, but the bound on orthogonality less good. For larger n there seems to be a good case for recommending the stringent value $\alpha = 1/\sqrt{2}$ or *always* perform one step of reorthogonalization ($\alpha = 1$).

Now consider the case $n > 2$. Assume we are given a matrix $Q_1 = (q_1, \dots, q_{k-1})$ with $\|q_1\|_2 = \dots = \|q_{k-1}\|_2 = 1$. Adding the new vector a_k, we want to compute a vector $\hat{q}_k$ such that

$$\hat{q}_k \in \operatorname{span}(Q_1, a_k) \perp \operatorname{span}(Q_1).$$

The solution equals $\hat{q}_k = a_k - Q_1 r_k$, where r_k solves the least squares problem

$$\min_{r_k} \|a_k - Q_1 r_k\|_2.$$

We first assume that Q_1 is accurately orthogonal. Then it can be rigorously proved that it suffices to run MGS twice on the matrix (Q_1, a_k). This generalizes the result by Kahan–Parlett to $n > 2$.

To solve the problem, when the columns of Q_1 are not accurately orthogonal, we can use **iterated** Gram–Schmidt methods. In the iterated CGS algorithm we put $\hat{q}_k^{(0)} := a_k$, $r_k^{(0)} := 0$, and for $p = 0, 1, \dots$ compute

$$s_k^{(p)} := Q_1^T \hat{q}_k^{(p)}, \quad \hat{q}_k^{(p+1)} := \hat{q}_k^{(p)} - Q_1 s_k^{(p)}, \quad r_k^{(p+1)} := r_k^{(p)} + s_k^{(p)}.$$

The first step of this algorithm is the usual CGS algorithm, and each step is a reorthogonalization. The iterated MGS algorithm is similar, except that each projection is subtracted as soon as it computed: As in the Kahan–Parlett algorithm, the iterations can be stopped when $\|\hat{q}_k^{(p+1)}\|_2 > \alpha \|\hat{q}_k^p\|_2$.

The iterated Gram–Schmidt algorithm can be used recursively, adding one column a_k at a time, to compute the factorization $A = Q_1 R$. If A has full numerical column rank, then with $\alpha = 1/\sqrt{2}$ both iterated CGS and MGS computes a factor Q_1, which is orthogonal to almost full working precision, *using at most one reorthogonalization*. Hence, in this case iterated CGS is *not* inferior to the iterated MGS.

8.3.5 Least Squares Problems by Gram–Schmidt

To solve a least squares problems the MGS algorithm is applied to the augmented matrix $(A\, b)$. If we skip the normalization of the $(n+1)$st column we obtain a factorization

$$\begin{pmatrix} A & b \end{pmatrix} = \begin{pmatrix} Q_1 & r \end{pmatrix} \begin{pmatrix} R & z \\ 0 & 1 \end{pmatrix}, \tag{8.3.45}$$

and hence $r = b - Q_1 z$. Since the product of the computed factors accurately reproduce the matrix $(A,\, b)$, we have

$$\|Ax - b\|_2 = \left\| \begin{pmatrix} A & b \end{pmatrix} \begin{pmatrix} x \\ -1 \end{pmatrix} \right\|_2 = \|Q_1(Rx - z) - r\|_2.$$

If $Q_1^T r = 0$, then the minimum of the last expression occurs when $Rx - z = 0$. Note that it is not necessary to assume that Q_1 is accurately orthogonal for this conclusion to hold.

This heuristic argument leads to the following algorithm for solving the linear least squares problem by MGS. It can be proved to be backward stable for computing the solution x:

Algorithm 8.6. *Linear Least Squares Solution by MGS.*

Carry out MGS on $A \in R^{m\times n}$, $\mathrm{rank}(A) = n$, to give $Q_1 = (q_1, \ldots, q_n)$ and R, and put $b^{(1)} = b$. Compute the vector $z = (z_1, \ldots, z_n)^T$ by

$$\begin{aligned}
&\text{for } k = 1 : n\\
&\quad z_k = q_k^T b^{(k)}; \quad b^{(k+1)} = b^{(k)} - z_k q_k;\\
&\text{end}\\
&r = b^{(n+1)};\\
&\text{solve } Rx = z;
\end{aligned}$$

Note that the right hand side b is treated as if it were an extra $(n+1)$st column of A. An error that can still found in some textbooks, is to instead compute $c = Q_1^T b$ and solve $Rx = c$. *This will destroy the good accuracy achieved by MGS!*

The residual norm $\|r\|_2$ is accurately obtained from the computed residual However, because of cancellation r will not be accurately orthogonal to $\mathcal{R}(A)$. If this is required, then the computed residual r should be reorthogonalized with respect to $Q_1 = (q_1, q_2, \ldots, q_n)$. This reorthogonalization should be performed in *reverse order* $q_n, q_{n-1}, \ldots, q_1$ as shown in the algorithm below.

Algorithm 8.7. *Orthogonal projection by MGS.*

To make Algorithm 8.3.5 backward stable for r it suffices to add a loop where the vector $b^{(n+1)}$ is orthogonalized against $q_n, q_{n-1}, \ldots, q_1$ (*note the order*):

$$\begin{aligned}
&\text{for } k = n, n-1, \ldots, 1\\
&\quad z_k = q_k^T b^{(k+1)}; \quad b^{(k)} = b^{(k+1)} - z_k q_k;\\
&\text{end}\\
&r = b^{(1)};
\end{aligned}$$

It can be proved that this step "magically" compensates for the lack of orthogonality of Q_1 and the $\bar{r}$ computed by Algorithm 8.3.5 satisfies (8.3.46). An explanation of this subtle point is given at the end of Sec. 8.3.1. , and a result similar to that in Theorem 8.3.3 holds for the computed $\hat{R}$ and $\hat{x}$.

This algorithm is backwards stable for the computed residual $\bar{r}$, i.e. there holds a relation

$$(A + E)^T \bar{r} = 0, \quad \|E\|_2 \le cu\|A\|_2, \tag{8.3.46}$$

for some constant c. This implies that $A^T\bar{r} = -E^T\bar{r}$, and

$$\|A^T\bar{r}\|_2 \le cu\|\bar{r}\|_2\|A\|_2. \tag{8.3.47}$$

Note that this is much better than if we compute the residual in the straightforward way as $r = b - Ax$, or in floating point arithmetic

$$\bar{r} = fl(b - fl(Ax)) = fl\left(\begin{pmatrix} b & A \end{pmatrix} \begin{pmatrix} 1 \\ -x \end{pmatrix} \right)$$

even when x is the *exact* least squares solution. We obtain using (2.3.13) and $A^T r = 0$

$$|A^T \bar{r}| < \gamma_{n+1} |A^T| (|b| + |A||x|).$$

From this we get the norm-wise bound

$$\|A^T \bar{r}\|_2 \leq n^{1/2} \gamma_{n+1} \|A\|_2 (\|b\|_2 + n^{1/2} \|A\|_2 \|x\|_2),$$

which is a much weaker than (8.3.47) when, as is often the case, $\|\bar{r}\|_2 \ll \|b\|_2$!

A similar idea is used to construct a backward stable algorithm for the minimum norm problem

$$\min \|y\|_2, \qquad A^T y = c.$$

Algorithm 8.8. *Minimum Norm Solution by MGS.*

Carry out MGS on $A^T \in R^{m \times n}$, with $\mathrm{rank}(A) = n$ to give $Q_1 = (q_1, \dots, q_n)$ and R. Then the minimum norm solution $y = y^{(0)}$ is obtained from

$$
\begin{aligned}
& R^T (\zeta_1, \dots, \zeta_n)^T = c; \\
& y^{(n)} = 0; \\
& \text{for } k = n, \dots, 2, 1 \\
& \quad \omega_k = q_k^T y^{(k)}; \quad y^{(k-1)} = y^{(k)} - (\omega_k - \zeta_k) q_k; \\
& \text{end}
\end{aligned}
$$

If the columns of Q_1 were orthogonal to working accuracy, then $\omega_k = 0$, $k = m : 1$. Hence, ω compensates for the lack of orthogonality to make this algorithm backwards stable!

$$A + E = \hat{Q}_1 \bar{R}, \qquad \hat{Q}_1^T \hat{Q}_1 = I, \quad \|E\|_2 \leq cu \|A\|_2. \tag{8.3.48}$$

Thus, $\bar{R}$ from MGS is comparable in accuracy to the upper triangular matrix from the Householder or Givens QR factorization applied to A alone.

A key observation for understanding the good numerical properties of the MGS algorithm is the suprising result that it can be interpreted as the Householder QR algorithm applied to the matrix A augmented with a square matrix of zero elements on top.[26] This is not true only in theory, but in the presence of rounding errors as well. Following Björck and Paige [43], we first look at the theoretical result.

Let $A \in \mathbf{R}^{m \times n}$ have rank n, and let O_n be square matrix of zeros. Consider the two QR factorizations, where we use Q for $m \times m$ and P for $(m+n) \times (m+n)$ orthogonal matrices,

$$A = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad \tilde{A} = \begin{pmatrix} O_n \\ A \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}. \tag{8.3.49}$$

[26]This observation was made by Charles Sheffield, apparently when comparing Fortran code for Householder and MGS QR factorization.

Since A has rank n, P_{11} is zero, P_{21} is an $m \times n$ matrix of orthogonal columns, and $A = Q_1 R = P_{21}\tilde{R}$. If the upper triangular matrices R and $\tilde{R}$ are both chosen to have positive diagonal elements in $A^T A = R^T R = \tilde{R}^T \tilde{R}$, then by uniqueness $R = \tilde{R}$ and $P_{21} = Q_1$. The last m columns of P are then arbitrary up to an $m \times m$ multiplier.

To see this, recall that the Householder transformation $Pa = e_1\rho$ uses

$$P = I - 2vv^T/\|v\|_2^2, \quad v = a - \rho e_1, \quad \rho = \pm\|a\|_2.$$

If (8.3.49) is obtained using Householder transformations, then

$$P^T = P_n \cdots P_2 P_1, \quad P_k = I - 2\hat{v}_k \hat{v}_k^T/\|\hat{v}_k\|_2^2, \quad k = 1:n, \tag{8.3.50}$$

where the vectors $\hat{v}_k$ are decribed below. Now, from MGS applied to $A^{(1)} = A$, $\rho_{11} = \|a_1^{(1)}\|_2$, and $a_1^{(1)} = q_1' = q_1\rho_{11}$. Thus for the first Householder transformation applied to the augmented matrix

$$\tilde{A}^{(1)} \equiv \begin{pmatrix} O_n \\ A^{(1)} \end{pmatrix}, \qquad \tilde{a}_1^{(1)} = \begin{pmatrix} 0 \\ a_1^{(1)} \end{pmatrix},$$

$$\hat{v}_1^{(1)} \equiv \begin{pmatrix} -e_1\rho_{11} \\ q_1' \end{pmatrix} = \rho_{11} v_1, \qquad v_1 = \begin{pmatrix} -e_1 \\ q_1 \end{pmatrix}.$$

Since there can be no cancellation we take $\rho_{kk} \geq 0$. But $\|v_1\|_2^2 = 2$, giving

$$P_1 = I - 2\hat{v}_1\hat{v}_1^T/\|\hat{v}_1\|_2^2 = I - 2v_1v_1^T/\|v_1\|_2^2 = I - v_1v_1^T,$$

and

$$P_1\tilde{a}_j^{(1)} = \tilde{a}_j^{(1)} - v_1v_1^T\tilde{a}_j^{(1)} = \begin{pmatrix} 0 \\ a_j^{(1)} \end{pmatrix} - \begin{pmatrix} -e_1 \\ q_1 \end{pmatrix} q_1^T a_j^{(1)} = \begin{pmatrix} e_1\rho_{1j} \\ a_j^{(2)} \end{pmatrix},$$

so

$$P_1\tilde{A}^{(1)} = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1n} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & a_2^{(2)} & \cdots & a_n^{(2)} \end{pmatrix}.$$

These values are clearly *numerically* the same as in the first step of MGS on A. We see the next Householder transformation produces the second row of R and $a_3^{(3)}, \ldots, a_n^{(3)}$, just as in MGS. Carrying on this way we see this Householder QR is numerically equivalent to MGS applied to A, and that every P_k is effectively defined by Q_1, since

$$P_k = I - v_kv_k^T, \qquad v_k = \begin{pmatrix} -e_k \\ q_k \end{pmatrix}, \quad k = 1:n. \tag{8.3.51}$$

From the numerical equivalence it follows that the backward error analysis for the Householder QR factorization of the augmented matrix can also be applied to the modified Gram–Schmidt algorithm on A. Let $\bar{Q}_1 = (\bar{q}_1, \dots, \bar{q}_n)$ be the matrix of vectors computed by MGS, and for $k = 1, \dots, n$ define

$$\bar{v}_k = \begin{pmatrix} -e_k \\ \bar{q}_k \end{pmatrix}, \qquad \bar{P}_k = I - \bar{v}_k \bar{v}_k^T, \qquad \bar{P} = \bar{P}_1 \bar{P}_2 \dots \bar{P}_n, \tag{8.3.52}$$

$$\tilde{q}_k = \bar{q}_k / \|\bar{q}_k\|_2, \quad \tilde{v}_k = \begin{pmatrix} -e_k \\ \tilde{q}_k \end{pmatrix}, \quad \tilde{P}_k = I - \tilde{v}_k \tilde{v}_k^T, \quad \tilde{P} = \tilde{P}_1 \tilde{P}_2 \dots \tilde{P}_n.$$

Then $\bar{P}_k$ is the computed version of the Householder matrix applied in the kth step of the Householder QR factorization of the augmented matrix and $\tilde{P}_k$ is its orthonormal equivalent, so that $\tilde{P}_k^T \tilde{P}_k = I$. From the error analysis for Householder QR (see Theorem 8.3.8 it follows that for $\bar{R}$ computed by MGS,

$$\begin{pmatrix} E_1 \\ A + E_2 \end{pmatrix} = \tilde{P} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \quad \bar{P} = \tilde{P} + E',$$

where

$$\|E_i\|_2 \le c_i u \|A\|_2, \quad i = 1, 2; \quad \|E'\|_2 \le c_3 u, \tag{8.3.53}$$

and c_i are constants depending on m, n and the details of the arithmetic. Using this result it can be shown that there exist an *exactly orthonormal matrix* $\hat{Q}_1$ and E such that

$$A + E = \hat{Q}_1 R, \quad \|E\|_2 \le cu \|A\|_2.$$

Thus $\bar{R}$ from MGS is comparable in accuracy to the upper triangular matrices obtained from Householder or Givens QR factorization.

8.3.6 Condition and Error Estimation

Using QR factorization with column pivoting a lower bound for $\kappa(A) = \kappa(R)$ can be obtained from the diagonal elements of R. We have $|r_{11}| \le \sigma_1 = \|R\|_2$, and since the diagonal elements of R^{-1} equal r_{ii}^{-1}, $i = 1 : n$, it follows that $|r_{nn}^{-1}| \le \sigma_n^{-1} = \|R^{-1}\|_2$, provided $r_{nn} \ne 0$. Combining these estimates we obtain the *lower bound*

$$\kappa(A) = \sigma_1 / \sigma_n \ge |r_{11} / r_{nn}| \tag{8.3.54}$$

Although this may considerably underestimate $\kappa(A)$, it has proved to give a fairly reliable estimate in practice. Extensive numerical testing has shown that (8.3.54) usually underestimates $\kappa(A)$ only by a factor of 2–3, and seldom by more than 10.

When column pivoting has not been performed, the above estimate of $\kappa(A)$ is not reliable. Then a condition estimator similar to that described in Sec. 7.6.5 can be used. Let u be a given vector and define v and w from

$$R^T v = u, \qquad Rw = v.$$

We have $w = R^{-1}(R^{-T} u) = (A^T A)^{-1} u$ so this is equivalent to one step of inverse iteration with $A^T A$, and requires about $0(n^2)$ multiplications. Provided that u is suitably chosen (cf. Sec. 7.6.5)

$$\sigma_n^{-1} \approx \|w\|_2 / \|v\|_2$$

will usually be a good estimate of σ_n^{-1}. We can also take u as a random vector and perform and 2–3 steps of inverse iteration. This condition estimator will usually detect near rank deficiency even in the case when this is not revealed by a small diagonal element in R.

More reliable estimates can be based on the componentwise error bounds. Assume that the perturbations δA and δb satisfy

$$|\delta A| \leq \omega E, \qquad |\delta b| \leq \omega f. \tag{8.3.55}$$

Substituting this in (8.2.29)–(8.2.30) yields the component-Wise bounds

$$|\delta x| \lessapprox \omega \left(|A^\dagger|(f + E|x|) + |(A^T A)^{-1}|E^T|r| \right), \tag{8.3.56}$$

$$|\delta r| \lessapprox \omega \left(|I - AA^\dagger|(f + E|x|) + |(A^\dagger)^T|E^T|r| \right). \tag{8.3.57}$$

where terms of order $O(\omega^2)$ have been neglected. In particular, if $E = |A|$, $f = |b|$, we obtain taking norms

$$\|\delta x\| \lessapprox \omega \left(\| \, |A^\dagger|(|b| + |A||x|)\| + \| \, |(A^T A)^{-1}||A|^T|r| \, \| \right), \tag{8.3.58}$$

$$\|\delta r\| \lessapprox \omega \left(\| \, |I - AA^\dagger|(|A||x| + |b|)\| + \| \, |(A^\dagger)^T||A|^T|r| \, \| \right). \tag{8.3.59}$$

This estimate has the form

$$\|\delta x\|_\infty \leq \omega(\| \, |B_1|g_1\|_\infty + \| \, |B_2|g_2\|_\infty), \tag{8.3.60}$$

where

$$B_1 = A^\dagger, \quad g_1 = |b| + |A||x|, \quad B_2 = (A^T A)^{-1}, \quad g_2 = |A^T||r|. \tag{8.3.61}$$

Consider now a general expression of the form $\| \, |B^{-1}|d\|_\infty$, where $d > 0$ is a known nonnegative vector. Writing $D = \mathrm{diag}\,(d)$ and $e = (1, 1, \ldots, 1)$, we have[27]

$$\| \, |B^{-1}|d\|_\infty = \| \, |B^{-1}|De\|_\infty = \| \, |B^{-1}D|e\|_\infty = \| \, |B^{-1}D|\|_\infty = \|B^{-1}D\|_\infty. \tag{8.3.62}$$

Hence, the problem is equivalent to that of estimating $\|C\|_\infty$, where $C = B^{-1}D$. There are algorithms that produce reliable order-of-magnitude estimates of $\|C^T\|_1 = \|C\|_\infty$ using only a few matrix-vector products of the form Cx and $C^T y$ for some carefully selected vectors x and y. Since these are rather tricky we will not describe them in detail here. An excellent discussion is given in Higham [211, Chapter 15].

If A has full rank and $A = QR$ then $A^\dagger = R^{-1}Q^T$ and $(A^\dagger)^T = QR^{-T}$. Hence, the required products can be computed inexpensively.

8.3.7 Iterative Refinement of Least Squares Solutions

The solution x to the least squares problem and the scaled residuals $\alpha y = r$, $\alpha > 0$, satisfy the augmented system

$$\begin{pmatrix} \alpha I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix} \tag{8.3.63}$$

[27] This clever observation is due to Arioli, Demmel, and Duff [9].

where $c = 0$. From Theorem we know that the extreme singular values of the system matrix equal

$$\sigma_1(M(\alpha)) = \frac{\alpha}{2} + \left(\frac{\alpha^2}{4} + \sigma_1^2\right)^{1/2},$$
$$\sigma_{m+n}(M(\alpha)) = -\frac{\alpha}{2} + \left(\frac{\alpha^2}{4} + \sigma_n^2\right)^{1/2}.$$

If we use a solution algorithm for (8.3.63) which is numerically invariant under a scaling α of the $(1,1)$-block then the relevant condition number is the smallest condition number of $M(\alpha)$. It can be shown that the minimum occurs for $\alpha^2 = \frac{1}{2}\sigma_n$ and

$$\min_\alpha \kappa(M(\alpha)) = \frac{1}{2} + \left(\frac{1}{4} + 2\kappa(A)^2\right)^{1/2} \leq 2\kappa(A), \tag{8.3.64}$$

where $\kappa(A) = \sigma_1/\sigma_n$.

We now show how to use the QR factorization to solve the augmented system (8.3.63). Let

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}, \quad Q \in \mathbf{R}^{m\times m}, \quad R \in \mathbf{R}^{n\times n}.$$

Using this we can transform the system (8.3.63) into

$$\begin{pmatrix} I & \begin{pmatrix} R \\ 0 \end{pmatrix} \\ (R^T \quad 0) & 0 \end{pmatrix} \begin{pmatrix} Q^T y \\ x \end{pmatrix} = \begin{pmatrix} Q^T b \\ c \end{pmatrix}.$$

It is easily verified that this gives the solution method

$$z = R^{-T}c, \qquad \begin{pmatrix} d \\ e \end{pmatrix} = Q^T b, \qquad y = Q\begin{pmatrix} z \\ e \end{pmatrix}, \qquad x = R^{-1}(d - z). \tag{8.3.65}$$

Here, if $c = 0$ then $z = 0$ and we retrieve the Householder QR algorithm for linear least squares problems. If $b = 0$, then $d = f = 0$, and (8.3.65) gives the QR algorithm for the minimum norm solution of $A^Ty = c$. Clearly the algorithm in (8.3.65) is numerically invariant under a scaling of the

A simple way to improve the accuracy of a solution $\bar{x}$ computed by the method of normal equations is by fixed precision iterative refinement, see Sec. 7.7.8. This requires that the data matrix A is saved and used to compute the residual vector $b - A\bar{x}$. In this way information lost when A^TA was formed can be recovered. If also the corrections are computed from the normal equations we obtain the following algorithm:

Iterative Refinement with Normal Equations:

Set $x_1 = \bar{x}$, and for $s = 1, 2, \ldots$ until convergence do

$$r_s := b - Ax_s, \qquad R^T R\delta x_s = A^T r_s,$$
$$x_{s+1} := x_s + \delta x_s.$$

Here R is computed by Cholesky factorization of the matrix of normal equation A^TA. This algorithm only requires one matrix-vector multiplication each with A and A^T and the solution of two triangular systems. Note that the first step, i.e., for $i = 0$, is identical to solving the normal equations. It can be shown that initially the errors will be reduced with rate of convergence equal to

$$\bar{\rho} = cu\kappa'(A^TA), \tag{8.3.66}$$

where c is a constant depending on the dimensions m, n. Several steps of the refinement may be needed to get good accuracy. (Note that $\bar{\rho}$ is proportional to $\kappa'(A^TA)$ even when no scaling of the normal equations has been performed!)

Example 8.3.4.

If $\kappa'(A^TA) = \kappa(A^TA)$ and $c \approx 1$ the error will be reduced to a backward stable level in p steps if $\kappa^{1/2}(A^TA) \leq u^{-p/(2p+1)}$. (As remarked before $\kappa^{1/2}(A^TA)$ is the condition number for a small residual problem.) For example, with $u = 10^{-16}$, the maximum value of $\kappa^{1/2}(A^TA)$ for different values of p are:

$$10^{5.3}, 10^{6.4}, 10^{8}, \qquad p = 1, 2, \infty.$$

For moderately ill-conditioned problems the normal equations combined with iterative refinement can give very good accuracy. For more ill-conditioned problems the methods based QR factorization described in Secs. 8.3 and 8.4 are usually to be preferred.

In Sec. 7.7.3 we considered mixed precision iterative refinement to compute an accurate solution $\bar{x}$ to a linear system $Ax = b$. In this scheme the residual vector $\bar{r} = b - A\bar{x}$ is computed in high precision. Then the system $A\delta = \bar{r}$ for a correction δ to $\bar{x}$ using a lower precision LU factorization of A. If this refinement process is iterated we obtain a solution with an accuracy comparable to that obtained by doing all computations in high precision. Moreover the overhead cost of the refinement is small.

We would like to have a similar process to compute highly accurate solutions to the linear least squares problems $\min_x \|Ax - b\|_2$. In the naive approach, we would then compute the residual $\bar{r} = b - A\bar{x}$ in high precision and then solve $\min_x \|A\delta x - \bar{r}\|_2$ for a correction δ. If a QR factorization in low precision of A is known we compute

$$\delta x = R^{-1}Q^T\bar{r},$$

and then iterate the process. The accuracy of this refinement scheme is not satisfactory unless the true residual $r = b - Ax$ of the least squares problem equals zero. The solution to an efficient algorithm for iterative refinement is to apply the refinement to the augmented system and refine both the solution x and the residual r in each step. In floating-point arithmetic with base β this process of iterative refinement can be described as follows:

$$s := 0; \quad x^{(0)} := 0; r^{(0)} := b;$$

repeat
  $f^{(s)} := b - r^{(s)} - Ax^{(s)}$;
  $g^{(s)} := c - A^T r^{(s)}$; (in precision $u_2 = \beta^{-t_2}$)
  solve augmented system in precision $u_1 = \beta^{-t_1}$)
  $x^{(s+1)} := x^{(s)} + \delta x^{(s)}$;
  $r^{(s+1)} := r^{(s)} + \delta r^{(s)}$;
  $s := s + 1$;
end

Using the Householder QR factorization with the computed factors $\bar{Q}$ and $\bar{R}$ the method in (8.3.65) can be used to solve for the corrections giving

$$z^{(s)} = \bar{R}^{-T} g^{(s)}, \qquad \begin{pmatrix} d^{(s)} \\ e^{(s)} \end{pmatrix} = \bar{Q}^T f^{(s)}, \tag{8.3.67}$$

$$\delta r^{(s)} = \bar{Q} \begin{pmatrix} z^{(s)} \\ e^{(s)} \end{pmatrix}, \qquad \delta x^{(s)} = \bar{R}^{-1}(d^{(s)} - z^{(s)}). \tag{8.3.68}$$

Recall that i $\bar{Q} = P_n \cdots P_2 P_1$, where P_i are Householder reflections, then $\bar{Q}^T = P_1 P_2 \cdots P_n$. The computation of the residuals and corrections, takes $4mn$ flops in high precision. Computing the solution from (8.3.67)–(8.3.68) takes $2n^2$ for operations with $\bar{R}$ and takes $8mn - 4n^2$ for operations with $\bar{Q}$. The total work for a refinement step is an order of magnitude less than the $4n^3/3$ flops required for the QR factorization.

A portable and parallelizable implementation of this algorithm using the extended precision BLAS is available and described in [94].

Review Questions

3.1 Let $w \in \mathbf{R}^n$, $\|w\|_2 = 1$. Show that $I - 2ww^T$ is orthogonal. What is the geometrical significance of the matrix $I - 2ww^T$? Give the eigenvalues and eigenvectors of these matrices.

3.2 Define a Givens transformations $G_{ij}(\phi) \in \mathbf{R}^{n \times n}$. Give a geometrical interpretations for the case $n = 2$.

3.3 Describe the difference between the classical and modified Gram–Schmidt methods for computing the factorization $A = Q_1 R$. What can be said about the orthogonality of the computed matrix Q_1 for these two algorithms?

3.4 Define the QR factorization of a matrix $A \in \mathbf{R}^{m \times n}$, in the case that rank $(A) = n \leq m$. What is its relation to the Cholesky factorization of $A^T A$?

Problems

3.1 Compute using Householder reflectors P_1, P_2, the factorization

$$Q^T A = P_2 P_1 A = \begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad A = (a_1, a_2) = \begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{pmatrix},$$

to four decimal places

3.2 Solve the least squares problem $\min_x \|Ax - b\|_2$, where

$$\begin{pmatrix} \sqrt{2} & 0 \\ 1 & -1 \\ 1 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

using a QR factorization computed with Givens transformation;

3.3 (a) Derive a *square root free* version of the modified Gram–Schmidt orthogonalization method, by omitting the normalization of the vectors $\tilde{q}_k$. Show that this version computes a factorization

$$A = \tilde{Q}_1 \tilde{R},$$

where $\tilde{R}$ is **unit** upper triangular.

(b) Suppose the square root free version of modified Gram–Schmidt is used. Modify Algorithm 8.3.3 for computing the least squares solution and residual from this factorization.

Comment: There is no square root free version of Householder QR factorization!

3.4 For any c and s such that $c^2 + s^2 = 1$ we have

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} 0 & s \\ 0 & c \end{pmatrix} = QR.$$

Here rank $(A) = 1 < 2 = n$. Show that the columns of Q do not provide any information about an orthogonal basis for $\mathcal{R}(A)$.

3.5 (a) If the matrix Q in the QR factorization is explicitly required in the Householder algorithm it can be computed by setting $Q^{(n)} = I_m$, and computing $Q = Q^{(0)}$ by the backward recursion

$$Q^{(k-1)} = P_k Q^{(k)}, \quad k = n : -1 : 1.$$

Show that if advantage is taken of the property that $P_k = \mathrm{diag}\,(I_{k-1}, \tilde{P}_k)$ this accumulation requires $4(mn(m-n) + n^3/3)$ flops. What is the corresponding operation count if forward recursion is used?

(b) Show how we can compute

$$Q_1 = Q \begin{pmatrix} I_n \\ 0 \end{pmatrix}, \qquad Q_2 = Q \begin{pmatrix} 0 \\ I_{m-n} \end{pmatrix}$$

separately in $2(mn^2 - n^3/3)$ and $2(2m^2n - 3mn^2 + n^3)$ multiplications, respectively.

3.6 Let $Q = Q_1 = (q_1, q_2, \dots, q_n) \in \mathbf{R}^{n\times n}$ be a real orthogonal matrix.

(a) Determine a reflector $P_1 = I - 2v_1 v_1^T$, such that $P_1 q_1 = e_1 = (1, 0, \dots, 0)^T$, and show that $P_1 Q_1 = Q_2$ has the form

$$Q_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \tilde{Q}_2 & \\ 0 & & & \end{pmatrix},$$

where $\tilde{Q}_2 = (\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_n) \in \mathbf{R}^{(n-1)\times(n-1)}$ is a real orthogonal matrix.

(b) Show, using the result in (a), that Q can be transformed to diagonal form with a sequence of orthogonal transformations

$$P_{n-1} \cdots P_2 P_1 Q = \mathrm{diag}\,(1, \dots, 1, \pm 1).$$

3.7 Show how to compute the QR factorization of the product $A = A_p \cdots A_2 A_1$ without explicitly forming the product matrix A.

Hint: For $p = 2$ first determine Q_1 such that $Q_1^T A_1 = R_1$, and form $A_2 Q_1$. Then, if Q_2 is such that $Q_2^T A_2 Q_1 = R_2$ it follows that $Q_2^T A_2 A_1 = R_2 R_1$.

3.8 Test the recursive QR algorithm `recqr(A)` given in Sec. sec8.2.rec on some matrices. Check that you obtain the same result as from the built-in function `qr(A)`.

3.9 (Stewart and Stewart [349]).

If properly implemented, hyperbolic Householder transformations have the same good stability as the mixed scheme of hyperbolic rotations.

(a) Show that the hyperbolic rotations $\breve{G}$ can be rewritten as

$$\breve{G} = \frac{1}{c}\begin{pmatrix} 1 & -s \\ -s & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \frac{1}{c}\begin{pmatrix} t \\ -s/t \end{pmatrix}\begin{pmatrix} t & -s/t \end{pmatrix}, \quad t = \sqrt{1-c},$$

which now has the form of a hyperbolic Householder transformation. If $\breve{G}$ is J-orthogonal so is

$$J\breve{G} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{c}\begin{pmatrix} t \\ s/t \end{pmatrix}\begin{pmatrix} t & -s/t \end{pmatrix}, \quad t = \sqrt{1-c},$$

(b) Show that the transformation can be computed form

$$S\breve{G}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \gamma\begin{pmatrix} 1 \\ s/(1-c) \end{pmatrix}, \qquad \gamma = \frac{1}{c}((1-c)x - sy).$$

8.4 Rank Deficient Problems

8.4.1 Modified Least Squares Problems

In many applications it is desired to solve a sequence of related least squares problems, where in each step a simple modification of the data $(A,\ b)$ is performed. The following type of modifications frequently arise:

1. A general rank-one change to to $(A\ b)$.
2. Adding (deleting) a row of $(A\ b)$.
3. Deleting (adding) a column of A and a component of x.

In various time-series problems data are arriving sequentially and a least squares solution has to be updated at each time step. Such modifications are usually referred to as updating when (new) data is added and down-dating when (old) data is removed. Applications in signal processing often require real-time solutions so efficiency is critical.

Other applications arise in optimization and statistics. Indeed, the first systematic use of such algorithms seems to have been in optimization. In linear regression efficient and stable procedure for adding and/or deleting observations is needed. In stepwise regression one wants to examine different models by adding and/or deleting variables in each step. Another important application occurs in active set methods for solving least squares problems with inequality constraints.

Assume that $A \in \mathbf{R}^{m\times n}$, $m > n$, is nonsingular. If a column is deleted from A, then by the interlacing property (Theorem 8.1.15) the smallest singular value of the modified matrix will not decrease. On the other hand, when a column is added the modified matrix may become singular. This indicates that adding a column is a more delicate problem than removing a column. In general, we can not expect modification methods to be stable when the unmodified problem is much worse conditioned than the modified problem,

Similarly, when a row is added the smallest singular value will not decrease; when a row is deleted the rank can decrease. Thus, deleting a column and adding a row are "easy" operations, whereas adding a column and deleting a row are more delicate.

We first note that there is a simple relationship between the problem of updating matrix factorizations and that of updating the least squares solutions. Recall that if A has full column rank and the R-factor of the matrix (A, b) is

$$\begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}, \tag{8.4.1}$$

then the solution to the least squares problem $\min_x \|Ax - b\|_2$ is given by

$$Rx = z, \qquad \|Ax - b\|_2 = \rho. \tag{8.4.2}$$

The upper triangular matrix (8.4.1) can be computed either from the QR decomposition of (A, b) or as the Cholesky factor of $(A, b)^T(A, b)$. Hence, updating algorithms for matrix factorizations applied to the extended matrix (A, b) give updating algorithms for least squares solutions.

The modification of the singular value decomposition $A = U\Sigma V^T$ under a rank one change in A will in general require $O(mn^2)$ flops. Since this is the same order of operations as for recomputing the SVD from scratch, algorithms for modifying the SVD will not be considered here.

The updating of the Householder QR decomposition of A, where Q is stored as a product of Householder transformations, is not feasible. This is because there seems to be no efficient way to update the Householder transformations when, for example, a row is added. Therefore, we develop methods for updating the factorization

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{8.4.3}$$

where the orthogonal factor $Q \in \mathbf{R}^{m\times m}$ is stored explicitly. These updating algorithms require $O(m^2)$ multiplications, and are (almost) normwise backward stable.

Assume that we know the full QR decomposition (8.4.3) of A. For a general rank one change we want to compute the decomposition

$$\tilde{A} = A + uv^T = \tilde{Q}\begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \tag{8.4.4}$$

where $u \in \mathbf{R}^m$ and $v \in \mathbf{R}^n$ are given. For simplicity we assume that $\operatorname{rank}(A) = \operatorname{rank}(\tilde{A}) = n$, so that R and $\tilde{R}$ are uniquely determined.

To update a least squares solution we can apply the updating procedure to compute the QR decomposition of

$$\begin{pmatrix} A + uv^T & b \end{pmatrix} = \begin{pmatrix} A & b \end{pmatrix} + u\begin{pmatrix} v^T & 0 \end{pmatrix}.$$

where the right hand side has been appended. For simplicity of notation we will therefore in the following not include the right hand side explicitly in the description of the algorithms.

For simplicity we assume that $\operatorname{rank}(A) = \operatorname{rank}(\tilde{A}) = n$, so that R and $\tilde{R}$ are uniquely determined. Then we proceed as follows. We first compute $w = Q^T u \in \mathbf{R}^m$, so that

$$A + uv^T = Q\left[\begin{pmatrix} R \\ 0 \end{pmatrix} + wv^T\right]. \tag{8.4.5}$$

Next we determine a sequence of Givens rotations $J_k = G_{k,k+1}(\theta_k)$, $k = m-1, \dots, 1$ such that

$$J_1^T \cdots J_{m-1}^T w = \alpha e_1, \qquad \alpha = \pm\|w\|_2.$$

Note that these transformations zero the last $m-1$ components of w from bottom up. (For details on how to compute J_k see Algorithm 2.3.1.) If these transformations are applied to the R-factor in (8.4.5) we obtain

$$\tilde{H} = J_1^T \cdots J_{m-1}^T\left[\begin{pmatrix} R \\ 0 \end{pmatrix} + wv^T\right] = H + \alpha e_1 v^T. \tag{8.4.6}$$

(Note that $J_{n+1}, \dots, J_{m-1}$ have no effect on R.) Because of the structure of the Givens rotations the matrix H will be an upper Hessenberg matrix, i.e., H is triangular except for extra nonzero elements $h_{k+1,k}, k = 1, 2, \dots, n$ (e.g., $m = 6$, $n = 4$),

$$H = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since only the first row of H is modified by the term $\alpha e_1 v^T$, $\tilde{H}$ is also upper Hessenberg. Then we can determine Givens rotations $\tilde{J}_k = G_{k,k+1}(\phi_k)$, $k = 1, \dots, n$, which zero the element in position $(k+1, k)$, so that

$$\tilde{J}_n^T \cdots \tilde{J}_1^T \tilde{H} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} \tag{8.4.7}$$

is upper triangular. Finally, the transformations are accumulated into Q to get

$$\tilde{Q} = QU, \qquad U = J_{m-1} \cdots J_1 \tilde{J}_1 \cdots \tilde{J}_n.$$

$\tilde{Q}$ and $\tilde{R}$ now give the desired decomposition (8.4.4). The work needed for this update is as follows: computing $w = Q^T u$ takes m^2 flops. Computing H and $\tilde{R}$ takes $4n^2$ flops and accumulating the transformations J_k and $\tilde{J}_k$ into Q takes $4(m^2 + mn)$ flops for a total of $5m^2 + 4mn + 4n^2$ flops. Hence, the work has been decreased from $O(mn^2)$ to $O(m^2)$. However, if n is small updating may still be more expensive than computing the decomposition from scratch.

Algorithms for modifying the full QR decomposition of (A, b) when a column is deleted or added are special cases of the rank one change; see Björck [40, Sec. . 3.2]. Adding a row is simple, since the QR factorization is easily organized to treat a row at a time.

A more subtle problem is to modify the QR decomposition when a row is *deleted*, which is called the **down-dating** problem. This corresponds to the problem of deleting an observation in a least squares problem. For example, in the sliding window method, when a new data row has been added, one wants simultaneously to delete an old data row. Another instance when a data row has to be removed is when it has somehow been identified as faulty.

There is no loss of generality in assuming that it is the *first row* of A that is to be deleted. We wish to obtain the QR decomposition of the matrix $\tilde{A} \in \mathbf{R}^{(m-1)\times n}$ when

$$A = \begin{pmatrix} a_1^T \\ \tilde{A} \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \tag{8.4.8}$$

is known. We now show that this is equivalent to finding the QR decomposition of (e_1, A), where a dummy column $e_1 = (1, 0, \dots, 0)^T$ has been added. We have

$$Q^T(e_1, A) = \begin{pmatrix} q_1 & R \\ q_2 & 0 \end{pmatrix},$$

where $q^T = (q_1^T, q_2^T) \in \mathbf{R}^m$ is the *first row* of Q. We now determine Givens rotations $J_k = G_{k,k+1}$, $k = m-1, \dots, 1$, so that

$$J_1^T \cdots J_{m-1}^T q = \alpha e_1, \quad \alpha = \pm 1. \tag{8.4.9}$$

Then we have

$$J_1^T \cdots J_{m-1}^T \begin{pmatrix} q_1 & R \\ q_2 & 0 \end{pmatrix} = \begin{pmatrix} \alpha & v^T \\ 0 & \tilde{R} \\ 0 & 0 \end{pmatrix}, \tag{8.4.10}$$

where the matrix $\tilde{R}$ is upper triangular. Note that the transformations $J_{n+1}, \dots, J_{m-1}$ will not affect R. Further, if we compute

$$\bar{Q} = QJ_{m-1}\cdots J_1,$$

it follows from (8.4.9) that the first row of $\bar{Q}$ equals αe_1^T. Since $\bar{Q}$ is orthogonal it must have the form

$$\bar{Q} = \begin{pmatrix} \alpha & 0 \\ 0 & \tilde{Q} \end{pmatrix},$$

with $|\alpha| = 1$ and $\tilde{Q} \in \mathbf{R}^{(m-1)\times(m-1)}$ orthogonal. Hence, from (8.4.9),

$$\begin{pmatrix} a_1^T \\ \tilde{A} \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \tilde{Q} \end{pmatrix} \begin{pmatrix} v^T \\ \tilde{R} \\ 0 \end{pmatrix},$$

and hence the desired decomposition is

$$\tilde{A} = \tilde{Q}\begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}.$$

This algorithm for down-dating is a special case of the rank one change algorithm, and is obtained by taking $u = -e_1$, $v^T = a_1^T$ in (8.4.4).

8.4.2 Regularized Least Squares Solutions

Failure to detect possible rank deficiency in solving least squares problems can be catastrophic, since it may lead to a meaningless solution of very large norm, or even to failure of the algorithm.

Example 8.4.1.

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

$$\int_0^1 k(s,t)f(s)\,ds = g(t), \quad k(s,t) = e^{-(s-t)^2}, \tag{8.4.11}$$

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

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

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

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

For $n = 100$ the singular values σ_k of the matrix K computed in IEEE double precision are displayed in logarithmic scale in Figure 8.4.1. Note that for $k > 30$ all

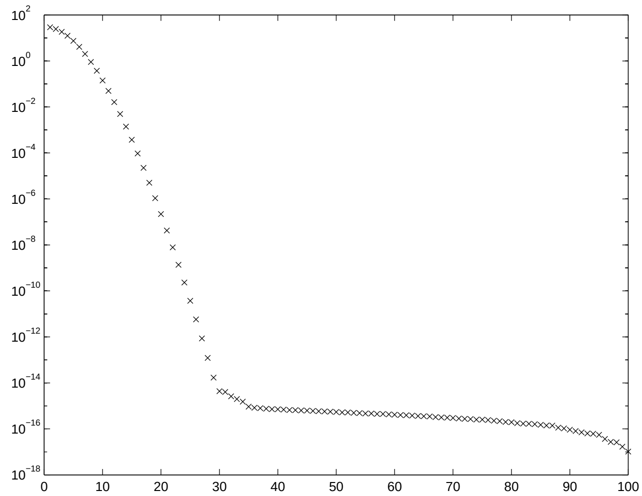

Figure 8.4.1. *Singular values of a numerically singular matrix.*

σ_k are close to roundoff level, so the numerical rank of K certainly is smaller than 30. This means that the linear system $Kf = g$ is *numerically under-determined* and has a meaningful solution only for special right-hand sides g.

The equation (8.4.11) is a **Fredholm integral equation** of the first kind. It is known that such equations are **ill-posed** in the sense that the solution f does not depend continuously on the right hand side g. This example illustrate how this inherent difficulty in the continuous problem carry over to the discretized problem!

The choice of the parameter δ in Definition 7.6.1 of numerical rank is not always an easy matter. If the errors in a_{ij} satisfy $|e_{ij}| \leq \epsilon$, for all i, j, an appropriate choice is $\delta = (mn)^{1/2}\epsilon$. On the other hand, if the absolute size of the errors e_{ij} differs widely, then Definition 7.6.1 is not appropriate. One could then scale the rows and columns of A so that the magnitude of the errors become nearly equal. (Note that any such diagonal scaling D_rAD_c will induce the same scaling D_rED_c of the error matrix.)

Consider the linear least squares problem

$$\min_x \|Ax - b\|_2, \tag{8.4.12}$$

where the matrix A is ill-conditioned and possibly rank deficient. If A has numerical rank equal to $k < n$, we can get a more stable approximative solution by discarding terms in the expansion (8.1.34) corresponding to singular values smaller or equal to δ, and take the solution as the **truncated SVD (TSVD) solution**

$$x(\delta) = \sum_{\sigma_i > \delta} \frac{c_i}{\sigma_i} v_i. \tag{8.4.13}$$

If $\sigma_k > \delta \geq \sigma_{k+1}$ then the TSVD solution is $x(\delta) = A_k^\dagger b$ and solves the related least squares problem

$$\min_x \|A_k x - b\|_2, \qquad A_k = \sum_{\sigma_i > \delta} \sigma_i u_i v_i^T,$$

where A_k is the best rank k approximation of A. We have

$$\|A - A_k\|_2 = \|AV_2\|_2 \leq \delta, \quad V_2 = (v_{k+1}, \dots, v_n).$$

In general the most reliable way to determine an approximate pseudo-inverse solution of a numerically rank deficient least squares problems is by first computing the SVD of A and then using an appropriate truncated SVD solution (8.4.13). However, this is also an expensive method. In practice the QR factorization often works as well, provided that some form of column pivoting is carried out. An alternative to truncated SVD is to add a quadratic constraint on the solution; see Sec. 8.6.4.

8.4.3 QR Factorization of Rank Deficient Matrices

Let $A \in \mathbf{R}^{m\times n}$ be a matrix with $\operatorname{rank}(A) = r \leq \min(m, n)$ and Π a permutation matrix such that the first r columns in $A\Pi$ are linearly independent. Then the QR factorization of $A\Pi$ will have the form

$$A\Pi = (Q_1 \quad Q_2) \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \tag{8.4.14}$$

where $R_{11} \in \mathbf{R}^{r\times r}$ is upper triangular with positive diagonal elements. Note that it is not required that $m \geq n$. The matrices Q_1 and Q_2 form orthogonal bases for $\mathcal{R}(A)$ and $\mathcal{N}(A^T)$, respectively. This factorization is not unique since there are many ways to choose Π.

To simplify notations we assume in the following that $\Pi = I$. (This is no restriction since the column permutation of A can always be assumed to have been carried out in advance.) Using (8.4.14) the least squares problem $\min_x \|Ax - b\|_2$ is reduced to

$$\min_x \left\| \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \right\|_2, \tag{8.4.15}$$

where $c = Q^T b$. Since R_{11} is nonsingular the first r equations can be satisfied exactly for any x_2. Hence, the general least squares solutions becomes

$$x_1 = R_{11}^{-1}(c_1 - R_{12}x_2) \tag{8.4.16}$$

where x_2 is arbitrary. Setting

$$x_2 = 0, \qquad x_1 = R_{11}^{-1} c_1,$$

we obtain a particular solution with at most $r = \operatorname{rank}(A)$ nonzero components. This solution is appropriate when we want to fit a vector b of observations using *as few columns of* A as possible. It depends on the initial column permutation and is not unique. Any solution x such that Ax only involves at most r columns of A, is called a **basic solution**.

In general we have $x_1 = d - Cx_2$, where

$$d = R_{11}^{-1}c_1, \qquad C = R_{11}^{-1}R_{12} \in \mathbf{R}^{r\times(n-r)}. \tag{8.4.17}$$

The pseudo-inverse solution $x = A^\dagger b$ is the least squares solution of minimum norm. From (8.4.16) it follows that this is obtained by solving the linear least squares problem for x_2

$$\min \left\| \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right\|_2 = \min_{x_2} \left\| \begin{pmatrix} d \\ 0 \end{pmatrix} - \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix} x_2 \right\|_2 . \tag{8.4.18}$$

Note that this problem has full rank and hence always has a unique solution x_2. The pseudo-inverse solution $x = A^\dagger b$ equals *the residual* of the problem (8.4.18).

To compute x_2 we could form and solve the normal equations $(I + CC^T)x_2 = C^T d$. It is preferable to compute the QR factorization

$$Q_C^T \begin{pmatrix} C \\ I_{n-r} \end{pmatrix} = \begin{pmatrix} R_C \\ 0 \end{pmatrix}, \qquad Q_C^T \begin{pmatrix} d \\ 0 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix},$$

where Q_C is a product of Householder transformations. We then have

$$x = A^\dagger b = Q_C \begin{pmatrix} 0 \\ d_2 \end{pmatrix}.$$

We remark that the pseudo-inverse solution can also be obtained by applying the modified Gram–Schmidt algorithm to

$$\begin{pmatrix} C & d \\ -I_{n-r} & 0 \end{pmatrix}. \tag{8.4.19}$$

to compute the residual of the problem (8.4.18).

We further note that

$$A \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix} z = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} R_{11}^{-1}R_{12} \\ -I_{n-r} \end{pmatrix} z = 0,$$

From this it follows that the null space of A is given by

$$\mathcal{N}(A) = \mathcal{R} \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix}. \tag{8.4.20}$$

8.4.4 Complete QR Factorizations

A **complete QR factorization** of $A \in \mathbf{R}^{m\times n}$ of rank r is a factorization of the form

$$A = Q \begin{pmatrix} T & 0 \\ 0 & 0 \end{pmatrix} V^T \tag{8.4.21}$$

where Q and V are orthogonal matrices and $T \in \mathbf{R}^{r\times r}$ is upper (or lower) triangular with positive diagonal elements. If Q and V are partitioned as

$$Q = (Q_1, Q_2), \qquad V = (V_1, V_2),$$

where $Q_1 \in \mathbf{R}^{m\times r}$, and $V_1 =\in \mathbf{R}^{n\times r}$, then Q_1 and V_2 give orthogonal bases for the range and null spaces of A. Similarly, V_1 and Q_2 give orthogonal bases for the range and null spaces of A^T.

To compute a complete QR factorization we start from the factorization in (8.4.14). A sequence of Householder reflections can then be found such that

$$(R_{11} \quad R_{12})\, P_r \cdots P_1 = (T \quad 0),$$

and hence $V = \Pi P_1 \cdots P_r$. Here P_k, $k = r, r-1, \dots, 1$, is constructed to zero elements in row k and only affect columns $k, r+1 : n$. These transformations require $r^2(n-r)$ multiplications.

Using the orthogonal invariance of the l_2-norm it follows that $x = V_1 R^{-1} Q_1^T b$ is the minimum norm solution of the least squares problem $\min_x \|Ax - b\|_2$. Since the pseudo-inverse is uniquely defined by this property, cf. Theorem 8.1.8, we find that the pseudo-inverse of A is

$$A^\dagger = V \begin{pmatrix} T^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T = V_1 T^{-1} Q_1^T. \tag{8.4.22}$$

In signal processing problems it is often the case that one wants to determine the rank of A as well as the range (signal subspace) and null space of A. Since the data analyzed arrives in real time it is necessary to update an appropriate matrix decompositions at each time step. For such applications the SVD has the disadvantage that it cannot in general be updated in less than $\mathcal{O}(n^3)$ operations, when rows and columns are added or deleted to A. Although the RRQR decomposition can be updated, it is less suitable in applications where a basis for the approximate null space of A is needed, since the matrix W in (8.4.20) cannot easily be updated.

For this reason we introduce the **URV decomposition**

$$A = URV^T = (U_1 \quad U_2) \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}, \tag{8.4.23}$$

where U and V are orthogonal matrices, $R_{11} \in \mathbb{R}^{k\times k}$, and

$$\sigma_k(R_{11}) \ge \frac{1}{c}\sigma_k, \qquad \big(\|R_{12}\|_F^2 + \|R_{22}\|_F^2\big)^{1/2} \le c\sigma_{k+1}. \tag{8.4.24}$$

Note that here both submatrices R_{12} and R_{22} have small elements.

From (8.4.23) we have

$$\|AV_2\|_2 = \left\| \begin{pmatrix} R_{12} \\ R_{22} \end{pmatrix} \right\|_F \le c\sigma_{k+1},$$

and hence the orthogonal matrix V_2 can be taken as an approximation to the numerical null space $\mathcal{N}_k$.

Algorithms for computing an URV decomposition start with an initial QR decomposition, followed by a rank revealing stage in which singular vectors corresponding to the smallest singular values of R are estimated. Assume that w is a unit vector such that $\|Rw\| = \sigma_n$. Let P and Q be a orthogonal matrices such that $Q^T w = e_n$ and $P^T RQ = \hat{R}$ where $\hat{R}$ is upper triangular. Then

$$\|\hat{R}e_n\| = \|P^T RQQ^T w\| = \|P^T Rw\| = \sigma_n,$$

which shows that the entire last column in $\hat{R}$ is small. Given w the matrices P and Q can be constructed as a sequence of Givens rotations, Algorithms can also be given for updating an URV decomposition when a new row is appended.

Like the RRQR decompositions the URV decomposition yield approximations to the singular values. In [274] the following bounds are derived

$$f\sigma_i \leq \sigma_i(R_{11}) \leq \sigma_i, \quad i = 1 : r,$$

and

$$\sigma_i \leq \sigma_{i-k}(R_{22}) \leq \sigma_i/f, \quad i = r+1 : n,$$

where

$$f = \left(1 - \frac{\|R_{12}\|_2^2}{\sigma_{\min}(R_{11})^2 - \|R_{22}\|_2^2}\right)^{1/2}.$$

Hence, the smaller the norm of the off-diagonal block R_{12}, the better the bounds will be. Similar bounds can be given for the angle between the range of V_2 and the right singular subspace corresponding to the smallest $n-r$ singular values of A.

An alternative decomposition that is more satisfactory for applications where an accurate approximate null space is needed, is the rank-revealing **ULV decomposition**

$$A = U \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} V^T. \tag{8.4.25}$$

where the middle matrix has lower triangular form. For this decomposition

$$\|AV_2\|_2 = \|L_{22}\|_F, \qquad V = (V_1,\ V_2),$$

and hence the size of $\|L_{21}\|$ does not adversely affect the null space approximation. On the other hand the URV decomposition usually gives a superior approximation for the numerical range space and the updating algorithm for URV is much simpler.

For recent work see also [20, 19, 130]

We finally mention that rank-revealing QR decompositions can be effectively computed only if the numerical rank r is either high, $r \approx n$ or low, $r \ll n$. The low rank case is discussed in [65]. Matlab templates for rank-revealing UTV decompositions are described in [131].

An advantage of the complete QR factorization of A is that V_2 gives an orthogonal basis for the null space $\mathcal{N}(A)$. This is often useful, e.g., in signal processing applications, where one wants to determine the part of the signal that corresponds to noise. The factorization (8.4.22) can be generalized to the case when A is only numerically rank deficient in a similar way as done above for the QR factorization. The resulting factorizations have one of the forms

$$A = Q \begin{pmatrix} R & F \\ 0 & G \end{pmatrix} V^T \qquad A = Q \begin{pmatrix} R^T & 0 \\ F^T & G^T \end{pmatrix} V^T \tag{8.4.26}$$

where R is upper triangular and

$$\sigma_k(R) > \frac{1}{c}, \qquad (\|F\|_F^2 + \|G\|_F^2)^{1/2} \le c\sigma_{k+1}.$$

An advantage is that unlike the SVD it is possible to efficiently update the factorizations (8.4.26) when rows/columns are added/deleted.

8.4.5 Subset Selection by SVD and RRQR

A related problems is the **subset selection** problem in which one wants to determine the $k < n$ most linearly independent columns of A. More precisely we want to find a permutation P such that the smallest singular value of the k first columns of AP are maximized. Given $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$ we want to find a permutation matrix Π such that the first k columns of $A\Pi$ are sufficiently independent and a vector $z \in \mathbf{R}^k$ which solves

$$\min_z \|A_1 z - b\|_2\,, \quad A_1 = AP \begin{pmatrix} I_k \\ 0 \end{pmatrix}. \tag{8.4.27}$$

The following SVD-based algorithm for finding a good approximation to the solution of the subset selection problem has been suggested by Golub, Klema, and Stewart [174]; see also Golub and Van Loan [184, Sec. 12.2]. First compute Σ and V in the SVD of A,

$$A = U\Sigma V^T, \quad \Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_n),$$

and use it to determine the numerical rank k of A. Note that the right singular vectors are the eigenvectors of A^TA, that is the principal components of the problem. If P is any permutation matrix, the

$$U^T(AP)(P^TV) = \Sigma.$$

so that permutation of the columns of A correspond to a permutation of the rows of the orthogonal matrix V of right singular vectors. The theoretical basis for this selection strategy is the following result; see [174, Theorem 6.1]

Theorem 8.4.1.

Let $A = U\Sigma V^T \in \mathbf{R}^{m\times n}$ be the SVD of A. Partition A and V conformally as $A = (\, A_1 \quad A_2 \,)$ and $V = (\, V_1 \quad V_2 \,)$, where

$$(\, V_1 \quad V_2 \,) = \begin{array}{c} \\ k \\ n-k \end{array} \!\!\begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \end{array}, \tag{8.4.28}$$

and $\hat{V}_{11}$ *is nonsingular. Then*

$$\frac{\sigma_k(A)}{\|V_{11}^{-1}\|_2} \le \sigma_k(A_1) \le \sigma_k(A). \tag{8.4.29}$$

Proof. The upper bound in (8.4.29) follows directly. If we set $A\,(\,V_1 \quad V_2\,) = (\,S_1 \quad S_2\,)$, then $S_1^T S_2 = 0$. Now since $A = SV^T$, we have

$$A_1 = S_1 V_{11}^T + S_2 V_{21}^T.$$

If we define $\inf(A) = \inf_{\|x\|_2=1} \|Ax\|_2$, then

$$\inf(A_1) \ge \inf(S_1 V_{11}) \ge \sigma_k(A) \inf(V_{11}).$$

Since $\inf(V_{11}) = \|V_{11}^{-1}\|_2$ the lower bound follows. □

This result suggest that we choose the permutation P such that the with $\hat{V} = VP$ the resulting matrix $\hat{V}_{11}$ is as well-conditioned as possible. This can be achieved by using QR factorization with column pivoting to compute

$$Q^T(V_{11}^T,\ V_{21}^T)P = (R_{11},\ R_{12}).$$

The least squares problem (8.4.27) can then be solved by QR factorization.

From Theorem 8.1.22 it follows that in (8.4.28) $\|V_{11}\|_2 = \|V_{22}\|_2$. When $k > n/2$ it is more economical to use the QR factorization with column pivoting applied to the smaller submatrix $V_2^T = (V_{12}^T,\ V_{22}^T)$. This will then indicate *what columns in A to drop.*

In case $k = n-1$, the column permutation P corresponds to the largest component in the right singular vector $V_2 = v_n$ belonging to the smallest singular value σ_n. The index indicates what column to drop. In a related strategy due to T. F. Chan [63] repeated use is made of this, by dropping one column at a time. The right singular vectors needed are determined by inverse iteration (see Sec. 9.4.3).

A comparison between the RRQR and the above SVD-based algorithms is given by Chan and Hansen [64, 1992]. Although in general the methods will not necessarily compute equivalent solutions, the subspaces spanned by the two sets of selected columns are still almost identical whenever the ratio σ_{k+1}/σ_k is small.

8.4.6 Bidiagonalization and Partial Least Squares

So far we have considered methods for solving least squares problems based on reducing the matrix $A \in \mathbf{R}^{m\times n}$ to upper triangular (or trapezoidal) form using unitary operations. It is possible to carry this reduction further and obtain a lower bidiagonal matrix ($m \ge n$)

$$A = UBV^T, \quad B \in \mathbf{R}^{m\times n}, \tag{8.4.30}$$

where U and V are orthogonal matrices. The bidiagonal form is the closest that can be achieved by a finite process to the diagonal form in SVD. Th bidiagonal decomposition (8.4.30) therefore is usually the first step in computing the SVD of A; see Sec. 9.7. It is also powerful tool for solving various least squares problems.

Note that from (8.4.30) it follows that $A^T = VB^TU^T$, where B^T is upper bidiagonal. Thus, if we apply the same process to A^T we obtain an *upper* bidiagonal form. A complex matrix $A \in \mathbf{C}^{m\times n}$ can be reduced by a similar process to *real* bidiagonal form using unitary transformations U and V. We consider here only the real case, since the generalization to the complex case is straightforward.

The simplest method for constructing the bidiagonal decomposition is the **Golub–Kahan algorithm** in which the reduction is achieved by applying a sequence of Householder reflections alternately from left and right. We set $A = A^{(1)}$ and in the first double step compute

$$A^{(2)} = Q_1(AP_1) = \begin{pmatrix} \alpha_1 & 0 & \cdots & 0 \\ \beta_2 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ 0 & \tilde{a}_{32} & \cdots & \tilde{a}_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{a}_{m2} & \cdots & \tilde{a}_{mn} \end{pmatrix}.$$

Here the Householder reflection P_1 is chosen to zero the last $n-1$ elements in the first row of A. Next Q_1 is chosen to zero the last $m-2$ elements in the the first column of AP_1. This key thing to observe is that Q_1 will leave the first row in AP_1 unchanged and thus not affect the zeros introduced by P_1. All later steps are similar. In the kth step, $k = 1 : \min(m,n)$, we compute

$$A^{(k+1)} = Q_k(A^{(k)}P_k),$$

where Q_k and P_k are Householder reflections. Here P_k is chosen to zero the last $n-k$ elements in the kth row of $A^{(k)}$. Then Q_k is chosen to zero the last $m-(k+1)$ elements in the kth column of $A^{(k)}P_k$.

The process is continued until either the rows or columns are exhausted- When $m > n$ the process ends with the factorization

$$U^TAV = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad B = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix} \in \mathbf{R}^{(n+1)\times n}, \tag{8.4.31}$$

$$U = Q_1Q_2\cdots Q_n, \qquad V = P_1P_2\cdots P_{n-1}. \tag{8.4.32}$$

Note that since Q_k, only works on rows $k+1 : m$, and P_k, only works on columns $k : m$. It follows that

$$u_1 = e_1. \quad u_k = Ue_k = Q_1\cdots Q_ke_k, \quad k = 2 : n, \tag{8.4.33}$$

$$v_k = Ve_k = P_1\cdots P_ke_k, \quad k = 1 : n-1, \quad v_n = e_n. \tag{8.4.34}$$

If $m \le n$ then we obtain

$$U^T AV = (\,B \quad 0\,), \quad B = \begin{pmatrix} \alpha_1 & & & & \\ \beta_2 & \alpha_2 & & & \\ & \beta_3 & \ddots & & \\ & & \ddots & \alpha_{m-1} & \\ & & & \beta_m & \alpha_m \end{pmatrix} \in \mathbf{R}^{m \times m}.$$

$$U = Q_1 Q_2 \cdots Q_{m-2}, \qquad V = P_1 P_2 \cdots P_{m-1}.$$

The above process can always be carried through although some elements in B may vanish. Note that the singular values of B equal those of A; in particular, $\operatorname{rank}(A) = \operatorname{rank}(B)$. Using complex Householder transformations (see Sec. sec8.3.2) a complex matrix A can be reduced to *real* bidiagonal form by the algorithm above. In the nondegenerate case when all elements in B are nonzero, the bidiagonal decomposition is uniquely determined first column $u_1 := Ue_1$, which can be chosen arbitrarily.

The reduction to bidiagonal form is backward stable in the following sense. The computed $\bar{B}$ can be shown to be the exact result of an orthogonal transformation from left and right of a matrix $A + E$, where

$$\|E\|_F \le cn^2 u \|A\|_F, \tag{8.4.35}$$

and c is a constant of order unity. Moreover, if we use the information stored to generate the products $U = Q_1 \cdots Q_n$ and $V = P_1 \cdots P_{n-2}$ then the computed matrices are close to the exact matrices U and V which reduce $A + E$. This will guarantee that the singular values and transformed singular vectors of $\bar{B}$ are accurate approximations to those of a matrix close to A.

The bidiagonal reduction algorithm as described above requires approximately

$$4(mn^2 - n^3/3) \text{ flops}$$

when $m \ge n$, which is twice the work for a Householder QR factorization. The Householder vectors associated with U can be stored in the lower triangular part of A and those associated with V in the upper triangular part of A. Normally U and V are not explicitly required. They can be accumulated at a cost of $4(m^2n - mn^2 + \frac{1}{3}n^3)$ and $\frac{4}{3}n^3$ flops respectively.

When $m \gg n$ it is more efficient to use a two-step procedure as originally suggested by Lawson and Hanson [257] and later analyzed by T. Chan. In the first step the QR factorization of A is computed (possibly using column pivoting)

$$AP = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R \in \mathbf{R}^{n \times n},$$

which requires $4mn^2 - \frac{2}{3}n^3$ flops. In the second step the upper triangular matrix R is transformed to bidiagonal form using the algorithm described above. Note that no advantage can be taken of the triangular structure of R in the Householder algorithm. Already the first postmultiplication of R with P_1 will cause the lower triangular part of R to fill in. Hence, the Householder reduction of R to bidiagonal form will require $\frac{4}{3}n^3$ flops. The complete reduction to bidiagonal form then takes a total of

$$2(mn^2 + n^3) \text{ flops}.$$

This is less than the original Golub–Kahan algorithm when $m/n > 5/3$. Trefethen and Bau [360, pp. 237–238] have suggested a blend of the two above approaches that reduces the operation count slightly for $1 < m/n < 2$. They note that after k steps of the Golub–Kahan reduction the aspect ratio of the reduced matrix is $(m-k)/(n-k)$. and thus increases with k. To minimize the total operation count one should switch to the Chan algorithm when $(m-k)/(n-k) = 2$. This gives the operation count

$$4(mn^2 - n^3/3 - (m-n)^3/6) \text{ flops},$$

a modest approval over the two other methods when $n > m > 2n$.

We now derive an algorithm for solving the linear least squares problem $\min \|Ax - b\|_2$, where $A \in \mathbf{R}^{m\times n}$, $m \geq n$. Let Q_0 be a Householder reflection such that

$$Q_1 b = \beta_1 e_1. \tag{8.4.36}$$

Using the Golub–Kahan algorithm $Q_1 A$ to lower triangular form. we obtain

$$U^T \begin{pmatrix} b & AV \end{pmatrix} = \begin{pmatrix} \beta_1 e_1 & B_n \\ 0 & 0 \end{pmatrix}, \tag{8.4.37}$$

where e_1 is the first unit vector, and B_n is lower bidiagonal,

$$B = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_n & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix} \in \mathbf{R}^{(n+1)\times n}, \tag{8.4.38}$$

and

$$U = Q_1 Q_2 \cdots Q_{n+1}. \qquad V = P_1 P_2 \cdots P_{n-1}. \tag{8.4.39}$$

(Note the minor difference in notation in that Q_{k+1} now zeros elements in the kth column of A.)

Setting $x = Vy$ and using the invariance of the l_2-norm it follows that

$$\begin{aligned} \|b - Ax\|_2 &= \left\| \begin{pmatrix} b & A \end{pmatrix} \begin{pmatrix} -1 \\ x \end{pmatrix} \right\|_2 = \left\| U^T \begin{pmatrix} b & AV \end{pmatrix} \begin{pmatrix} -1 \\ y \end{pmatrix} \right\|_2 \\ &= \|\beta_1 e_1 - B_n y\|_2. \end{aligned}$$

Hence, if y solves the bidiagonal least squares problem

$$\min_y \|B_n y - \beta_1 e_1\|_2, \tag{8.4.40}$$

then $x = Vy$ minimizes $\|Ax - b\|_2$.

The least squares solution to (8.4.40) is obtained by a QR factorization of B_n, which takes the form

$$G_n(B_n \mid \beta_1 e_1) = \left(\begin{array}{c|c} R_n & f_k \\ & \bar{\phi}_{n+1} \end{array} \right) = \left(\begin{array}{ccccc|c} \rho_1 & \theta_2 & & & & \phi_1 \\ & \rho_2 & \theta_3 & & & \phi_2 \\ & & \rho_3 & \ddots & & \phi_3 \\ & & & \ddots & \theta_n & \vdots \\ & & & & \rho_n & \phi_n \\ \hline & & & & & \bar{\phi}_{n+1} \end{array} \right) \tag{8.4.41}$$

where G_n is a product of n Givens rotations. The solution is obtained by back-substitution from $R_n y = d_n$. The norm of the corresponding residual vector equals $|\bar{\phi}_{n+1}|$. To zero out the element β_2 we premultiply rows (1,2) with a rotation G_{12}, giving

$$\begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix} \left(\begin{array}{cc|c} \alpha_1 & 0 & \beta_1 \\ \beta_2 & \alpha_2 & 0 \end{array} \right) = \left(\begin{array}{cc|c} \rho_1 & \theta_2 & \phi_1 \\ 0 & \bar{\rho}_2 & \bar{\phi}_2 \end{array} \right).$$

(Here and in the following only elements affected by the rotation are shown.) Here the elements ρ_1, θ_2 and ϕ_1 in the first row are final, but $\bar{\rho}_2$ and $\bar{\phi}_2$ will be transformed into ρ_2 and ϕ_2 in the next step.

Continuing in this way in step j the rotation $G_{j,j+1}$ is used to zero the element β_{j+1}. In steps, $j = 2 : n-1$, the rows $(j, j+1)$ are transformed

$$\begin{pmatrix} c_j & s_j \\ -s_j & c_j \end{pmatrix} \left(\begin{array}{cc|c} \bar{\rho}_j & 0 & \bar{\phi}_j \\ \beta_{j+1} & \alpha_{j+1} & 0 \end{array} \right) = \left(\begin{array}{cc|c} \rho_j & \theta_{j+1} & \phi_j \\ 0 & \bar{\rho}_{j+1} & \bar{\phi}_{j+1} \end{array} \right).$$

where

$$\begin{aligned} &\phi_j = c_j \bar{\phi}_j, \qquad \bar{\phi}_{j+1} = -s_j \bar{\phi}_j, \qquad \rho_j = \sqrt{\bar{\rho}_j^2 + \beta_{j+1}^2}, \\ &\theta_{j+1} = s_j \alpha_{j+1}, \qquad \bar{\rho}_{n+1} = c_j \alpha_{j+1}. \end{aligned}$$

Note that by construction $|\bar{\phi}_{j+1}| \le \bar{\phi}_j$. Finally, in step n we obtain

$$\begin{pmatrix} c_n & s_n \\ -s_n & c_n \end{pmatrix} \left(\begin{array}{c|c} \bar{\rho}_n & \bar{\phi}_n \\ \beta_{n+1} & 0 \end{array} \right) = \left(\begin{array}{c|c} \rho_n & \phi_n \\ 0 & \bar{\phi}_{n+1} \end{array} \right).$$

After n steps, we have obtained the factorization (8.4.41) with

$$G_n = G_{n,n+1} \cdots G_{23} G_{12}.$$

Now consider the result after $k < n$ steps of the above bidiagonalization process have been carried out. At this point we have computed $Q_1, Q_2, \dots, Q_{k+1}$, $P_1, P_2, \dots, P_k$ such that the first k columns of A are in lower bidiagonal form, i.e.

$$Q_{k+1} \cdots Q_2 Q_1 \, A \, P_1 P_2 \cdots P_k \begin{pmatrix} I_k \\ 0 \end{pmatrix} = \begin{pmatrix} B_k \\ 0 \end{pmatrix} = \begin{pmatrix} I_{k+1} \\ 0 \end{pmatrix} B_k,$$

where $B_k \in \mathbf{R}^{(k+1)\times k}$ is a leading submatrix of B_n. Multiplying both sides with $Q_1Q_2\cdots Q_{k+1}$ and using orthogonality we obtain the relation

$$AV_k = U_{k+1}B_k = \hat{B}_k + \beta_{k+1}v_{k+1}e_k^T, \quad k = 1:n, \tag{8.4.42}$$

where

$$\begin{aligned} P_1P_2\cdots P_k \begin{pmatrix} I_k \\ 0 \end{pmatrix} &= V_k = (v_1,\dots,v_k), \\ Q_1Q_2\cdots Q_{k+1} \begin{pmatrix} I_{k+1} \\ 0 \end{pmatrix} &= U_{k+1} = (u_1,\dots,u_{k+1}). \end{aligned}$$

If we consider the intermediate result after applying also P_{k+1} the first $k+1$ rows have been transformed into bidiagonal form, i.e.

$$(\, I_{k+1} \quad 0\,)\, Q_{k+1}\cdots Q_2Q_1\, A\, P_1P_2\cdots P_{k+1} = (\, B_k \quad \alpha_{k+1}e_{k+1}\,)\,(\, I_{k+1} \quad 0\,)\,.$$

Transposing this gives a second relation

$$U_{k+1}^T A = B_k V_k^T + \alpha_{k+1}e_{k+1}v_{k+1}^T, \tag{8.4.43}$$

We now show that the bidiagonalization can be stopped prematurely if a zero element occurs in B. Assume first that the first zero element to occur is $\alpha_{k+1} = 0$. Then we have obtained the decomposition

$$\tilde{U}_{k+1}^T A \tilde{V}_k = \begin{pmatrix} B_k & 0 \\ 0 & A_k \end{pmatrix},$$

where $A_k \in \mathbf{R}^{(m-k-1)\times(n-k)}$, and

$$\tilde{U}_{k+1} = Q_{k+1}\cdots Q_2Q_1, \qquad \tilde{V}_k = P_1P_2\cdots P_k,$$

are square orthogonal matrices. Then, setting $x = \tilde{V}_k y$, the transformed least squares problem takes the form

$$\min_y \left\| \begin{pmatrix} B_k & 0 \\ 0 & A_k \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} \beta_1 e_1 \\ 0 \end{pmatrix} \right\|_2, \quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \tag{8.4.44}$$

$y_1 \in \mathbf{R}^k$, $y_2 \in \mathbf{R}^{n-k}$. This problem is separable and decomposes into two independent subproblems

$$\min_{y_1} \|B_k y_1 - \beta_1 e_1\|_2\,, \qquad \min_{y_2} \|A_k y_2\|_2. \tag{8.4.45}$$

By construction B_k has nonzero elements in its two diagonals. Thus, it has full column rank and the solution y_1 to the first subproblem is unique. Further, the minimum norm solution of the initial problem is obtained simply by taking $y_2 = 0$. We call the first subproblem (8.4.45) a **core subproblem**. It can be solved by QR factorization exactly as outlined for the full system when $k = n$.

When $\beta_{k+1} = 0$ is the first zero element to occur then the reduced problem has a similar separable form similar to (8.4.45). The core subproblem is now

$$\hat{B}_k y_1 = \beta_1 e_1, \quad \hat{B}_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{pmatrix} \in \mathbf{R}^{k\times k}. \tag{8.4.46}$$

Here $\hat{B}_k$ is square and lower triangular, and the solution y_1 is obtained by forward substitution. Taking $y_2 = 0$ the corresponding residual $b - AVy$ is zero and hence the original system $Ax = b$ is consistent.

We give two simple examples of when premature termination occurs. First assume that $b \perp \mathcal{R}(A)$. Then the reduction will terminate with $\alpha_1 = 0$. The core system is empty and $x = Vy_2 = 0$ is the minimal norm least squares solution.

If the bidiagonalization instead terminates with $\beta_2 = 0$, then the system $Ax = b$ is consistent and the minimum norm solution equals

$$x = (\beta_1/\alpha_1)v_1, \quad v_1 = V_1 e_1 = P_1 e_1 4.$$

Paige and Strakoš [302] have shown the following important properties of the core subproblem obtained by the bidiagonalization algorithm:

Theorem 8.4.2.

Assume that the bidiagonalization of $(\,b \quad A\,)$ *terminates prematurely with* $\alpha_k = 0$ *or* $\beta_{k+1} = 0$*. Then the core corresponding subproblem* (8.4.45) *or* (8.4.46) *is minimally dimensioned. Further, the singular values of the core matrix* B_k *or* $\hat{B}_k$*, are simple and the right hand side* βe_1 *has nonzero components in along each left singular vector.*

Proof. Sketch: The minimal dimension is a consequence of the uniqueness of the decomposition (8.4.37), as long as no zero element in B appears. That the matrix $\hat{B}_k$ has simple singular values follows from the fact that all subdiagonal elements are nonzero. The same is true for the square bidiagonal matrix $(B_k\ 0)$ and therefore also for B_k. Finally, if βe_1 did not have nonzero components along a left singular vector, then the reduction must have terminated earlier. For a complete proof we refer to [302].) $\square$

In many applications the numerical rank of the matrix A is much smaller than $\min\{m, n\}$. For example, in multiple linear regression often some columns are nearly linearly dependent. Then one wants to express the solution by restricting it to lie in a lower dimensional subspace. This can be achieved by neglecting small singular values of A and using a truncated SVD solution; see Sec. 8.4.2. In **partial least squares** (PLS) method this is achieved by a partial bidiagonalization of the matrix $(\,b \quad A\,)$. It is known that PLS often gives a faster reduction of the residual than TSVD.

We remark that the solution steps can be interleaved with the reduction to bidiagonal form. This makes it possible to compute a sequence of *approximate* *solutions* $x_k = P_1 P_2 \cdots P_k y_k$, where $y_k \in \mathbf{R}^k$ solves

$$\min_y \|\beta_1 e_1 - B_k y\|_2, \quad k = 1, 2, 3, \ldots. \tag{8.4.47}$$

After each (double) step in the bidiagonalization we advance the QR decomposition of B_k. The norm of the least squares residual corresponding to x_k is then given by

$$\|b - Ax_k\|_2 = |\bar{\phi}_{k+1}|.$$

The sequence of residual norms is nonincreasing. We stop and accept $x = V_k y_k$ as an approximate solution of the original least squares problem. if this residual is sufficiently small. This method is called the **Partial Least Squares** (PLS) method in statistics.

The sequential method outlined here is mathematically equivalent to a method called LSQR, which is a method of choice for solving *sparse* linear least squares. LSQR uses a Lanczos-type process for the bidiagonal reduction, which works only with the original sparse matrix. A number of important properties of the successive approximations x_k in PLS are best discussed in connection with LSQR; see Sec. 10.6.4.

Review Questions

4.1 When and why should column pivoting be used in computing the QR factorization of a matrix? What inequalities will be satisfied by the elements of R if the standard column pivoting strategy is used?

4.2 Show that the singular values and condition number of R equal those of A. Give a simple lower bound for the condition number of A in terms of its diagonal elements. Is it advisable to use this bound when no column pivoting has been performed?

4.3 Give a simple lower bound for the condition number of A in terms of the diagonal elements of R. Is it advisable to use this bound when no column pivoting has been performed?

4.4 What is meant by a Rank-revealing QR factorization? Does such a factorization always exist?

4.5 How is the *numerical* rank of a matrix A defined? Give an example where the numerical rank is not well determined.

Problems

4.1 (a) Describe how the QR factorizations of a matrix of the form

$$\begin{pmatrix} A \\ \mu D \end{pmatrix}, \quad A \in \mathbf{R}^{m \times n},$$

where $D \in \mathbf{R}^{n\times n}$ is diagonal, can be computed using Householder transformations in mn^2 flops.

(b) Estimate the number of flops that are needed for the reduction using Householder transformations in the special case that $A = R$ is upper triangular? Devise a method using Givens rotations for this special case!

Hint: In the Givens method zero one diagonal at a time in R working from the main diagonal inwards.

4.2 Let the vector v, $\|v\|_2 = 1$, satisfy $\|Av\|_2 = \epsilon$, and let Π be a permutation such that

$$|w_n| = \|w\|_\infty, \qquad \Pi^T v = w.$$

(a) Show that if R is the R factor of $A\Pi$, then $|r_{nn}| \le n^{1/2}\epsilon$.

Hint: Show that $\epsilon = \|Rw\|_2 \ge |r_{nn}w_n|$ and then use the inequality $|w_n| = \|w\|_\infty \ge n^{-1/2}\|w\|_2$.

(b) Show using (a) that if $v = v_n$, the right singular vector corresponding to the smallest singular value $\sigma_n(A)$, then

$$\sigma_n(A) \ge n^{-1/2}|r_{nn}|.$$

4.4 Consider a nonsingular 2×2 upper triangular matrix and its inverse

$$R = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}, \qquad R^{-1} = \begin{pmatrix} a^{-1} & a^{-1}bc^{-1} \\ 0 & c^{-1} \end{pmatrix}.$$

(a) Suppose we want to choose Π to *maximize* the $(1,1)$ element in the QR factorization of $R\Pi$. Show that this is achieved by taking $\Pi = I$ if $|a| \ge \sqrt{b^2+c^2}$, else $\Pi = \Pi_{12}$, where Π_{12} interchanges columns 1 and 2.

(b) Unless $b = 0$ the permutation chosen in (a) may not *minimize* the (2,2) element in the QR factorization of $R\Pi$. Show that this is achieved by taking $\Pi = I$ if $|c^{-1}| \ge \sqrt{a^{-2} + b^2(ac)^{-2}}$ else $\Pi = \Pi_{12}$. Hence, the test compares *row* norms in R^{-1} instead of *column* norms in R.

4.6 To minimize $\|x\|_2$ is not always a good way to resolve rank deficiency, and therefore the following generalization of problem (8.4.18) is often useful: For a given matrix $B \in \mathbf{R}^{p\times n}$ consider the problem

$$\min_{x\in S} \|Bx\|_2, \qquad S = \{x \in \mathbf{R}^n \mid \|Ax - b\|_2 = \min\}.$$

(a) Show that this problem is equivalent to

$$\min_{x_2} \|(BC)x_2 - (Bd)\|_2,$$

where C and d are defined by (8.4.17).

(b) Often one wants to choose B so that $\|Bx\|_2$ is a measure of the smoothness of the solution x. For example one can take B to be a discrete approximation to the second derivative operator,

$$B = \begin{pmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{pmatrix} \in \mathbf{R}^{(n-2)\times n}.$$

Show that provided that $\mathcal{N}(A)\cap\mathcal{N}(B) = \emptyset$ this problem has a unique solution, and give a basis for $\mathcal{N}(B)$.

4.5 Let $A \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(A) = r$. A rank revealing LU factorizations of the form

$$\Pi_1 A \Pi_2 = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \end{pmatrix},$$

where Π_1 and Π_2 are permutation matrices and $L_{11}, U_{11} \in \mathbf{R}^{r\times r}$ are triangular and nonsingular can also be used to compute pseudo-inverse solutions $x = A^\dagger b$. Show, using Theorem 8.1.10 that

$$A^\dagger = \Pi_2 \begin{pmatrix} I_r & S \end{pmatrix}^\dagger U_{11}^{-1} L_{11}^{-1} \begin{pmatrix} I_r \\ T \end{pmatrix}^\dagger \Pi_1,$$

where $T = L_{21}L_{11}^{-1}$, $S = U_{11}^{-1}U_{12}$. (Note that S is empty if $r = n$, and T empty if $r = m$.)

4.6 Consider the block upper-bidiagonal matrix

$$A = \begin{pmatrix} B_1 & C_1 & \\ & B_2 & C_2 \\ & & B_3 \end{pmatrix}$$

Outline an algorithm for computing the QR factorization of A, which treats one block row at a time. (It can be assumed that A has full column rank.) Generalize the algorithm to an arbitrary number of block rows!

4.7 (a) Suppose that we have computed the pivoted QR factorization of A,

$$Q^T A \Pi = \begin{pmatrix} R \\ 0 \end{pmatrix} \in \mathbf{R}^{m\times n},$$

of a matrix $A \in \mathbf{R}^{m\times n}$. Show that by postmultiplying the *upper* triangular matrix R by a sequence of Householder transformations we can transform R into a *lower* triangular matrix $L = RP \in \mathbf{R}^{n\times n}$ and that by combining these two factorizations we obtain

$$Q^T A \Pi P = \begin{pmatrix} L \\ 0 \end{pmatrix}. \tag{8.4.48}$$

Comment: This factorization, introduced by G. W. Stewart, is equivalent to one step of the unshifted QR–SVD algorithm; see Sec. 9.5.3.

(b) Show that the total cost for computing the QLP decomposition is roughly

$2mn^2 + 2n^3/3$ flops. How does that compare with the cost for computing the bidiagonal decomposition of A?

(c) Show that the two factorizations can be interleaved. What is the cost for performing the first k steps?

4.8 Work out the details of an algorithm for transforming a matrix $A \in \mathbf{R}^{m\times n}$ to *lower* bidiagonal form. Consider both cases when $m > n$ and $m \le n$.

Hint: It can be derived by applying the algorithm for transformation to upper bidiagonal form to A^T.

8.5 Some Structured Least Squares Problems

8.5.1 Blocked Form of QR Factorization

In many least squares problems the unknowns x can be naturally partitioned into two groups with n_1 and n_2 components, respectively. Then the problem has the form

$$\min_{x_1,x_2} \left\| \begin{pmatrix} A_1 & A_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - b \right\|_2, \tag{8.5.1}$$

where $A = \begin{pmatrix} A_1 & A_2 \end{pmatrix} \in \mathbf{R}^{m\times n}$ and $n = n_1 + n_2$. For example, in separable nonlinear least squares problems subproblems of the form (8.5.1) arise, see Sec. 11.2.5.

Assume that the matrix A has full column rank and let $P_{\mathcal{R}(A_1)}$ be the orthogonal projection onto $\mathcal{R}(A_1)$. For any x_2 we can split the vector $b - A_2x_2 = r_1 + r_2$ into two orthogonal components

$$r_1 = P_{\mathcal{R}(A_1)}(b - A_2x_2), \quad r_2 = (I - P_{\mathcal{R}(A_1)})(b - A_2x_2).$$

Then the problem (8.5.1) takes the form

$$\min_{x_1,x_2} \left\| (A_1x_1 - r_1) - P_{\mathcal{N}(A_1^T)}(b - A_2x_2) \right\|_2. \tag{8.5.2}$$

Here, since $r_1 \in \mathcal{R}(A_1)$ the variables x_1 can always be chosen so that $A_1x_1 - r_1 = 0$. It follows that in the least squares solution to (8.5.1) x_2 is the solution to the reduced least squares problem

$$\min_{x_2} \|P_{\mathcal{N}(A_1^T)}(A_2x_2 - b)\|_2. \tag{8.5.3}$$

where the variables x_1 have been eliminated. When this reduced problem has been solved for x_2 then x_1 can be computed from the least squares problem

$$\min_{x_1} \|A_1x_1 - (b - A_2x_2)\|_2. \tag{8.5.4}$$

Sometimes it may be advantageous to carry out a *partial* QR factorization, where only the $k = n_1 < n$ columns of A are reduced. Using Householder QR

$$Q_1^T(A\ b) = \begin{pmatrix} R_{11} & \tilde{A}_{12} & \tilde{b}_1 \\ 0 & \tilde{A}_{22} & \tilde{b}_2 \end{pmatrix},$$

with R_{11} nonsingular. Then x_2 is the solution to the reduced least squares problem

$$\min_{x_2} \|\tilde{b}_2 - \tilde{A}_{22)} x_2\|_2,$$

If this is again solved by Householder QR nothing new comes out—we have only emphasized a new aspect of this algorithm. However, it may be advantageous to use another method for the reduced problem.

In some applications, e.g., when A has block angular structure (see Sec. 8.5.2), it may be preferable instead not to save R_{11} and R_{12} and instead to refactorize A_1 and solve (8.5.4) for x_1.

If we use perform n_1 steps of MGS on the full problem, this yields the partial factorization

$$(A\ b) = (\,Q_1 \quad \tilde{A}_2 \quad \tilde{b}\,) \begin{pmatrix} R_{11} & R_{12} & z_1 \\ 0 & I & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

where R_{11} is nonsingular. The residual is decomposed as as $r = r_1 + r_2$, $r_1 \perp r_2$, where

$$r_1 = Q_1(z_1 - R_{12}x_2 - R_{11}x_1), \qquad r_2 = \tilde{b}_1 - \tilde{A}_2 x_2.$$

Then x_2 is the solution to the reduced least squares problem $\min_{x_2} \|\tilde{b} - \tilde{A}x_2\|_2$. With x_2 known x_1 can been computed by back-substitution from $R_{11}x_1 = z_k - R_{12}x_2$.

To obtain near-peak performance for large dense matrix computations on current computing architectures requires code that is dominated by matrix-matrix operations since these involve less data movement per floating point computation. The QR factorization should therefore be organized in partitioned or blocked form, where the operations have been reordered and grouped into matrix operations.

For the QR factorization $A \in \mathbf{R}^{m\times n}$ $(m \geq n)$ is partitioned as

$$A = (A_1,\ A_2), \qquad A_1 \in \mathbf{R}^{m\times nb}, \tag{8.5.5}$$

where nb is a suitable block size and the QR factorization

$$Q_1^T A_1 = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \qquad Q_1 = P_1 P_2 \cdots P_{nb}, \tag{8.5.6}$$

is computed, where $P_i = I - u_i u_i^T$ are Householder reflections. Then the remaining columns A_2 are are updated

$$Q_1^T A_2 = Q_1^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} = \begin{pmatrix} R_{12} \\ \tilde{A}_{22} \end{pmatrix}. \tag{8.5.7}$$

In the next step we partition $\tilde{A}_{22} = (B_1, B_2)$, and compute the QR factorization of $B_1 \in \mathbf{R}^{(m-r)\times r}$. Then B_2 is updated as above, and we continue in this way until the columns in A are exhausted.

A major part of the computation in spent in the updating step (8.5.7). As written this step cannot use BLAS-3, which slows down the execution. To achieve better performance it is essential that this part is sped up. The solution is to aggregate the Householder transformations so that their application can be expressed as matrix operations. For use in the next subsection, we give a slightly more general result.

Lemma 8.5.1.

Let $P_1, P_2, \ldots, P_r$ be a sequence of Householder transformations. Set $r = r_1 + r_2$, and assume that

$$Q_1 = P_1 \cdots P_{r_1} = I - Y_1 T_1 Y_1^T, \qquad Q_2 = P_{r_1+1} \cdots P_r = I - Y_2 T_2 Y_2^T,$$

where $T_1, T_2 \in \mathbf{R}^{r\times r}$ are upper triangular matrices. Then for the product $Q_1 Q_2$ we have

$$Q = Q_1 Q_2 = (I - Y_1 T_1 Y_1^T)(I - Y_2 T_2 Y_2^T) = (I - YTY^T) \tag{8.5.8}$$

where

$$\hat{Y} = (Y_1,\ Y_2), \qquad \hat{T} = \begin{pmatrix} T_1 & -(T_1 Y_1^T)(Y_2 T_2) \\ 0 & T_2 \end{pmatrix}. \tag{8.5.9}$$

Note that Y is formed by concatenation, but computing the off-diagonal block in T requires extra operations.

For the partitioned algorithm we use the special case when $r_2 = 1$ to aggregate the Householder transformations for each processed block. Starting with $Q_1 = I - \tau_1 u_1 u_1^T$, we set $Y = u_1$, $T = \tau_1$ and update

$$Y := (Y,\ u_{k+1}), \qquad T := \begin{pmatrix} T & -\tau_k T Y^T u_k \\ 0 & \tau_k \end{pmatrix}, \quad k = 2 : nb. \tag{8.5.10}$$

Note that Y will have a trapezoidal form and thus the matrices Y and R can overwrite the matrix A. With the representation $Q = (I - YTY^T)$ the updating of A_2 becomes

$$B = Q_1^T A = (I - YT^T Y^T)A_2 = A_2 - YT^T Y^T A_2,$$

which now involves only matrix operations.

This partitioned algorithm requires more storage and operations than the point algorithm, namely those needed to produce and store the T matrices. However, for large matrices this is more than offset by the increased rate of execution.

As mentioned in Chapter 7 recursive algorithms can be developed into highly efficient algorithms for high performance computers and are an alternative to the currently more used partitioned algorithms by LAPACK. The reason for this is that recursion leads to automatic variable blocking that dynamically adjusts to an arbitrary number of levels of memory hierarchy.

Consider the partitioned QR factorization

$$A = (\, A_1 \quad A_2 \,) = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

where Let A_1 consist of the first $\lfloor n/2 \rfloor$ columns of A. To develop a recursive algorithm we start with a QR factorization of A_1 and update the remaining part A_2 of the matrix,

$$Q_1^T A_1 = \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}, \qquad Q_1^T A_2 = Q_1^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} = \begin{pmatrix} R_{12} \\ \tilde{A}_{22} \end{pmatrix}.$$

Next $\tilde{A}_{22}$ is recursively QR decomposed giving Q_2, R_{22}, and $Q = Q_1 Q_2$.

As an illustration we give below a simple implementation in Matlab, which is convenient to use since it allows for the definition of recursive functions.

```
function [Y,T,R] = recqr(A)
%
% RECQR computes the QR factorization of the m by n matrix A,
% (m >= n). Output is the n by n triangular factor R, and
% Q = (I - YTY')  represented in aggregated form, where Y is
% m by n and unit lower trapezoidal, and T is n by n upper
% triangular
    [m,n] = size(A);
    if n == 1
    [Y,T,R] = house(A);
    else
      n1 = floor(n/2);
      n2 = n - n1; j = n1+1;
      [Y1,T1,R1]= recqr(A(1:m,1:n1));
      B = A(1:m,j:n) - (Y1*T1')*(Y1'*A(1:m,j:n));
      [Y2,T2,R2] = recqr(B(j:m,1:n2));
      R = [R1, B(1:n1,1:n2); zeros(n-n1,n1), R2];
      Y2 = [zeros(n1,n2); Y2];
      Y = [Y1, Y2];
      T = [T1, -T1*(Y1'*Y2)*T2; zeros(n2,n1), T2];
    end
%
```

The algorithm uses the function house(a) to compute a Householder transformation $P = I - +\tau\, uu^T$, such that $Pa = \sigma\, e1$, $\sigma = -\text{sign}\,(a_1)\|a\|_2$. A serious defect of this algorithm is the overhead in storage and operations caused by the T matrices. In the partitioned algorithm n/nb T-matrices of size we formed and stored, giving a storage overhead of $\frac{1}{2}n \cdot nb$. In the recursive QR algorithm in the end a T-matrix of size $n \times n$ is formed and stored, leading to a much too large storage and operation overhead. Therefore, a better solution is to use a hybrid between the partitioned and the recursive algorithm, where the recursive QR algorithm is used to factorize the blocks in the partitioned algorithm.

8.5.2 Block Angular Least Squares Problems

There is often a substantial similarity in the structure of many large scale sparse least squares problems. In particular, the problem can often be put in the following bordered block diagonal or **block angular form**:

$$A = \left(\begin{array}{cccc|c} A_1 & & & & B_1 \\ & A_2 & & & B_2 \\ & & \ddots & & \vdots \\ & & & A_M & B_M \end{array}\right), \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \\ x_{M+1} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix}, \tag{8.5.11}$$

where

$$A_i \in \mathbf{R}^{m_i \times n_i}, \quad B_i \in \mathbf{R}^{m_i \times n_{M+1}}, \quad i = 1 : M,$$

and

$$m = m_1 + m_2 + \cdots + m_M, \qquad n = n_1 + n_2 + \cdots + n_{M+1}.$$

Note that the variables $x_1, \ldots, x_M$ are coupled only to the variables x_{M+1}, which reflects a "local connection" structure in the underlying physical problem. Applications where the form (8.5.11) arises naturally include photogrammetry, Doppler radar and GPS positioning, and geodetic survey problems. The block matrices A_i, B_i $i = 1 : M$, may also have some structure that can be taken advantage of, but in the following we ignore this.

The normal matrix of A in (8.5.11) has a doubly bordered block diagonal form,

$$A^T A = \left(\begin{array}{cccc|c} A_1^T A_1 & & & & A_1^T B_1 \\ & A_2^T A_2 & & & A_2^T B_2 \\ & & \ddots & & \vdots \\ & & & A_M^T A_M & A_M^T B_M \\ \hline B_1^T A_1 & B_2^T A_2 & \cdots & B_M^T A_M & C \end{array}\right),$$

where

$$C = \sum_{k=1}^{M} B_k^T B_k.$$

We assume in the following that $\operatorname{rank}(A) = n$, which implies that the matrices $A_i^T A_i$, $i = 1 : M$, and C are positive definite. It is easily seen that then the Cholesky factor R of $A^T A$ will have a block structure similar to that of A,

$$R = \left(\begin{array}{cccc|c} R_1 & & & & S_1 \\ & R_2 & & & S_2 \\ & & \ddots & & \vdots \\ & & & R_M & S_M \\ \hline & & & & R_{M+1}. \end{array}\right), \tag{8.5.12}$$

Here $R_i \in \mathbf{R}^{n_i \times n_i}$, $i = 1 : M$, are the Cholesky factors of $A_i^T A_i$, R_{M+1} the Cholesky factor of C, and

$$S_i = (A_i R_i^{-1})^T B_i, \quad i = 1 : M + 1.$$

Clearly the blocks in (8.5.12) can also be computed by QR factorization. We now outline an algorithm for solving least squares problems of block angular form based on QR factorization. It proceeds in three steps:

1. For $i = 1 : M$ reduce the diagonal block A_i to upper triangular form by a sequence of orthogonal transformations applied to (A_i, B_i) and the right-hand side b_i, yielding

$$Q_i^T (A_i, B_i) = \begin{pmatrix} R_i & S_i \\ 0 & T_i \end{pmatrix}, \qquad Q_i^T b_i = \begin{pmatrix} c_i \\ d_i \end{pmatrix}.$$

It is usually advantageous to continue the reduction in step 1 so that the matrices T_i, $i = 1 : M$, are brought into upper trapezoidal form.

2. Set

$$T = \begin{pmatrix} T_1 \\ \vdots \\ T_M \end{pmatrix}, \qquad d = \begin{pmatrix} d_1 \\ \vdots \\ d_M \end{pmatrix}$$

and compute the QR decomposition

$$\tilde{Q}_{M+1}^T \,(T \quad d) = \begin{pmatrix} R_{M+1} & c_{M+1} \\ 0 & d_{M+1} \end{pmatrix}.$$

The solution to $\min_{x_{M+1}} \|Tx_{M+1} - d\|_2$ is then obtained from the triangular system

$$R_{M+1} x_{M+1} = c_{M+1},$$

and the residual norm is given by $\rho = \|d_{M+1}\|_2$.

3. For $i = M : (-1) : 1$ compute $x_M, \dots, x_1$ by back-substitution in the triangular systems

$$R_i x_i = c_i - S_i x_{M+1}.$$

In steps 1 and 3 the computations can be performed in parallel on the M subsystems. There are alternative ways to organize this algorithm. Note that when x_{M+1} has been computed in step 2, then the vectors x_i, $i = 1, \dots, M$, solves the least squares problem

$$\min_{x_i} \|A_i x_i - g_i\|_2, \quad g_i = b_i - B_i x_{M+1}.$$

Hence, it is possible to discard the R_i, S_i and c_i in step 1 if the QR factorizations of A_i are recomputed in step 3. In some practical problems this modification can reduce the storage requirement by an order of magnitude, while the recomputation of R_i may only increase the operation count by a few percent.

Using the structure of the R-factor in (8.5.12), the diagonal blocks of the variance-covariance matrix $C = (R^T R)^{-1} = R^{-1} R^{-T}$ can be written

$$\begin{aligned} &C_{M+1,M+1} = R_{M+1}^{-1} R_{M+1}^{-T}, \\ &C_{i,i} = R_i^{-1}(I + W_i^T W_i) R_i^{-T}, \quad W_i^T = S_i R_{M+1}^{-1}, \quad i = 1, \dots, M. \end{aligned} \tag{8.5.13}$$

If we compute the QR decompositions

$$Q_i \begin{pmatrix} W_i \\ I \end{pmatrix} = \begin{pmatrix} U_i \\ 0 \end{pmatrix}, \quad i = 1, \dots, M,$$

we have $I + W_i^T W_i = U_i^T U_i$ and then

$$C_{i,i} = (U_i R_i^{-T})^T (U_i R_i^{-T}), \quad i = 1, \dots, M.$$

This assumes that all the matrices R_i and S_i have been retained.

8.5.3 Banded Least Squares Problems

We now consider orthogonalization methods for the special case when A is a banded matrix of row bandwidth w, see Definition 8.2.1. From Theorem 8.2.2 we know that A^TA will be a symmetric band matrix with only the first $r = w - 1$ super-diagonals nonzero. Since the factor R in the QR factorization equals the unique Cholesky factor of A^TA it will have only w nonzeros in each row.

Even though the *final* factor R is independent of the row ordering in A, the intermediate fill-in will vary. For banded rectangular matrices the QR factorization can be obtained efficiently by sorting the rows of A and suitably subdividing the Householder transformations. The rows of A should be sorted by leading entry order (i.e., increasing minimum column subscript order) That is, if $f_i, i = 1 : m$ denotes the column indices of the first nonzero element in row i we should have,

$$i \le k \Rightarrow f_i \le f_k.$$

Such a band matrix can then be written as

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_q \end{pmatrix}, \quad q \le n,$$

is said to be in **standard form**. where in block A_i the first nonzero element of each row is in column i. The Householder QR process is then applied to the matrix in q major steps. In the first step a QR decomposition of the first block A_1 is computed, yielding R_1. Next at step k, $k = 2 : q$, R_{k-1} will be merged with A_k yielding

$$Q_k^T \begin{pmatrix} R_{k-1} \\ A_k \end{pmatrix} = R_k.$$

Since the rows of block A_k has their first nonzero elements in column k, the first $k-1$ rows of R_{k-1} will not be affected. The matrix Q can be implicitly represented in terms of the Householder vectors of the factorization of the subblocks. This sequential Householder algorithm, which is also described in [257, Ch. 27], requires $(m + 3n/2)w(w + 1)$ multiplications or about twice the work of the less stable Cholesky approach. For a detailed description of this algorithm, see Lawson and Hanson [257, Ch. 11].

Example 8.5.1.

In a frequently occurring special case the matrix is originally given in the form

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix},$$

where both A_1 and A_2 are banded. If the QR factorizations of A_1 and A_2 are first computed separately, the problem is reduced computing the QR factorization of

$$\begin{pmatrix} R_1 \\ R_2 \end{pmatrix},$$

with R_1 and R_2 banded with bandwidth w_1 and w_2, respectively. Then either of the two row ordering algorithms will interleave the rows of R_1 and R_2.

It can be shown that a row-wise Givens algorithm requires about $2n(w_1^2 + w_2^2)$ multiplications and creates no unnecessary fill-in. Below we s consider the case $w_1 = 3$, $w_2 = 1$, and $n = 6$. The result after 3 steps without reordering is

$$\begin{pmatrix} \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \\ \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{pmatrix} \Rightarrow \begin{pmatrix} \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \\ \otimes & \oplus & \oplus & + & + & \\ & \otimes & \oplus & + & + & \\ & & \otimes & + & + & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{pmatrix}.$$

It can be deduced that $1 + 2 + \cdots + n = \frac{1}{2}n(n+1)$ rotations are needed for the reduction, each using $4w$ multiplications, i.e., a total of about $2n(n+1)w_1$. Now consider the same step with the optimal row ordering ordering:

$$\begin{pmatrix} \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ \times & \times & \times & & & \\ & & & \times & & \\ & \times & \times & \times & & \\ & & & & \times & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \\ & & & & & \times \end{pmatrix} \Rightarrow \begin{pmatrix} \times & + & + & & & \\ & \times & + & + & & \\ & & \times & + & + & \\ \otimes & \otimes & \otimes & & & \\ & & & \times & & \\ & \otimes & \otimes & \times & & \\ & & & & \times & \\ & & \otimes & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \\ & & & & & \times \end{pmatrix}.$$

Now the reduction takes a total of about $2nw(w_1+1)$ multiplications and no nonzero elements are created outside the final matrix R.

In Sec. 4.6.4 we considered the interpolation of a function f where with a linear combination of $m + k$ B-splines of degree k, see (4.6.18), on $\Delta = \{x_0 < x_1 < \cdots < x_m\}$. Assume that we are given function values $f_j = f(\tau_j)$, where $\tau_1 < \tau_2 < \ldots < \tau_n$ are distinct points and $n \geq m + k$. Then we consider the least squares approximation problem

$$\min \sum_{j=1}^{n} e_j^2, \qquad e_j = w_i \left(f_j - \sum_{i=-k}^{m-1} c_i B_{i,k+1}(\tau_j) \right). \tag{8.5.14}$$

where w_j are positive weights. This is an overdetermined linear system for c_i, $i = -k, \ldots, m-1$. The elements of its coefficient matrix $B_{i,k+1}(\tau_j)$ can be evaluated by the recurrence (4.6.19). The coefficient matrix has a band structure since in the jth row the ith element will be zero if $\tau_j \notin [x_i, x_{i+k+1}]$. It can be shown, see de Boor [1978, p. 200], that the coefficient matrix will have full rank equal to $m+k$ if and only if there is a subset of points τ_j satisfying

$$x_{j-k-1} < \tau_j < x_j, \qquad \forall j = 1 : m+k. \tag{8.5.15}$$

Example 8.5.2.

The least squares approximation of a discrete set of data by a linear combination of cubic B-splines gives rise to a banded linear least squares problem. Let

$$s(t) = \sum_{j=1}^{n} x_j B_j(t),$$

where $B_j(t)$, $j = 1 : n$ are the normalized cubic B-splines, and let (y_i, t_i), $i = 1 : m$ be given data points. If we determine x to minimize

$$\sum_{i=1}^{m} (s(t_i) - y_i)^2 = \|Ax - y\|_2^2,$$

then A will be a banded matrix with $w = 4$. In particular, if $m = 13$, $n = 8$ the matrix may have the form shown in Figure 8.5.1. Here A consists of blocks A_k^T, $k = 1 : 7$. In the Figure 8.5.1 we also show the matrix after the first three blocks have been reduced by Householder transformations $P_1, \ldots, P_9$. Elements which have been zeroed by P_j are denoted by j and fill-in elements by $+$. In step $k = 4$ only the indicated part of the matrix is involved.

$$\begin{pmatrix}
\times & \times & \times & \times & & & & \\
1 & \times & \times & \times & + & & & \\
1 & 2 & \times & \times & + & + & & \\
 & 3 & 4 & \times & \times & + & & \\
 & 3 & 4 & 5 & \times & + & & \\
 & & 6 & 7 & 8 & \times & & \\
 & & 6 & 7 & 8 & 9 & & \\
 & & 6 & 7 & 8 & 9 & & \\
 & & & \times & \times & \times & \times & \\
 & & & \times & \times & \times & \times & \\
 & & & & \times & \times & \times & \times \\
 & & & & \times & \times & \times & \times \\
 & & & & \times & \times & \times & \times
\end{pmatrix}$$

Figure 8.5.1. *The QR factorization of a banded rectangular matrix A.*

In the algorithm the Householder transformations can also be applied to one or several right hand sides b to produce

$$c = Q^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad c_1 \in \mathbf{R}^n.$$

The least squares solution is then obtained from $Rx = c_1$ by back-substitution. The vector c_2 is not stored but used to accumulate the residual sum of squares $\|r\|_2^2 = \|c_2\|_2^2$.

It is also possible to perform the QR factorization by treating one row at a time using Givens' rotations. Each step then is equivalent to updating a full triangular matrix formed by columns $f_i(A)$ to $l_i(A)$. Further, if the matrix A is in standard form the first $f_i(A)$ rows of R are already finished at this stage. The reader is encouraged to work through Example 8.5.2 below in order to understand how the algorithm proceeds!

Example 8.5.3.

Often one needs to represent an orthogonal rotation $Q \in \mathbf{R}^{3\times 3}$, $\det(Q) = 1$ as three successive plane rotations or by the angles of these rotations. The classical choice corresponds to a product of three Givens rotations

$$G_{23}(\phi)G_{12}(\theta)G_{23}(\psi)Q = I.$$

The three angles ϕ, θ, and ψ are called the **Euler angles**.

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_3 & s_3 \\ 0 & -s_3 & c_3 \end{pmatrix}\begin{pmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_1 & s_1 \\ 0 & -s_1 & c_1 \end{pmatrix}\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Here the first Givens rotation $G_{23}(\psi)$ is used to zero the element a_{31}. Next $G_{12}(\theta)$ is used to zero the element a_{21}. Finally, $G_{23}(\phi)$ is used to zero the element a_{32}. The final product is orthogonal and lower triangular and thus must be equal the unit matrix.

A problem with this representation is that the Euler angles may not depend continuously on the data. If Q equals the unit matrix plus small terms then a small perturbation may change an angle as much as 2π.

A different set of angles based on zeroing the elements in the order a_{21}, a_{31}, a_{32} is to be preferred. This corresponds to a product of three Givens rotations

$$G_{23}(\phi)G_{13}(\theta)G_{12}(\psi)Q = I.$$

This yields a continuous representation of Q.

8.5.4 Block Triangular Form

An arbitrary rectangular matrix $A \in \mathbf{R}^{m\times n}$, $m \geq n$, can by row and column permutations be brought into the block triangular form

$$PAQ = \begin{pmatrix} A_h & U_{hs} & U_{hv} \\ & A_s & U_{sv} \\ & & A_v \end{pmatrix}, \tag{8.5.16}$$

where the diagonal block A_h is underdetermined (i.e., has more columns than rows), A_s is square and A_v is overdetermined (has more rows than columns), and all three blocks have a nonzero diagonal; see the example in Figure 8.5.2. The submatrices A_v and A_h^T both have the strong Hall property. The off-diagonal blocks denoted by U are possibly nonzero matrices of appropriate dimensions. This block triangular form (8.5.16) of a sparse matrix is based on a canonical decomposition of bipartite graphs.

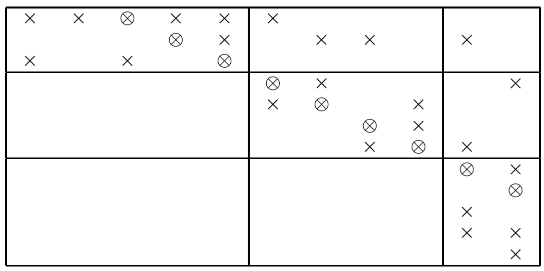

Figure 8.5.2. *The coarse block triangular decomposition of A.*

We call the decomposition of A into the submatrices A_h, A_s, and A_v the **coarse decomposition**. One or two of the diagonal blocks may be absent in the coarse decomposition. It may be possible to further decompose the diagonal blocks in (8.5.16) to obtain the **fine decompositions** of these submatrices. Each of the blocks A_h and A_v may be further decomposable into block diagonal form,

$$A_h = \begin{pmatrix} A_{h1} & & \\ & \ddots & \\ & & A_{hp} \end{pmatrix}, \quad A_v = \begin{pmatrix} A_{v1} & & \\ & \ddots & \\ & & A_{vq} \end{pmatrix},$$

where each $A_{h1}, \dots, A_{hp}$ is underdetermined and each $A_{v1}, \dots, A_{vq}$ is overdetermined. The submatrix A_s may be decomposable in block upper triangular form

$$A_s = \begin{pmatrix} A_{s1} & U_{12} & \dots & U_{1,t} \\ & A_{s2} & \dots & U_{2,t} \\ & & \ddots & \vdots \\ & & & A_{st} \end{pmatrix} \tag{8.5.17}$$

with square diagonal blocks $A_{s1}, \dots, A_{st}$ which have nonzero diagonal elements. The resulting decomposition can be shown to be essentially unique. Any one block triangular form can be obtained from any other by applying row permutations that involve the rows of a single block row, column permutations that involve the columns of a single block column, and symmetric permutations that reorder the blocks.

An algorithm for the more general block triangular form described above due to Pothen and Fan depends on the concept of matchings in bipartite graphs. The algorithm consists of the following steps:

1. Find a maximum matching in the bipartite graph $G(A)$ with row set R and column set C.
2. According to the matching, partition R into the sets VR, SR, HR and C into the sets VC, SC, HC corresponding to the horizontal, square, and vertical blocks.
3. Find the diagonal blocks of the submatrix A_v and A_h from the connected components in the subgraphs $G(A_v)$ and $G(A_h)$. Find the block upper triangular form of the submatrix A_s from the strongly connected components in the associated directed subgraph $G(A_s)$, with edges directed from columns to rows.

The reordering to block triangular form in a preprocessing phase can save work and intermediate storage in solving least squares problems. If A has structural rank equal to n, then the first block row in (8.5.16) must be empty, and the original least squares problem can after reordering be solved by a form of block back-substitution. First compute the solution of

$$\min_{\tilde{x}_v} \|A_v \tilde{x}_v - \tilde{b}_v\|_2, \tag{8.5.18}$$

where $\tilde{x} = Q^T x$ and $\tilde{b} = Pb$ have been partitioned conformally with PAQ in (8.5.16). The remaining part of the solution $\tilde{x}_k, \dots, \tilde{x}_1$ is then determined by

$$A_{si}\tilde{x}_i = \tilde{b}_i - \sum_{j=i+1}^{k} U_{ij}\tilde{x}_j, \quad i = k, \dots, 2, 1. \tag{8.5.19}$$

Finally, we have $x = Q\tilde{x}$. We can solve the subproblems in (8.5.18) and (8.5.19) by computing the QR decompositions of A_v and $A_{s,i}$, $i = 1, \dots, k$. Since $A_{s1}, \dots, A_{sk}$ and A_v have the strong Hall property the structures of the matrices R_i are correctly predicted by the structures of the corresponding normal matrices.

If the matrix A has structural rank less than n, then we have an underdetermined block A_h. In this case we can still obtain the form (8.5.17) with a square block A_{11} by permuting the extra columns in the first block to the end. The least squares solution is then not unique, but a unique solution of minimum length can be found as outlined in Section 2.7.

8.5.5 General Sparse QR Factorization

An algorithm using the normal equations for solving sparse linear least squares problems is often split up in a symbolical and a numerical phase as follows. (We assume that $\operatorname{rank}(A) = n$; for modifications needed to treat the case when $\operatorname{rank}(A) < n$, see Section 6.7.1.)

Algorithm 8.9. *Sparse Normal Equations.*

1. Determine symbolically the structure of A^TA.

2. Determine a column permutation P_c such that $P_c^T A^T A P_c$ has a sparse Cholesky factor R.

3. Perform the Cholesky factorization of $P_c^T A^T A P_c$ symbolically to generate a storage structure for R.

4. Compute $B = P_c^T A^T A P_c$ and $c = P_c^T A^T b$ numerically, storing B in the data structure of R.

5. Compute the Cholesky factor R numerically and solve $R^T z = c$, $Ry = z$, giving the solution $x = P_c y$.

Here steps 1, 2, and 3 involve only symbolic computation. It should be emphasized that the reason why the ordering algorithm in step 2 can be done symbolically only working on the structure of $A^T A$ is that pivoting is not required for numerical stability of the Cholesky algorithm.

For well-conditioned problems the method of normal equations is quite satisfactory, and often provides a solution of sufficient accuracy. For moderately ill-conditioned problems using the normal equations with iterative refinement (see Sec. 8.3.7) may be a good choice. For ill-conditioned or stiff problems methods based on the QR factorization avoid the loss of information caused by explicitly forming $A^T A$ and $A^T b$.

Mathematically the Cholesky factor of $A^T A$ equals to the upper triangular factor in the QR decomposition of A. Hence, the ordering methods discussed for the Cholesky factorization in Sec. 7.8.5, which work on the structure of these matrices, apply equally well to the QR factorization. In this section we discuss the numerical phase of sparse factorization methods. The main steps of a sparse QR algorithm are outlined below.

Algorithm 8.10. *Sparse QR Algorithm.*

1. Same as steps 1–3 in Algorithm 8.5.5.

2. Find a suitable row permutation P_r and reorder the rows to obtain $P_r A P_c$ (see Section 6.6.3).

3. Compute R and c numerically by applying orthogonal transformations to $(P_r A P_c, P_r b)$

4. Solve $Ry = c$ and take $x = P_c y$.

In predicting the structure of R from that of $B = P^T A^T A P$ by performing the Cholesky factor symbolically, $R^T + R$ will be at least as full as B. This may overestimate the structure of R. Both work and storage can sometimes be saved by first permuting the matrix A into a certain canonical block upper triangular form; see Sec. 8.5.4.

For dense problems the most effective serial method for computing the QR decomposition is to use a sequence of Householder reflections; see Algorithm 2.3.2. In this algorithm we put $A^{(1)} = A$, and compute $A^{(k+1)} = P_k A^{(k)}$, $k = 1, \ldots, n$, where

P_k is chosen to annihilate the subdiagonal elements in the kth column of $A^{(k)}$. In the sparse case this method will cause each column in the remaining unreduced part of the matrix, which has a nonzero inner product with the column being reduced, to take on the sparsity pattern of their union. In this way, even though the final R may be sparse, a lot of intermediate fill-in will take place with consequent cost in operations and storage. However, as was shown in Section 6.2.4, the Householder method can be modified to work efficiently for sparse banded problems, by applying the Householder reductions to a sequence of small dense subproblems. The generalization of this leads to multifrontal sparse QR methods; see Section 6.6.4. Here we first consider a row sequential algorithm by George and Heath [155, 1980], in which the problem with intermediate fill-in in the orthogonalization method is avoided by using a row-oriented method employing Givens rotations.

Assume that R_0 is initialized to have the structure of the final R and has all elements equal to zero. In the **row sequential QR algorithm** the rows a_k^T of A are processed sequentially, $k = 1, 2, \ldots, m$, and we denote by $R_{k-1} \in \mathbf{R}^{n\times n}$ the upper triangular matrix obtained after processing rows $a_1^T, \ldots, a_{k-1}^T$. The kth row $a_k^T = (a_{k1}, a_{k2}, \ldots, a_{kn})$ is processed as follows: we uncompress this row into a full vector and scan the nonzeros from left to right. For each $a_{kj} \neq 0$ a Givens rotation involving row j in R_{k-1} is used to annihilate a_{kj}. This may create new nonzeros both in R_{k-1} and in the row a_k^T. We continue until the whole row a_k^T has been annihilated. Note that if $r_{jj} = 0$, this means that this row in R_{k-1} has not yet been touched by any rotation and hence the entire jth row must be zero. When this occurs the remaining part of row k is inserted as the jth row in R.

To illustrate this algorithm we use the example in Figure 8.5.3, taken from George and Ng [157, 1983]. We assume that the first k rows of A have been processed to generate $R^{(k)}$. Nonzero elements of $R^{(k-1)}$ are denoted by $\times$, nonzero elements introduced into $R^{(k)}$ and a_k^T during the elimination of a_k^T are denoted by $+$, and all the elements involved in the elimination of a_k^T are circled. Nonzero elements created in a_k^T during the elimination are of course ultimately annihilated. The sequence of row indices involved in the elimination are $\{2, 4, 5, 7, 8\}$, where 2 is the column index of the first nonzero in a_k^T.

Note that unlike the Householder method intermediate fill now only takes place in the row that is being processed. It can be shown that if the structure of R has been predicted from that of A^TA, then any intermediate matrix R_{i-1} will fit into the predicted structure.

For simplicity we have not included the right-hand side in Figure 8.5.3, but the Givens rotations should be applied simultaneously to b to form Q^Tb. In the implementation by George and Heath [155, 1980] the Givens rotations are not stored but discarded after use. Hence, only enough storage to hold the final R and a few extra vectors for the current row and right-hand side(s) is needed in main memory. Discarding Q creates a problem if we later wish to solve additional problems having the same matrix A but a different right-hand side b since we cannot form Q^Tb. In most cases a satisfactory method to deal with this problem is to use the corrected seminormal equations; see [40, Sec. 6.6.5].

If Q is required, then the Givens rotations should be saved separately. This in

$$\begin{bmatrix} R_{k-1} \\ a_k^T \end{bmatrix} = \left[\begin{array}{cccccccccc}
\times & 0 & \times & 0 & 0 & \times & 0 & 0 & 0 & 0 \\
 & \otimes & 0 & \oplus & \otimes & 0 & 0 & 0 & 0 & 0 \\
 & & \times & 0 & \times & 0 & 0 & 0 & \times & 0 \\
 & & & \otimes & \oplus & 0 & \otimes & 0 & 0 & 0 \\
 & & & & \otimes & \oplus & 0 & 0 & 0 & 0 \\
 & & & & & \times & 0 & 0 & \times & 0 \\
 & & & & & & \otimes & \otimes & 0 & 0 \\
 & & & & & & & \otimes & 0 & 0 \\
 & & & & & & & & \times & \times \\
 & & & & & & & & 0 & \times \\
0 & \otimes & 0 & \otimes & \oplus & 0 & \oplus & \oplus & 0 & 0
\end{array}\right]$$

Figure 8.5.3. *Circled elements $\otimes$ in R_{k-1} are involved in the elimination of a_k^T; fill elements are denoted by $\oplus$.*

general requires far less storage and fewer operations than computing and storing Q itself; see Gilbert, Ng and Peyton [159].

Row Orderings for Sparse QR Factorization.

Assuming that the columns have been permuted by some ordering method, the final R is independent of the ordering of the rows in A. However, the number of operations needed to compute the QR decomposition may depend on the row ordering. This fact was stressed already in the discussion of algorithms for the QR decomposition of banded matrices; see Sec. 8.5.3. Another illustration is given by the contrived example (adapted from George and Heath [155, 1980]) in Figure 8.5.4. Here the cost for reducing A is $O(mn^2)$, but that for PA is only $O(n^2)$.

Assuming that the rows of A do not have widely differing norms, the row ordering does not affect numerical stability and can be chosen based on sparsity consideration only. We consider the following heuristic algorithm for determining a row ordering, which is an extension of the row ordering recommended for banded sparse matrices.

Algorithm 8.11. *Row Ordering Algorithm.*
Denote the column index for the first and last nonzero elements in the ith row of A by $f_i(A)$ and $l_i(A)$, respectively. First sort the rows after increasing $f_i(A)$, so that $f_i(A) \le f_k(A)$ if $i < k$. Then for each group of rows with $f_i(A) = k$, $k = 1, \ldots, \max_i f_i(A)$, sort all the rows after increasing $l_i(A)$.

Note that using this row ordering algorithm on the matrix A in Figure 8.5.4 will produce the good row ordering PA. This rule does not in general determine a unique ordering. One way to resolve ties is to consider the cost of symbolically rotating a row a_i^T into all other rows with a nonzero element in column $l_i(A)$. Here, by cost we mean the total number of new nonzero elements created. The rows are

$$A = \left(\begin{array}{ccccc} \times & \times & \times & \times & \times \\ \times & & & & \\ \vdots & & & & \\ \times & & & & \\ \hline \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{array}\right) \begin{array}{l} \left.\vphantom{\begin{array}{c}\times\\\times\\\vdots\\\times\end{array}}\right\} m \\ \left.\vphantom{\begin{array}{c}\times\\\times\\\times\\\times\\\times\end{array}}\right\} n \end{array}, \qquad PA = \left(\begin{array}{ccccc} \times & & & & \\ \times & & & & \\ \vdots & & & & \\ \times & & & & \\ \hline \times & \times & \times & \times & \times \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{array}\right) \begin{array}{l} \left.\vphantom{\begin{array}{c}\times\\\times\\\vdots\\\times\end{array}}\right\} m \\ \left.\vphantom{\begin{array}{c}\times\\\times\\\times\\\times\\\times\end{array}}\right\} n \end{array}.$$

Figure 8.5.4. *A bad and a good row ordering for QR factorization.*

then ordered according to ascending cost. For this ordering it follows that the rows $1, \ldots, f_i(A)-1$ in R_{i-1} will not be affected when the remaining rows are processed. These rows therefore are the final first $f_i(A)-1$ rows in R.

An alternative row ordering has been found to work well is obtained by ordering the rows after increasing values of $l_i(A)$. With this ordering only the columns $f_i(A)$ to $l_i(A)$ of R_{i-1} when row a_i^T is being processed will be involved, since all the previous rows only have nonzeros in columns up to at most $l_i(A)$. Hence, R_{i-1} will have zeros in column $l_{i+1}(A), \ldots, n$, and no fill will be generated in row a_i^T in these columns.

A significant advance in direct methods for sparse matrix factorization is the **multifrontal method**. This method reorganizes the factorization of a sparse matrix into a sequence of partial factorizations of small dense matrices, and is well suited for parallelism. A multifrontal algorithm for the QR decomposition was first developed by Liu[268]. Liu generalized the row-oriented scheme of George and Heath by using submatrix rotations and remarked that this scheme is essentially equivalent to a multifrontal method. This algorithm can give a significant reduction in QR decomposition time at a modest increase in working storage. George and Liu [156, 1987] presented a modified version of Liu's algorithm which uses Householder transformations instead of Givens rotations.

There are several advantages with the multifrontal approach. The solution of the dense subproblems can more efficiently be handled by vector machines. Also, it leads to independent subproblems which can be solved in parallel. The good data locality of the multifrontal method gives fewer page faults on paging systems, and out-of-core versions can be developed. Multifrontal methods for sparse QR decompositions have been extensively studied and several codes developed, e.g., by Matstoms [275].

8.5.6 Kronecker and Tensor Product Problems

Tensor decompositions: see [13, 243, 254, 255, 123].

Sometimes least squares problems occur which have a highly regular block structure. Here we consider least squares problems of the form

$$\min_x \|(A \otimes B)x - c\|_2, \quad c = \operatorname{vec} C, \tag{8.5.20}$$

where the $A \otimes B$ is the **Kronecker product** of $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{p\times q}$. This product is the $mp \times nq$ block matrix,

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

Problems of Kronecker structure arise in several application areas including signal and image processing, photogrammetry, and multidimensional approximation. It applies to least squares fitting of multivariate data on a rectangular grid. Such problems can be solved with great savings in storage and operations. Since often the size of the matrices A and B is large, resulting in models involving several hundred thousand equations and unknowns, such savings may be essential.

We recall from Sec. 7.7.3 the important rule (7.7.14) for the inverse of a Kronecker product

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}.$$

In particular, if P and Q are orthogonal $n \times n$ matrices then

$$(P \otimes Q)^{-1} = P^{-1} \otimes Q^{-1} = P^T \otimes Q^T = (P \otimes Q)^T,$$

where the last equality follows from the definition. Hence, $P \otimes Q$ is an orthogonal $n^2 \times n^2$ matrix. The rule for the inverse holds also for pseudo-inverses.

Lemma 8.5.2.

Let $A^\dagger$ and $B^\dagger$ be the pseudo-inverses of A and B. Then

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger.$$

Proof. The theorem follows by verifying that $X = A^\dagger \otimes B^\dagger$ satisfies the four Penrose conditions in (8.1.35)–(8.1.36). □

Using Lemmas 7.7.6 and 8.5.2 the solution to the Kronecker least squares problem (8.5.20) can be written

$$x = (A \otimes B)^\dagger \operatorname{vec} C = (A^\dagger \otimes B^\dagger)\operatorname{vec} C = \operatorname{vec}(B^\dagger C (A^\dagger)^T). \tag{8.5.21}$$

This formula leads to a great reduction in the cost of solving Kronecker least squares problems. For example, if A and B are both $m \times n$ matrices, the cost of computing is reduced from $O(m^2n^4)$ to $O(mn^2)$.

In some areas the most common approach to computing the least squares solution to (8.5.20) is to use the normal equations. If we assume that both A and B have full column rank, then we can use the expressions

$$A^\dagger = (A^TA)^{-1}A^T, \qquad B^\dagger = (B^TB)^{-1}B^T.$$

However, because of the instability associated with the explicit formation of A^TA and B^TB, an approach based on orthogonal decompositions should generally be preferred. If we have computed the complete QR decompositions of A and B,

$$A\Pi_1 = Q_1 \begin{pmatrix} R_1 & 0 \\ 0 & 0 \end{pmatrix} V_1^T, \qquad B\Pi_2 = Q_2 \begin{pmatrix} R_2 & 0 \\ 0 & 0 \end{pmatrix} V_2^T,$$

with R_1, R_2 upper triangular and nonsingular, then from Section 2.7.3 we have

$$A^\dagger = \Pi_1 V_1 \begin{pmatrix} R_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q_1^T, \qquad B^\dagger = \Pi_2 V_2 \begin{pmatrix} R_2^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q_2^T.$$

These expressions can be used in (8.5.21) to compute the pseudo-inverse solution of problem (8.5.20) even in the rank deficient case.

We finally note that the singular values and singular vectors of the Kronecker product $A \otimes B$ can be simply expressed in terms of the singular values and singular vectors of A and B.

Lemma 8.5.3. *Let A and B have the singular value decompositions*

$$A = U_1 \Sigma_1 V_1^T, \qquad B = U_2 \Sigma_2 V_2^T.$$

Then we have

$$A \otimes B = (U_1 \otimes U_2)(\Sigma_1 \otimes \Sigma_2)(V_1 \otimes V_2)^T.$$

Proof. The proof follows from Lemma 8.5.2. □

Review Questions

5.1 What is meant by the standard form of a banded rectangular matrix A? Why is it important that a banded matrix is permuted into standard form before its orthogonal factorization is computed?

5.2 In least squares linear regression the first column of A often equals the vector $a_1 = e = (1, 1, \dots, 1)^T$ (cf. Example 8.2.1). Setting $A = (\, e \quad A_2 \,)$, show that performing one step in MGS is equivalent to "subtracting out the means".

Problems

5.1 Consider the two-block least squares problem (8.5.1). Work out an algorithm to solve the reduced least squares problem $\min_{x_2} \|P_{\mathcal{N}(A_1^T)}(A_2x_2 - b)\|_2$ using the method of normal equations.

Hint: First show that $P_{\mathcal{N}(A_1^T)}(A) = I - A_1(R_1^TR_1)^{-1}A_1^T$, where R_1 is the Cholesky factor of $A_1^TA_1$.

5.2 (a) Write a MATLAB program for fitting a straight line $c_1x+c_2y = d$ to given points $(x_i, y_i) \in \mathbf{R}^2$, $i = 1 : m$. The program should handle all exceptional cases, e.g., $c_1 = 0$ and/or $c_2 = 0$.

(b) Suppose we want to fit two set of points $(x_i, y_i) \in \mathbf{R}^2$, $i = 1, \dots, p$, and $i = p+1, \dots, m$, to two *parallel* lines

$$cx + sy = h_1, \qquad cx + sy = h_2, \quad c^2 + s^2 = 1,$$

so that the sum of orthogonal distances are minimized. Generalize the approach in (a) to write an algorithm for solving this problem.

(c) Modify the algorithm in (a) to fit two *orthogonal* lines.

5.3 Use the Penrose conditions to prove the formula

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger,$$

where $\otimes$ denotes the Kronecker product

8.6 Some Generalized Least Squares Problems

8.6.1 Least Squares for General Linear Models

We consider a general univariate linear model where the error covariance matrix equals $\sigma^2 V$, where V is a positive definite symmetric matrix. Then the best unbiased linear estimate equals the minimizer of

$$(Ax - b)^T V^{-1} (Ax - b) \tag{8.6.1}$$

For the case of a general positive definite covariance matrix we assume that the Cholesky factorization $V = R^T R$ of the covariance matrix can be computed. Then the least squares problem (8.6.1) becomes

$$(Ax - b)^T V^{-1} (Ax - b) = \|R^{-T}(Ax - b)\|_2^2. \tag{8.6.2}$$

This could be solved in a straightforward manner by forming $\tilde{A} = R^{-T}A$, $\tilde{b} = R^{-T}b$ and solving the standard least squares problem

$$\min_x \|\tilde{A}x - \tilde{b}\|_2.$$

A simple special case of (8.6.1) is when the covariance matrix V is a positive diagonal matrix,

$$V = \sigma^2 \mathrm{diag}\,(v_1, v_2, \dots, v_m) > 0.$$

This leads to the **weighted** linear least squares problem (8.1.19)

$$\min_x \|D(Ax - b)\|_2 = \min_x \|(DA)x - Db|_2 \tag{8.6.3}$$

where $D = \mathrm{diag}\,(v_{ii}^{-1/2})$. Note that the weights will be large when the corresponding error component in the linear model has a small variance. We assume in the following that the matrix A is row equilibrated, that is,

$$\max_{1 \le j \le n} |a_{ij}| = 1, \quad i = 1 : m.$$

and that the weights $D = \operatorname{diag}(d_1, d_2, \dots, d_n)$ have been ordered so that

$$\infty > d_1 \geq d_2 \geq \cdots \geq d_m > 0. \tag{8.6.4}$$

Note that only the *relative size* of the weights influences the solution.

If the ratio $\gamma = d_1/d \gg 1$ we call the weighted problems **stiff**. Stiff problems arise, e.g., in electrical networks, certain classes of finite element problems, and in interior point methods for constrained optimization. Special care is needed in solving stiff weighted linear least squares problems. The method of normal equations is not well suited for solving such problems. To illustrate this, we consider the special case where only the first $p < n$ equations are weighted,

$$\min_x \left\| \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} x - \begin{pmatrix} \gamma b_1 \\ b_2 \end{pmatrix} \right\|_2^2, \tag{8.6.5}$$

$A_1 \in \mathbf{R}^{p\times n}$ and $A_2 \in \mathbf{R}^{(m-p)\times n}$. Such problems occur, for example, when the method of weighting is used to solve a least squares problem with the linear equality constraints $A_1 x = b_1$; see Section 5.1.4. For this problem the matrix of normal equations becomes

$$B = \begin{pmatrix} \gamma A_1^T & A_2^T \end{pmatrix} \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} = \gamma^2 A_1^T A_1 + A_2^T A_2.$$

If $\gamma > u^{-1/2}$ (u is the unit roundoff) and $A_1^T A_1$ is dense, then $B = A^T A$ will be completely dominated by the first term and the data contained in A_2 may be lost. If the number p of very accurate observations is less than n, then this is a disaster since the solution depends critically on the less precise data in A_2. (The matrix in Example 2.2.1 is of this type.)

Clearly, if the problem is stiff the condition number $\kappa(DA)$ will be large. An upper bound of the condition number is given by

$$\kappa(DA) \leq \kappa(D)\kappa(A) = \gamma\kappa(A).$$

It is important to note that this does not *mean that the problem of computing x from given data $\{D, A, b\}$ is ill-conditioned when $\gamma \gg 1$.*

We now examine the use of methods based on the QR factorization of A for solving weighted problems. We first note that it is essential that *column pivoting* is performed when QR factorization is used for weighted problems. To illustrate the need for column pivoting, consider an example of the form (8.6.5), where

$$A_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \end{pmatrix},$$

Then stability is lost without column pivoting because the first two columns of the matrix A_1 are linearly dependent. When column pivoting is introduced this difficulty disappears.

The following example shows that the Householder QR method can give poor accuracy for weighted problems even if column pivoting is used.

Example 8.6.1 (*Powell and Reid [315]*).
Consider the least squares problem with

$$DA = \begin{pmatrix} 0 & 2 & 1 \\ \gamma & \gamma & 0 \\ \gamma & 0 & \gamma \\ 0 & 1 & 1 \end{pmatrix}, \qquad Db = \begin{pmatrix} 3 \\ 2\gamma \\ 2\gamma \\ 2 \end{pmatrix}.$$

The exact solution is equal to $x = (1, 1, 1)$. Using exact arithmetic we obtain after the first step of QR factorization of A by Householder transformations the reduced matrix

$$\tilde{A}^{(2)} = \begin{pmatrix} \frac{1}{2}\gamma - 2^{1/2} & -\frac{1}{2}\gamma - 2^{-1/2} \\ -\frac{1}{2}\gamma - 2^{1/2} & \frac{1}{2}\gamma - 2^{-1/2} \\ 1 & 1 \end{pmatrix}.$$

If $\gamma > u^{-1}$ the terms $-2^{1/2}$ and $-2^{-1/2}$ in the first and second rows are lost. However, this is equivalent to the loss of all information present in the first row of A. This loss is disastrous because the number of rows containing large elements is less than the number of components in x, so there is a substantial dependence of the solution x on the first row of A. (However, compared to the method of normal equations, which fails already when $\gamma > u^{-1/2}$, this is an improvement!)

As shown by Cox and Higham [77], provided that an initial **row sorting** is performed the Householder QR method with column pivoting has very good stability properties for weighted problems. The rows of $\tilde{A} = DA$ and $\tilde{b} = Db$ should be sorted after decreasing ∞-norm,

$$\max_j |\tilde{a}_{1j}| \geq \max_j |\tilde{a}_{2j}| \geq \cdots \geq \max_j |\tilde{a}_{mj}|. \tag{8.6.6}$$

(In Example 8.6.1 this will permute the two large rows to the top.) Row pivoting could also be used, but row sorting has the advantage that after sorting the rows, any library routine for QR with column pivoting can be used.

If the Cholesky factor R is ill-conditioned there could be a loss of accuracy in forming $\tilde{A}$ and $\tilde{b}$. But assume that column pivoting has been used in the Cholesky factorization and set $R = D\hat{R}$, where $\hat{R}$ is *unit upper triangular* and D diagonal. Then any illcondioning is usually reflected in D. If neccessary a presorting of the rows of $\tilde{A} = D^{-1}\hat{R}^{-T}A$ can be performed before aplying the Householder QR method to the transformed problem.

In **Paige's method** [297] the general linear model with covariance matrix $V = BB^T$ is reformulated as

$$\min_{v,x} \|v\|_2 \quad \text{subject to} \quad Ax + Bv = b. \tag{8.6.7}$$

This formulation has the advantage that it is less sensitive to an ill-conditioned V and allows for rank deficiency in both A and V. For simplicity we consider here only the case when V is positive definite and $A \in \mathbf{R}^{m\times n}$ has full column rank n.

The first step is to compute the QR decomposition

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} \} & n \\ \} & m-n \end{matrix}, \quad Q^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \tag{8.6.8}$$

where R is nonsingular. The same orthogonal transformation is applied also to B, giving

$$Q^T B = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \begin{matrix} \} & n \\ \} & m-n \end{matrix},$$

where by the nonsingularity of B it follows that rank $(C_2) = m-n$. The constraints in (8.6.7) can now be written in the partitioned form

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} v + \begin{pmatrix} R \\ 0 \end{pmatrix} x = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix},$$

or

$$Rx = c_1 - C_1 v, \qquad C_2 v = c_2. \tag{8.6.9}$$

where $C_2 = \mathbf{R}^{(m-n)\times m}$. For any vector v we can determine x so that the first block of these equations is satisfied.

An orthogonal matrix $P \in R^{m\times m}$ can then be determined such that

$$C_2 P = (\,0 \quad S\,), \tag{8.6.10}$$

where S is upper triangular and nonsingular. Setting $u = P^T v$ the second block of the constraints in (8.6.9) becomes

$$C_2 P(P^T v) = (\,0 \quad S\,)\, u = c_2.$$

Since P is orthogonal $\|v\|_2 = \|u\|_2$ and so the minimum in (8.6.7) is found by taking

$$v = P \begin{pmatrix} 0 \\ u_2 \end{pmatrix}, \quad u_2 = S^{-1} c_2. \tag{8.6.11}$$

Then x is obtained from the triangular system $Rx = c_1 - C_1 v$. It can be shown that the computed solution is an unbiased estimate of x for the model (8.6.7) with covariance matrix $\sigma^2 C$, where

$$C = R^{-1} L^T L R^{-T}, \quad L^T = C_1^T P_1. \tag{8.6.12}$$

Paige's algorithm (8.6.8)–(8.6.11) as described above does not take advantage of any special structure the matrix B may have. It requires a total of about $4m^3/3 + 2m^2 n$ flops. If $m \gg n$ the work in the QR decomposition of C_2 dominates. It is not suitable for problems where B is sparse, e.g., diagonal as in the case of a weighted least squares problem.

Several generalized least squares problems can be solved by using a generalized SVD (GSVD) or QR (GQR) factorization involving a pair of matrices A, B. One motivation for using this approach is to avoid the explicit computation of products and quotients of matrices. For example, let A and B be square and nonsingular matrices and assume we need the SVD of AB^{-1} (or AB). Then the explicit calculation of AB^{-1} (or AB) may result in a loss of precision and should be avoided.

The factorization used in Paige's method is a special case of the generalized QR (GQR) factorization. Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times p}$ be a pair of matrices with the same number of rows. The GQR factorization of A and B is a factorization of the form

$$A = QR, \qquad B = QTZ, \tag{8.6.13}$$

where $Q \in \mathbb{R}^{m\times m}$ and $Z \in \mathbb{R}^{p\times p}$ are orthogonal matrices and R and T have one of the forms

$$R = \begin{pmatrix} R_{11} \\ 0 \end{pmatrix} \quad (m \geq n), \qquad R = \begin{pmatrix} R_{11} & R_{12} \end{pmatrix} \quad (m < n), \tag{8.6.14}$$

and

$$T = \begin{pmatrix} 0 & T_{12} \end{pmatrix} \quad (m \leq p), \qquad T = \begin{pmatrix} T_{11} \\ T_{21} \end{pmatrix} \quad (m > p). \tag{8.6.15}$$

If B is square and nonsingular GQR implicitly gives the QR factorization of $B^{-1}A$. There is also a similar generalized RQ factorization related to the QR factorization of AB^{-1}. Routines for computing a GQR decomposition of are included in LAPACK. These decompositions allow the solution of very general formulations of several least squares problems.

8.6.2 Indefinite Least Squares

A matrix $Q \in \mathbf{R}^{n\times n}$ is said to be J-orthogonal if

$$Q^T J Q = J, \tag{8.6.16}$$

where J is a **signature matrix**, i.e. a diagonal matrix with elements equal to ± 1. This implies that Q is nonsingular and $QJQ^T = J$. Such matrices are useful in the treatment of problems where there is an underlying indefinite inner product.

In order to construct J-orthogonal matrices we introduce the **exchange operator**. Consider the block 2×2 system

$$Qx = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}. \tag{8.6.17}$$

Solving the first equation for x_1 and substituting in the second equation gives

$$\operatorname{exc}(Q) \begin{pmatrix} y_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_2 \end{pmatrix},$$

where where x_1 and y_1 have been exchanged and

$$\operatorname{exc}(Q) = \begin{pmatrix} Q_{11}^{-1} & -Q_{11}^{-1}Q_{12} \\ Q_{21}Q_{11}^{-1} & Q_{22} - Q_{21}Q_{11}^{-1}Q_{12} \end{pmatrix} \tag{8.6.18}$$

Here the $(2,2)$ block is the Schur complement of Q_{11} in Q. From the definition it follows that the eschange operator satisfies

$$\mathrm{exc}\,(\mathrm{exc}\,(Q)) = Q,$$

that is it is involutary.

Theorem 8.6.1.

Let $Q \in \mathbf{R}^{n\times n}$ be partitioned as in (8.6.17). If Q is orthogonal and Q_{11} nonsingular then $\mathrm{exc}\,(Q)$ is J-orthogonal. If Q is J-orthogonal then $\mathrm{exc}\,(Q)$ is orthogonal.

Proof. A proof due to Chris Paige is given in Higham [212]. □

The indefinite least squares problem (ILS) has the form

$$\min_x (b - Ax)^T J(b - Ax), \tag{8.6.19}$$

where $A \in \mathbf{R}^{m\times n}$, $m \geq n$, and $b \in \mathbf{R}^m$ are given. In the following we assume for simplicity that

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad J = \begin{pmatrix} I_{m_1} & 0 \\ 0 & -I_{m_2} \end{pmatrix}, \tag{8.6.20}$$

where $m_1 + m_2 = m$, $m_1 m_2 \neq 0$. If the Hessian matrix $A^T JA$ is positive definite this problem has a unique solution which satisfies the normal equations$A^T JAx = A^T Jb$, or

$$(A_1^T A_1 - A_2^T A_2)x = A_1^T b_1 - A_2^T b_2. \tag{8.6.21}$$

Hence, if we form the normal equations and compute the Cholesky factorization $A^T JA = R^T R$ the solution is given by $x = R^{-1}R^{-T}c$, where $c = (A^T Jb)$. However, by numerical stability considerations we should avoid explicit formation of $A_1^T A_1$ and $A_2^T A_2$. We now show how J-orthogonal transformations can be used to solve this problem more directly.

Consider the orthogonal plane rotation

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c^2 + s^2 = 1.$$

where $c \neq 0$. As a special case of Theorem 8.6.1 it follows that

$$\breve{G} = \mathrm{exc}\,(G) = \frac{1}{c}\begin{pmatrix} 1 & -s \\ -s & 1 \end{pmatrix}, \tag{8.6.22}$$

is J-orthogonal

$$\breve{G}^T J \breve{G} = I, \qquad J = \mathrm{diag}\,(1, -1).$$

The matrix $\breve{G}$ is called a hyperbolic plane rotation since it can be written as

$$\breve{G} = \begin{pmatrix} \cosh\theta & -\sinh\theta \\ -\sinh\theta & \cosh\theta \end{pmatrix}, \quad \breve{c}^2 - \breve{s}^2 = 1.$$

A hyperbolic rotation $\breve{G}$ can be used to zero a selected component in a vector. Provided that $|\alpha| > |\beta|$ we have

$$\breve{G}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{c}\begin{pmatrix}1 & -s\\ -s & 1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\sigma\\ 0\end{pmatrix},$$

provided that

$$\sigma = \sqrt{\alpha^2 - \beta^2} = \sqrt{(\alpha+\beta)(\alpha-\beta)}, \qquad c = \sigma/\alpha, \quad s = \beta/\alpha. \tag{8.6.23}$$

Note that the elements of a hyperbolic rotation $\breve{G}$ are unbounded. Therefore, such transformations must be used with care. The direct application of $\breve{G}$ to a vector does not lead to a stable algorithm. Instead, as first suggested by Chambers [59], we note the equivalence of

$$\breve{G}\begin{pmatrix}x_1\\ x_2\end{pmatrix} = \begin{pmatrix}y_1\\ y_2\end{pmatrix} \quad \Leftrightarrow \quad G\begin{pmatrix}y_1\\ x_2\end{pmatrix} = \begin{pmatrix}x_1\\ y_2\end{pmatrix},$$

where $\breve{G} = \operatorname{exc}(G)$. We first determine y_1 from the hyperbolic rotation and then y_2 from the Given's rotation giving

$$y_1 = (x_1 - sx_2)/c, \qquad y_2 = cx_2 - sy_1. \tag{8.6.24}$$

We now describe an alternative algorithm for solving the indefinite least squares problem (8.6.19), which combines Householder transformations and hyperbolic rotations. In the first step we use two Householder transformations. The first transforms rows $1 : m_1$ in A_1 and zeros the elemnts $2 : m_1$. The second transforms rows $1 : m_2$ in A_2 and zeros elemnts $2 : m_2$. we now zero out the element left in the first column of A_2 using a hyperbolic rotation in the plane $(1, m_1 + 1)$. In the next step we proceed similarly. We first zero elements $3 : m_1$ in the second column of A_1 and elements $1 : m_2$ in the second column of A_2. A hyperbolic rotation in the plane $(2 : m_1 + 1)$ is then used to zero the remaining element in the second column of A_2. After the first two steps we have reduced the matrix A to the form

This method uses n hyperbolic rotations and n Householder transformations for the reduction. Since the cost for the hyperbolic rotations is $= (n^2)$ flops the total cost is about the same as for the usual Householder QR factorization.

Note that this can be combined with column pivoting so that at each step we maximize the diagonal element in R. It suffices to consider the first step; all remaining steps are similar. Changing notations to $A = (a_1, \ldots, a_n) \equiv A_1$, $B = (b_1, \ldots, b_n) \equiv A_2$, we do a modified Golub pivoting. Let p be the smallest index for which

$$s_p \geq s_j, \qquad s_j = \|a_j\|_2^2 - \|b_j\|_2^2, \quad \forall j = 1 : n,$$

and interchange columns 1 and p in A and B.

A special case of particular interest is when A_2 consists of a single row. In this case only hyperbolic transformations on A_2 occur.

8.6.3 Linear Equality Constraints

In some least squares problems in which the unknowns are required to satisfy a system of linear equations *exactly*. One source of such problems is in curve and surface fitting, where the curve is required to interpolate certain data points. Such problems can be considered as the limit of a sequence of weighted problems when some weights tend to infinity.

Given matrices $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{p\times n}$ we consider the problem **LSE** to find a vector $x \in \mathbf{R}^n$ which solves

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad Bx = d. \tag{8.6.25}$$

A solution to problem (8.6.25) exists if and only if the linear system $Bx = d$ is consistent. If $\operatorname{rank}(B) = p$ then B has linearly independent rows, and $Bx = d$ is consistent for any right hand side d. A solution to problem (8.6.25) is unique if and only if the null spaces of A and B intersect only trivially, i.e., if $\mathcal{N}(A)\cap\mathcal{N}(B) = \{0\}$, or equivalently

$$\operatorname{rank}\begin{pmatrix} A \\ B \end{pmatrix} = n. \tag{8.6.26}$$

If (8.6.26) is not satisfied then there is a vector $z \neq 0$ such that $Az = Bz = 0$. Hence, if x solves (8.6.25) then $x + z$ is a different solution. In the following we therefore assume that $\operatorname{rank}(B) = p$ and that (8.6.26) is satisfied.

A robust algorithm for problem LSE should check for possible inconsistency of the constraints $Bx = d$. If it is not known a priori that the constraints are consistent, then problem LSE may be reformulated as a **sequential least squares problem**

$$\min_{x\in S} \|Ax - b\|_2, \qquad S = \{x \mid \|Bx - d\|_2 = \min\ \}. \tag{8.6.27}$$

The most efficient way to solve problem LSE is to derive an equivalent unconstrained least squares problem of lower dimension. There are basically two different ways to perform this reduction: direct elimination and the null space method. We describe both these methods below.

In the method of **direct elimination** we start by reducing the matrix B to upper trapezoidal form. It is essential that column pivoting is used in this step. In order to be able to solve also the more general problem (8.6.27) we compute a QR factorization of B such that

$$Q_B^T B \Pi_B = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}, \quad R_{11} \in \mathbf{R}^{r\times r}, \tag{8.6.28}$$

where $r = \operatorname{rank}(B) \leq p$ and R_{11} is upper triangular and nonsingular. Using this factorization, and setting $\bar{x} = \Pi_B^T x$, the constraints become

$$(R_{11}, R_{12})\bar{x} = R_{11}\bar{x}_1 + R_{12}\bar{x}_2 = \bar{d}_1, \qquad \bar{d} = Q_B^T d = \begin{pmatrix} \bar{d}_1 \\ \bar{d}_2 \end{pmatrix}, \tag{8.6.29}$$

where $\bar{d}_2 = 0$ if and only if the constraints are consistent. If we apply the permutation Π_B also to the columns of A and partition the resulting matrix conformally with (8.6.28), $\bar{A}\Pi_B = (A_1, A_2)$. then $Ax - b = A_1\bar{x}_1 + A_2\bar{x}_2 - b$. Solving (8.6.29) for $\bar{x}_1 = R_{11}^{-1}(\bar{d}_1 - R_{12}\bar{x}_2)$, and substituting, we find that the unconstrained least squares problem

$$\begin{aligned} &\min_{\bar{x}_2} \|\hat{A}_2\bar{x}_2 - \hat{b}\|_2, \qquad \hat{A}_2 \in \mathbf{R}^{m\times(n-r)} \\ &\hat{A}_2 = \bar{A}_2 - \bar{A}_1 R_{11}^{-1} R_{12}, \qquad \hat{b} = b - \bar{A}_1 R_{11}^{-1}\bar{d}_1. \end{aligned} \tag{8.6.30}$$

is equivalent to the original problem LSE. Here $\hat{A}_2$ is the Schur complement of R_{11} in

$$\begin{pmatrix} R_{11} & R_{12} \\ \bar{A}_1 & \bar{A}_2 \end{pmatrix}.$$

It can be shown that if the condition in (8.6.26) is satisfied, then $\operatorname{rank}(A_2) = r$. Hence, the unconstrained problem has a unique solution, which can be computed from the QR factorization of $\hat{A}_2$. The coding of this algorithm can be kept remarkably compact as exemplified by the Algol program of Björck and Golub [42, 1967].

In the null space method we postmultiply B with an orthogonal matrix Q to transform B to lower triangular form. We also apply Q to the matrix A, which gives

$$\begin{pmatrix} B \\ A \end{pmatrix} Q = \begin{pmatrix} B \\ A \end{pmatrix} (\, Q_1 \quad Q_2 \,) = \begin{pmatrix} L & 0 \\ AQ_1 & AQ_2 \end{pmatrix}, \quad L \in \mathbf{R}^{p\times p}. \tag{8.6.31}$$

where Q_2 is an orthogonal basis for the null space of B, Note that this is equivalent to computing the QR factorization of B^T. The matrix Q can be constructed as a product of Householder transformations. The solution is now split into the sum of two orthogonal components by setting

$$x = Qy = x_1 + x_2 = Q_1 y_1 + Q_2 y_2, \quad y_1 \in \mathbf{R}^p, \quad y_2 \in \mathbf{R}^{(n-p)}, \tag{8.6.32}$$

where $Bx_2 = BQ_2y_2 = 0$. From the assumption that $\operatorname{rank}(B) = p$ it follows that L is nonsingular and the constraints equivalent to $y_1 = L^{-1}d$ and

$$b - Ax = b - AQy = c - AQ_2y_2, \quad c = b - (AQ_1)y_1.$$

Hence, y_2 is the solution to the unconstrained least squares problem

$$\min_{y_2} \|(AQ_2)y_2 - c\|_2. \tag{8.6.33}$$

This can be solved, for example, by computing the QR factorization of AQ_2. If (8.6.26) is satisfied then $\operatorname{rank}(AQ_2) = n - p$, then the solution to (8.6.33) is unique. If $y_2 = (AQ_2)^\dagger(b - AQ_1y_1)$ is the minimum length solution to (8.6.33), then since

$$\|x\|_2^2 = \|x_1\|_2^2 + \|Q_2y_2\|_2^2 = \|x_1\|_2^2 + \|y_2\|_2^2$$

$x = Qy$ is the minimum norm solution to problem LSE.

The representation in (8.6.32) of the solution x can be used as a basis for a perturbation theory for problem LSE. A strict analysis is given by Eldén [120, 1982], but the result is too complicated to be given here. If the matrix B is well conditioned, then the sensitivity is governed by $\kappa(AQ_2)$, for which $\kappa(A)$ is an upper bound.

The method of direct elimination and the null space method both have good numerical stability. If Gaussian elimination is used to derive the reduced unconstrained problem the operation count for the method of direct elimination is slightly lower.

8.6.4 Quadratic Inequality Constraints

Least squares problems with quadratic constraints arise, e.g., when one wants to balance a good fit to the data points and a smooth solution. Such problems arise naturally from inverse problems where one tries to determine the structure of a physical system from its behavior.

As an example, consider the integral equation of the first kind,

$$\int K(s,t)f(t)dt = g(s), \tag{8.6.34}$$

where the operator K is compact. It is well known that this is an ill-posed problem in the sense that the solution f does not depend continuously on the data g. This is because there are rapidly oscillating functions $f(t)$ which come arbitrarily close to being annihilated by the integral operator.

Let the integral equation (8.6.34) be discretized into a corresponding least squares problem

$$\min_f \|Ax - b\|_2. \tag{8.6.35}$$

The singular values of $A \in \mathbf{R}^{m\times n}$ will decay exponentially to zero. Hence, A will not have a well-defined numerical δ-rank r, since by (???) this requires that $\sigma_r > \delta \geq \sigma_{r+1}$ holds with a distinct gap between the singular values σ_r and σ_{r+1}. In general, any attempt to solve (8.6.35) without restricting f will lead to a meaningless solution of very large norm, or even to failure of the algorithm.

One of the most successful methods for solving ill-conditioned problems is **Tikhonov regularization**[28] (see [354, 1963]). In this method the solution space is restricted by imposing an a priori bound on $\|Lx\|_2$ for a suitably chosen matrix $L \in \mathbf{R}^{p\times n}$. Typically L is taken to be the identity matrix I or a discrete approximation to some derivative operator, e.g.,

$$L = \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{pmatrix} \in \mathbf{R}^{(n-1)\times n}, \tag{8.6.36}$$

[28]Andrei Nicholaevich Tikhonov (1906–1993), Russian mathematician. He made deep contributions in topology and function analysis, but was also interested in applications to mathematical physics. In the 1960's he introduced the concept of "regularizing operator" for ill-posed problems, for which he was awarded the Lenin medal.

which, except for a scaling factor, approximates the first derivative operator.

The above approach leads us to take x as the solution to the problem

$$\min_f \|Ax - b\|_2 \quad \text{subject to} \quad \|Lx\|_2 \leq \gamma. \tag{8.6.37}$$

Here the parameter γ governs the balance between a small residual and a smooth solution. The determination of a suitable γ is often a major difficulty in the solution process. Alternatively, we could consider the related problem

$$\min_f \|Lx\|_2 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \rho. \tag{8.6.38}$$

In the statistical literature the solution of problem (8.6.37) is called a **ridge estimate**.

Problems (8.6.37) and (8.6.38) are special cases of the general problem LSQI.

Problem LSQI:

Least Squares with Quadratic Inequality Constraint.

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad \|Lx - d\|_2 \leq \gamma, \tag{8.6.39}$$

where $A \in \mathbf{R}^{m\times n}$, $L \in \mathbf{R}^{p\times n}$, $\gamma > 0$.

Conditions for existence and uniqueness and properties of solutions to problem LSQI have been given by Gander [143, 1981]. Clearly, problem LSQI has a solution if and only if

$$\min_x \|Lx - d\|_2 \leq \gamma, \tag{8.6.40}$$

and in the following we assume that this condition is satisfied. We define a L-generalized solution $x_{A,L}$ to the problem $\min_x \|Ax - b\|_2$ to be a solution to the problem (cf. Section 2.7.4)

$$\min_{x\in S} \|Lx - d\|_2, \qquad S = \{x \in \mathbf{R}^n \| \;\; \|Ax - b\|_2 = \min\}. \tag{8.6.41}$$

These observation gives rise to the following theorem.

Theorem 8.6.2. *Assume that problem LSQI has a solution. Then either $x_{A,L}$ is a solution or* (8.6.48) *holds and the solution occurs on the boundary of the constraint region. In the latter case the solution $x = x(\lambda)$ satisfies the generalized normal equations*

$$(A^TA + \lambda L^TL)x(\lambda) = A^Tb + \lambda L^Td, \tag{8.6.42}$$

where $\lambda \geq 0$ is determined by the **secular equation**

$$\|Lx(\lambda) - d\|_2 = \gamma. \tag{8.6.43}$$

Proof. Since the solution occurs on the boundary we can use the method of Lagrange multipliers and minimize $\psi(x, \lambda)$, where

$$\psi(x, \lambda) = \tfrac{1}{2}\|Ax - b\|_2^2 + \tfrac{1}{2}\lambda(\|Lx - d\|_2^2 - \gamma^2). \tag{8.6.44}$$

A necessary condition for a minimum is that the gradient of $\psi(x, \lambda)$ with respect to x equals zero, which gives (8.6.42). □

As we shall see, only positive values of λ are of interest. Note that (8.6.42) are the normal equations for the least squares problem

$$\min_x \left\| \begin{pmatrix} A \\ \sqrt{\lambda} L \end{pmatrix} x - \begin{pmatrix} b \\ \mu d \end{pmatrix} \right\|_2. \tag{8.6.45}$$

Hence, to solve (8.6.42) for a given value of λ, it is not necessary to form the cross-product matrices $A^T A$ and $L^T L$.

In the following we assume that (8.6.48) holds so that the constraint $\|Lx(\lambda) - d\|_2 \le \gamma$ is binding. Then there is a unique solution to problem LSQI if and only if the null spaces of A and L intersect only trivially, i.e., $\mathcal{N}(A) \cap \mathcal{N}(L) = \{0\}$. This is equivalent to

$$\operatorname{rank} \begin{pmatrix} A \\ L \end{pmatrix} = n. \tag{8.6.46}$$

A particularly simple but important case is when $L = I_n$ and $d = 0$, i.e.,

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad \|x\|_2 \le \gamma, \tag{8.6.47}$$

We call this the **standard form** of LSQI. Notice that for this case we have $x_{A,I} = A^I b$ and the constraint is binding only if

$$\|Lx_{A,L} - d\|_2 > \gamma. \tag{8.6.48}$$

The special structure can be taken advantage of when computing the Householder QR factorization of the matrix in (8.6.45).

Notice that the first two rows of D have filled in, but the remaining rows of D are still not touched. For each step $k = 1 : n$ there are m elements in the current column to be annihilated. Therefore, the operation count for the Householder QR factorization will increase with $2n^3/3$ to $2mn^2$ flops. If $A = R$ already is in upper triangular form then the flop count for the reduction is reduced to approximately $2n^3/3$ (cf. Problem 1b).

If $L = I$ the singular values of the modified matrix in (8.6.45) are equal to

$$\tilde{\sigma}_i = (\sigma_i^2 + \lambda)^{1/2}, \quad i = 1 : n.$$

In this case the solution can be expressed in terms of the SVD as

$$x(\lambda) = \sum_{i=1}^{n} f_i \frac{c_i}{\sigma_i} v_i, \quad f_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda}. \tag{8.6.49}$$

The quantities f_i are often called **filter factors**. Notice that as long as $\sqrt{\lambda} \ll \sigma_i$ we have $f_i \approx 1$, and if $\sqrt{\lambda} \gg \sigma_i$ then $f_i \ll 1$. This establishes a relation to the truncated SVD solution (8.4.13) which corresponds to a filter factor which is a step function $f_i = 1$ if $\sigma_i > \delta$ and $f_i = 0$ otherwise.

Even with regularization we may not be able to compute the solution of an ill-conditioned problem with the accuracy that the data allows. In those cases it is possible to improve the solution by the following **iterated regularization** scheme. Take $x^{(0)} = 0$, and compute a sequence of approximate solutions by

$$x^{(q+1)} = x^{(q)} + \delta x^{(q)},$$

where $\delta x^{(q)}$ solves the least squares problem

$$\min_{\delta x} \left\| \begin{pmatrix} A \\ \mu I \end{pmatrix} \delta x - \begin{pmatrix} r^{(q)} \\ 0 \end{pmatrix} \right\|_2, \qquad r^{(q)} = b - Ax^{(q)}. \tag{8.6.50}$$

This iteration may be implemented very effectively since only one QR factorization is needed. The convergence of iterated regularization can be expressed in terms of the SVD of A.

$$x^{(q)}(\lambda) = \sum_{i=1}^{n} f_i^{(q)} \frac{c_i}{\sigma_i} v_i, \quad f_i^{(q)} = 1 - \Big(\frac{\lambda}{\sigma_i^2 + \lambda}\Big)^q. \tag{8.6.51}$$

Thus, for $q = 1$ we have the standard regularized solution and as $q \to \infty$ $x^{(q)} \to A^\dagger b$.

Solving the Secular Equation.

Methods for solving problem LSQI are usually based on solving the secular equation (8.6.43) using Newton's method. The secular equation can be written in the form

$$f_p(\lambda) = \|x(\lambda)\|^p - \gamma^p = 0. \tag{8.6.52}$$

where $p = \pm 1$ and $x(\lambda)$ be the solution to the least squares problem (8.6.45). From $\|x(\lambda)\|_2^p = (x(\lambda)^T x(\lambda))^{p/2}$, taking derivatives with respect to λ, we find

$$f_p'(\lambda) = p\frac{x^T(\lambda)x'(\lambda)}{\|x(\lambda)\|_2^{2-p}}, \qquad x'(\lambda) = -(A^TA + \lambda I)^{-1}x(\lambda). \tag{8.6.53}$$

Since $x(\lambda) = (A^TA + \lambda I)^{-1}A^Tb$, we obtain

$$x(\lambda)^T x(\lambda)' = -x(\lambda)^T (A^TA + \lambda I)^{-1} x(\lambda) = -\|z(\lambda)\|_2^2.$$

The choice $p = +1$ gives rise to the iteration

$$\lambda_{k+1} = \lambda_k + \left(1 - \frac{\gamma}{\|x(\lambda_k)\|_2}\right) \frac{\|x(\lambda_k)\|_2^2}{\|z(\lambda_k)\|_2^2}. \tag{8.6.54}$$

The choice $p = -1$ gives the iteration

$$\lambda_{k+1} = \lambda_k - \left(1 - \frac{\|x(\lambda_k)\|_2}{\gamma}\right) \frac{\|x(\lambda_k)\|_2^2}{\|z(\lambda_k)\|_2^2}, \tag{8.6.55}$$

which is due to Hebden [203] and Reinsch [320]).

For $p = \pm 1$ Newton's method will converge monotonically provided that the initial approximation satisfies $0 \leq \lambda^{(0)} < \lambda$. Therefore, $\lambda^{(0)} = 0$ is often used as a starting approximation. The asymptotic rate of convergence is quadratic. Close to the solution $\|x(\lambda_k)\| \approx \gamma$, and then the Newton correction is almost the same independent of p. However, when λ is small, we can have $\|x(\lambda_k)\| \gg \gamma$. For $p = -1$ the $f_p(\lambda)$ is close to linear for sufficiently small λ and this gives much more rapid convergence than $p = 1$.

It has been shown (Reinsch [320]) that $h(\lambda)$ is convex, and hence that the iteration (8.6.55) is monotonically convergent to the solution λ^* if started within $[0, \lambda^*]$. Note that the correction to λ_k in (8.6.55) equals the Newton correction in (8.6.54) multiplied by the factor $\|x(\lambda)\|/\gamma$.

Using the QR factorization

$$Q(\lambda)^T \begin{pmatrix} A \\ \sqrt{\lambda} I \end{pmatrix} = \begin{pmatrix} R(\lambda) \\ 0 \end{pmatrix}, \qquad c_1(\lambda) = \begin{pmatrix} I_n & 0 \end{pmatrix} Q(\lambda)^T \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{8.6.56}$$

we have

$$x(\lambda) = R(\lambda)^{-1} c_1(\lambda), \qquad z(\lambda) = R(\lambda)^{-T} x(\lambda). \tag{8.6.57}$$

The main cost in this method is for computing the QR decomposition (8.6.56) in each iteration step. On the other hand computing the derivative costs only one triangular solve. Assume that the function qr computes the "thin" QR factorization, with $Q \in \mathbf{R}^{m\times n}$. Then Hebden's algorithm is:

Algorithm 8.12. *Hebden's method.*

The algorithm performs p steps of Hebden's iteration to the constrained least squares problem $\min \|Ax - b\|_2$ subject to $\|x\|_2 = gamma > 0$, using QR factorization starting from a user supplied value λ_0..

$\lambda = \lambda_0;$
for $k = 1 : p \ldots$
  $[Q, R] = \mathrm{qr}\big([A; \sqrt{\lambda} I]\big);$
  $c = Q^T * b;$
  $x = R^{-1} * c;$
  if $k \leq p$ $\quad$ $\%update \lambda$
    $z = R^{-T} * x;$
    $n_x = \|x\|_2; \quad n_z = \|z\|_2;$
    $\lambda = \lambda + (n_x/\gamma - 1) * (n_x/n_z)^2;$
  end
end

It is slightly more efficient to initially compute the QR decompositions of A in $mn^2 - n^3/3$ multiplications. Then for each new value of λ the QR factorization of

$$Q^T(\lambda)\begin{pmatrix} R(0) \\ \sqrt{\lambda} I \end{pmatrix} = \begin{pmatrix} R(\lambda) \\ 0 \end{pmatrix}$$

can be computed in just $n^3/3$ multiplications. Then p Newton iterations will require a total of $mn^2 + (p-1)n^3/3$ multiplications. In practice $p \approx 6$ iterations usually suffice to achieve full accuracy. Further, savings are possible by initially transforming A to bidiagonal form; see Eldén [122].

Transformation to Standard Form

A Problem LSQI with $L \neq I$ can be transformed to standard form as we now describe. If L is nonsingular we can achieve this by the change of variables $y = Lx$. However, often $L \in \mathbf{R}^{(n-t)\times n}$ and has full row rank. For example, in with L as in (8.6.36) the rank deficiency is $t = 1$. The transformation to standard form can then be achieved using the pseudo-inverse of L. Let the QR decomposition of L^T be

$$L^T = (V_1, V_2)\begin{pmatrix} R_2 \\ 0 \end{pmatrix},$$

where V_2 spans the null space of L. If we set $y = Lx$, then

$$x = L^\dagger y + V_2 w, \qquad L^\dagger = V_1 R_2^{-T}, \tag{8.6.58}$$

where $L^\dagger$ is the pseudo-inverse of L, and

$$Ax - b = AL^\dagger y - b + AV_2 w.$$

We form $AV_2 \in \mathbf{R}^{m\times t}$ and compute its QR decomposition

$$AV_2 = (Q_1, Q_2)\begin{pmatrix} U \\ 0 \end{pmatrix}, \qquad U \in \mathbf{R}^{t\times t}.$$

Then

$$Q^T(Ax - b) = \begin{pmatrix} Q_1^T(AL^\dagger y - b) + Uw \\ Q_2^T(AL^\dagger y - b) \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}.$$

If A and L have no null space in common, then AV_2 has rank t and U is nonsingular. Thus, we can always determine w so that $r_1 = 0$ and Problem LSQI is equivalent to

$$\min_y \left\| \begin{pmatrix} \tilde{A} \\ \mu I \end{pmatrix} y - \begin{pmatrix} \tilde{b} \\ 0 \end{pmatrix} \right\|_2, \quad \tilde{A} = Q_2^T A L^\dagger, \quad \tilde{b} = Q_2^T b, \tag{8.6.59}$$

which is of standard form. We then retrieve x from (8.6.58).

An important special case is when in LSQI we have $A = K$, $L = L$, and both K and L are upper triangular Toeplitz matrices, i.e.,

$$K = \begin{pmatrix} k_1 & k_2 & \dots & k_{n-1} & k_n \\ & k_1 & k_2 & & k_{n-1} \\ & & \ddots & \ddots & \vdots \\ & & & k_1 & k_2 \\ & & & & k_1 \end{pmatrix}$$

and L is as in (8.6.36). Such systems arise when convolution-type Volterra integral equations of the first kind,

$$\int_0^t K(t-s)f(t)dt = g(s), \qquad 0 \le t \le T,$$

are discretized. Eldén [122] has developed a method for solving problems of this kind which only uses $\frac{9}{2}n^2$ flops for each value of μ. It can be modified to handle the case when K and L also have a few nonzero diagonals below the main diagonal. Although K can be represented by n numbers this method uses $n^2/2$ storage locations. A modification of this algorithm which uses only $O(n)$ storage locations is given in Bojanczyk and Brent [45, 1986].

8.6.5 Linear Orthogonal Regression

Let P_i, $i = 1 : m$, be a set of given points in $\mathbf{R}^n$. In the **orthogonal regression** problem we want to fit a hyper plane M to the points in such a way that the sum of squares of the orthogonal distances from the given points to M is minimized.

We first consider the special case of fitting a straight line to points in the plane. Let the coordinates of the points be (x_i, y_i) and let the line have the equation

$$c_1 x + c_2 y + d = 0, \tag{8.6.60}$$

where $c_1^2 + c_2^2 = 1$. Then the orthogonal distance from the point $P_i = (x_i, y_i)$ to the line equals $r_i = c_1 x_i + c_2 y_i + d$. Thus, we want to minimize

$$\sum_{i=1}^{m} (c_1 x_i + c_2 y_i + d)^2, \tag{8.6.61}$$

subject to the constraint $c_1^2 + c_2^2 = 1$. This problem can be written in matrix form

$$\min_{c,d} \left\| (\, e \quad Y \,) \begin{pmatrix} d \\ c \end{pmatrix} \right\|_2, \quad \text{subject to} \quad c_1 + c_2 = 1,$$

where $c = (c_1 \; c_2)^T$ and

$$(\, e \quad Y \,) = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \\ 1 & x_m & y_m \end{pmatrix}.$$

By computing the QR factorization of the matrix $(\, e \quad Y \,)$ and using the invariance of the Euclidean norm this problem is reduced to

$$\min_{d,c} \left\| R \begin{pmatrix} d \\ c \end{pmatrix} \right\|_2, \quad R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{pmatrix}.$$

For any values of c_1 and c_2 we can always determine d so that $r_{11}d+r_{12}c_1+r_{13}c_2 = 0$. Thus, it remains to determine c so that $\|Bc\|_2$ is minimized, subject to $\|c\|_2 = 1$, where

$$Bc = \begin{pmatrix} r_{22} & r_{23} \\ 0 & r_{33} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}.$$

By the min-max characterization of the singular values (Theorem 8.1.13) the solution equals the right singular vector corresponding to the smallest singular value of the matrix B. Let the SVD be

$$\begin{pmatrix} r_{21} & r_{22} \\ 0 & r_{33} \end{pmatrix} = (u_1 \ u_2) \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix},$$

where $\sigma_1 \geq \sigma_2 \geq 0$. (A stable algorithm for computing the SVD of an upper triangular matrix is given in Algorithm 9.4.2; see also Problem 9.4.5.) Then the coefficients in the equation of the straight line are given by

$$(c_1 \quad c_2) = v_2^T.$$

If $\sigma_2 = 0$ but $\sigma_1 > 0$ the matrix B has rank one. In this case the given points lie on a straight line. If $\sigma_1 = \sigma_2 = 0$, then $B = 0$, and all points coincide, i.e. $x_i = \bar{x}$, $y_i = \bar{y}$ for all $i = 1 : m$. Note that v_2 is uniquely determined if and only if $\sigma_1 \neq \sigma_2$. It is left to the reader to discuss the case $\sigma_1 = \sigma_2 \neq 0$!

In [146, Chapter 6] a similar approach is used to solve various other problems, such as fitting two parallel or orthogonal lines or fitting a rectangle or square.

We now consider the general problem of fitting $m > n$ points $P_i \in \mathbf{R}^n$ to a hyper plane M so that the sum of squares of the orthogonal distances is minimized. The equation for the hyper plane can be written

$$c^T z = d, \quad z, c \in \mathbf{R}^n, \quad \|c\|_2 = 1,$$

where $c \in \mathbf{R}^n$ is the normal vector of M, and $|d|$ is the orthogonal distance form the origin to the plane. Then the orthogonal projections of the points y_i onto M are given by

$$z_i = y_i - (c^T y_i - d)c. \tag{8.6.62}$$

It is readily verified that the point z_i lies on M and the residual $(z_i - y_i)$ is parallel to c and hence orthogonal to M. It follows that the problem is equivalent to minimizing

$$\sum_{i=1}^{m} (c^T y_i - d)^2, \quad \text{subject to} \quad \|c\|_2 = 1.$$

If we put $Y^T = (y_1, \ldots, y_m) \in \mathbf{R}^{n\times m}$ and $e = (1, \ldots, 1)^T \in \mathbf{R}^m$, this problem can be written in matrix form

$$\min_{c,d} \left\| (-e \quad Y) \begin{pmatrix} d \\ c \end{pmatrix} \right\|_2, \quad \text{subject to} \quad \|c\|_2 = 1. \tag{8.6.63}$$

For a fixed c, this expression is minimized when the residual vector $(Yc - de)$ is orthogonal to e, that is $e^T(Yc - de) = e^TYc - de^Te = 0$. Since $e^Te = m$ it follows that

$$d = \frac{1}{m}c^TY^Te = c^T\bar{y}, \qquad \bar{y} = \frac{1}{m}Y^Te, \tag{8.6.64}$$

where $\bar{y}$ is the mean value of the given points y_i. Hence, d is determined by the condition that the mean value $\bar{y}$ lies on the optimal plane M. Note that this property is shared by the usual linear regression.

We now subtract the mean value $\bar{y}$ from each given point, and form the matrix

$$\overline{Y}^T = (\bar{y}_1, \ldots, \bar{y}_m), \qquad \bar{y}_i = y_i - \bar{y}, \quad i = 1:m.$$

Since by (8.6.64)

$$\begin{pmatrix} -e & Y \end{pmatrix}\begin{pmatrix} d \\ c \end{pmatrix} = Yc - e\bar{y}^Tc = (Y - e\bar{y}^T)c = \bar{Y}c,$$

problem (8.6.63) is equivalent to

$$\min_n \|\bar{Y}c\|_2, \qquad \|c\|_2 = 1 \tag{8.6.65}$$

By the min-max characterization of the singular values a solution to (8.6.65) is given by $c = v_n$, where v_n is a right singular vector of $\bar{Y}$ corresponding to the smallest singular value σ_n. We further have

$$c = v_n, \qquad d = v_n^T\bar{y}, \qquad \sum_{i=1}^m (v_n^Ty_i - d)^2 = \sigma_n^2,$$

The fitted points $z_i \in M$ are obtained from

$$z_i = \bar{y}_i - (v_n^T\bar{y}_i)v_n + \bar{y},$$

i.e., by first orthogonalizing the shifted points $\bar{y}_i$ against v_n, and then adding the mean value back.

Note that the orthogonal regression problem always has a solution. The solution is unique when $\sigma_{n-1} > \sigma_n$, and the minimum sum of squares equals σ_n^2. We have $\sigma_n = 0$, if and only if the given points y_i, $i = 1:m$ all lie on the hyper plane M. In the extreme case, all points coincide and then $\bar{Y} = 0$, and any plane going through $\bar{y}$ is a solution.

The above method solves the problem of fitting a $(n-1)$ dimensional linear manifold to a given set of points in $\mathbf{R}^n$. It is readily generalized to the fitting of an $(n-p)$ dimensional manifold by orthogonalizing the shifted points y against the p right singular vectors of Y corresponding to p smallest singular values. A least squares problem that often arises is to fit to given data points a geometrical element, which may be defined in implicit form. For example, the problem of fitting circles, ellipses, spheres, and cylinders arises in applications such as computer graphics, coordinate meteorology, and statistics. Such problems are nonlinear and will be discussed in Sec. 11.4.7.

8.6.6 The Orthogonal Procrustes Problem

Let A and B be given matrices in $\mathbf{R}^{m\times n}$. A problem that arises, e.g., in factor analysis in statistics is the **orthogonal Procrustes**[29]**problem**

$$\min_Q \|A - BQ\|_F \quad \text{subject to} \quad Q^TQ = I. \tag{8.6.66}$$

Another example is in determining rigid body movements. Suppose that $a_1, a_2, \ldots, a_m$ are measured positions of $m \geq n$ landmarks in a rigid body in $\mathbf{R}^n$. Let $b_1, b_2, \ldots, b_m$ be the measured positions after the body has been rotated. We seek an orthogonal matrix $Q \in \mathbf{R}^{n\times n}$ representing the rotation of the body.

The solution can be computed from the polar decomposition of B^TA as shown by the following generalization of Theorem 8.1.17 by P. Schönemann [331]:

Theorem 8.6.3.

Let $\mathcal{M}_{m\times n}$ denote the set of all matrices in $\mathbf{R}^{m\times n}$ with orthogonal columns. Let A and B be given matrices in $\mathbf{R}^{m\times n}$. If $B^TA = PH$ is the polar decomposition then

$$\|A - BQ\|_F \geq \|A - BP\|_F$$

for any matrix $Q \in \mathcal{M}_{m\times n}$.

Proof. Recall from (7.1.75) that $\|A\|_F^2 = \operatorname{trace}(A^TA)$ and that $\operatorname{trace}(X^TY) = \operatorname{trace}(YX^T)$. Using this and the orthogonality of Q, we find that

$$\|A - BQ\|_F^2 = \operatorname{trace}(A^TA) + \operatorname{trace}(B^TB) - 2\operatorname{trace}(Q^TB^TA).$$

It follows that the problem (8.6.66) is equivalent to maximizing $\operatorname{trace}(Q^TB^TA)$. Let the SVD of B^TA be $B^TA = U\Sigma V^T$, where $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$. Set $Q = UZV^T$ where Z is orthogonal. Then we have $\|Z\|_2 = 1$ and hence its diagonal elements satisfy $|z_{ii}| \leq 1$, $i = 1 : n$. Hence

$$\begin{aligned}\operatorname{trace}(Q^TB^TA) &= \operatorname{trace}(VZ^TU^TB^TA) = \operatorname{trace}(Z^TU^TB^TAV)\\ &= \operatorname{trace}(Z^T\Sigma) = \sum_{i=1}^n z_{ii}\sigma_i \leq \sum_{i=1}^n \sigma_i.\end{aligned}$$

The upper bound is obtained for $Q = UV^T$. This solution is not unique unless $\operatorname{rank}(A) = n$. □

In many applications it is important that Q corresponds to a pure rotation, that is $\det(Q) = 1$. If $\det(UV^T) = 1$, the optimal is $Q = UV^T$ as before. If $\det(UV^T) = -1$, the optimal solution can be shown to be (see [201])

$$Q = UZV^T, \quad Z = \operatorname{diag}(1, \ldots, 1, -1)$$

[29]Procrustes was a giant of Attica in Greece who seized travelers, tied them to a bedstead, and either stretched them or chopped off their legs to make them fit it.

which has determinant equal to 1. For this choice

$$\sum_{i=1}^{n} z_{ii}\sigma_i = \operatorname{trace}(\Sigma) - 2\sigma_n.$$

Thus, in both cases the optimal solution can be written

$$Q = UZV^T, \quad Z = \operatorname{diag}(1, \dots, 1, \det(UV^T)).$$

Perturbation bounds for the polar factorization are derived in Barrlund [24]. A perturbation analysis of the orthogonal Procrustes problem is given by Söderlind [337].

In analysis of rigid body movements there is also a translation $c \in Rn$ involved. Then we have the model

$$A = BQ + ec^T, \quad e = (1, 1, \dots, 1)^T \in \mathbf{R}^m,$$

where we now want to estimate also the translation vector $c \in \mathbf{R}^n$. The problem now is

$$\min_{Q,c} \|A - BQ - ec^T\|_F \quad \text{subject to} Q^T Q = I \tag{8.6.67}$$

and $\det(Q) = 1$. For any Q including the optimal Q we do not yet know, the best least squares estimate of c is characterized by the condition that the residual is orthogonal to e. Multiplying by e^T we obtain

$$0 = e^T(A - BQ - ec^T) = e^T A - (e^T B)Q - mc^T = 0,$$

where $e^T A/m$ and $e^T B/m$ are the mean values of the rows in A and B, respectively. Hence, the optimal translation satisfies

$$c = \frac{1}{m}((B^T e)Q - A^T e). \tag{8.6.68}$$

Substituting this expression into (8.6.67) we can eliminate c and the problem becomes

$$\min_{Q} \|\tilde{A} - \tilde{B}Q\|_F,$$

where

$$\tilde{A} = A - \frac{1}{m}ee^T A, \qquad \tilde{B} = B - \frac{1}{m}ee^T B.$$

This is now a standard orthogonal Procrustes problem and the solution is obtained from the SVD of $\tilde{A}^T\tilde{B}$.

If the matrix A is close to an orthogonal matrix, then an iterative method for computing the polar decomposition can be used. Such methods are developed in Sec. 9.5.5.

Review Questions

7.1 What is meant by a saddle-point system? Which two optimization problems give rise to saddle-point systems?

Problems

7.1 Consider the overdetermined linear system $Ax = b$ in Example 8.2.4. Assume that $\epsilon^2 \leq u$, where u is the unit roundoff, so that $fl(1+\epsilon^2) = 1$.

(a) Show that the condition number of A is $\kappa = \epsilon^{-1}\sqrt{3+\epsilon^2} \approx \epsilon^{-1}\sqrt{3}$.

(b) Show that if no other rounding errors are made then the maximum deviation from orthogonality of the columns computed by CGS and MGS, respectively, are

$$\text{CGS}: \quad |q_3^T q_2| = 1/2, \qquad \text{MGS}: \quad |q_3^T q_1| = \frac{\epsilon}{\sqrt{6}} \leq \frac{\kappa}{3\sqrt{3}} u.$$

Note that for CGS orthogonality has been completely lost!

7.2 Assume that $A \in \mathbf{R}^{m\times m}$ is symmetric and positive definite and $B \in \mathbf{R}^{m\times n}$ a matrix with full column rank. Show that

$$M = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & -S \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix},$$

where $S = B^T A^{-1} B$ is the Schur complement (cf. (7.1.18)). Conclude that M is indefinite! (M is called a saddle point matrix.)

8.7 The Total Least Squares Problem

8.7.1 Total Least Squares and the SVD

In the standard linear model (8.1.15) it is assumed that the vector $b \in \mathcal{R}^m$ is related to the unknown parameter vector $x \in \mathcal{R}^n$ by a linear relation $Ax = b + e$, where $A \in \mathcal{R}^{m\times n}$ is an exactly known matrix and e a vector of random errors. If the components of e are uncorrelated, have zero means and the same variance, then by the Gauss–Markov theorem (Theorem 8.1.7) the best unbiased estimate of x is obtained by solving the least squares problem

$$\min_x \|r\|_2, \qquad Ax = b + r. \tag{8.7.1}$$

The assumption in the least squares problem that all errors are confined to the right hand side b is frequently unrealistic, and sampling or modeling errors often will affect also the matrix A. In the **errors-in-variables model** it is assumed that a linear relation

$$(A + E)x = b + r,$$

where the rows of the errors $(E,\ r)$ are *independently and identically distributed with zero mean and the same variance*. If this assumption is not satisfied it might be possible to find scaling matrices $D = \operatorname{diag}(d_1, \ldots, d_m)$, $T = \operatorname{diag}(d_1, \ldots, d_{n+1})$, such that $D(A,\ b)T)$ satisfies this assumptions.

Estimates of the unknown parameters x in this model can be obtained from the solution of the **total least squares** (TLS) problem The term “total least squares problem" was coined by Golub and Van Loan in [183]. The concept has been independently developed in other areas. For example, in statistics this is also known as "latent root regression".

$$\min_{E,\,r} \|(r,\, E)\|_F, \qquad (A+E)x = b + r, \tag{8.7.2}$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm defined by

$$\|A\|_F^2 = \sum_{i,j} a_{ij}^2 = \operatorname{trace}(A^T A).$$

The constraint in (8.7.2) implies that $b + r \in \mathcal{R}(A + E)$. Thus, the total least squares is equivalent to the problem of finding the "nearest" compatible linear system, where the distance is measured by the Frobenius norm. If a minimizing perturbation $(E,\ r)$ has been found for the problem (8.7.2) then any x satisfying $(A+E)x = b + r$ is said to solve the TLS problem.

The TLS solution will depend on the scaling of the data $(A,\, b)$. In the following we assume that this scaling has been carried out in advance, so that any statistical knowledge of the perturbations has been taken into account. In particular, the TLS solution depends on the relative scaling of A and b. If we scale x and b by a factor γ we obtain the **scaled TLS problem**

$$\min_{E,\,r} \|(E,\, \gamma r)\|_F \qquad (A+E)x = b + r.$$

Clearly, when γ is small perturbations in b will be favored. In the limit when $\gamma \to 0$ we get the ordinary least squares problem. Similarly, when γ is large perturbations in A will be favored. In the limit when $1/\gamma \to 0$, this leads to the **data least squares** (DLS) problem

$$\min_{E} \|E\|_F, \qquad (A+E)x = b, \tag{8.7.3}$$

where it is assumed that the errors in the data is confined to the matrix A.

In the following we assume that $b \notin \mathcal{R}(A)$, for otherwise the system is consistent. The constraint in (8.7.2) can be written

$$\begin{pmatrix} b + r & A + E \end{pmatrix} \begin{pmatrix} -1 \\ x \end{pmatrix} = 0.$$

This constraint is satisfied if the matrix $(b+r\ A+E)$ is rank deficient and $\begin{pmatrix} -1 & x \end{pmatrix}^T$ lies in its null space. Hence, the TLS problem involves finding a perturbation matrix having minimal Frobenius norm, which lowers the rank of the matrix $\begin{pmatrix} b & A \end{pmatrix}$.

The total least squares problem can be analyzed in terms of the SVD

$$\begin{pmatrix} b & A \end{pmatrix} = U\Sigma V^T = \sum_{i=1}^{k+1} \sigma_i u_i v_i^T, \tag{8.7.4}$$

where $\sigma_1 \geq \ldots \geq \sigma_n \geq \sigma_{n+1} \geq 0$ are the singular values of $(\, b \quad A\,)$. By the minimax characterization of singular values (Theorem 8.1.15) the singular values of $\hat{\sigma}_i$ of A interlace those of $(\, b \quad A\,)$, that is

$$\sigma_1 \geq \hat{\sigma}_1 \geq \sigma_2 > \cdots \geq \sigma_n \geq \hat{\sigma}_n \geq \sigma_{n+1}. \tag{8.7.5}$$

Assume first that $\hat{\sigma}_n > \sigma_{n+1}$. Then it follows that $\operatorname{rank}(A) = n$ and by (8.7.5) $\sigma_n > \sigma_{n+1}$. If $\sigma_{n+1} = 0$, then $Ax = b$ is consistent; otherwise by Theorem 8.1.16 the unique perturbation of minimum norm $\|\,(\, r \quad E\,)\,\|_F$ that makes $(A+E)x = b+r$ consistent is the rank one perturbation

$$(\, r \quad E\,) = -\sigma_{n+1} u_{n+1} v_{n+1}^T \tag{8.7.6}$$

for which $\min_{E,\,r} \|\,(\, r \quad E\,)\,\|_F = \sigma_{n+1}$. Multiplying (8.7.6) from the right with v_{n+1} gives

$$(\, b \quad A\,)\, v_{n+1} = -\,(\, r \quad E\,)\, v_{n+1}. \tag{8.7.7}$$

Writing the relation $(A+E)x = b+r$ in the form

$$(\, b \quad A\,) \begin{pmatrix} 1 \\ -x \end{pmatrix} = -\,(\, r \quad E\,) \begin{pmatrix} 1 \\ -x \end{pmatrix}$$

and comparing with (8.7.7) it is easily seen that the TLS solution can be written in terms of the right singular vector v_{n+1} as

$$x = -\frac{1}{\omega} y, \quad v_{n+1} = \begin{pmatrix} \omega \\ y \end{pmatrix}, \tag{8.7.8}$$

If $\omega = 0$ then the TLS problem has no solution. From (8.7.4) it follows that $(\, b \quad A\,)^T U = V \Sigma^T$ and taking the $(n+1)$st column of both sides

$$\begin{pmatrix} b^T \\ A^T \end{pmatrix} u_{n+1} = \sigma_{n+1} v_{n+1}. \tag{8.7.9}$$

Hence, if $\sigma_{n+1} > 0$, then $\omega = 0$ if and only if $b \perp u_{n+1}$. (This case can only occur when $\hat{\sigma}_n = \sigma_{n+1}$, since otherwise the TLS problem has a unique solution.) The case when $b \perp u_{n+1}$is called **nongeneric**. It can be treated by adding constraints on the solution; see the discussion [370].

Example 8.7.1.

For

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \qquad E = \begin{pmatrix} 0 & 0 \\ 0 & \epsilon \\ 0 & 0 \end{pmatrix}, \tag{8.7.10}$$

the system $(A+E)x = b$ is consistent for any $\epsilon > 0$. There is no smallest value of ϵ and $\|x\|_2 \to \infty$ when $\epsilon \to 0$ and the TLS problem fails to have a finite solution. Here A is singular, $\hat{\sigma}_2 = \sigma_3 = 0$ and $b \perp u_3 = e_3$.

Suppose now that σ_{n+1} is a repeated singular value,

$$\sigma_p \ > \sigma_{p+1} = \cdots = \sigma_{n+1}, \quad p < n.$$

Set $V_2 = (v_{p+1}, \dots, v_{n+1})$, and let $V_2 z$ be any unit vector in the subspace spanned by the right singular vectors corresponding to the multiple minimal singular value. Then any vector such that

$$x = -\frac{1}{\omega} y, \quad V_2 z = \begin{pmatrix} \omega \\ y \end{pmatrix},$$

is a TLS solution. A unique TLS solution of minimum norm can be obtained as follows. Since $V_2 z$ has unit length, minimizing $\|x\|_2$ is equivalent to choosing the unit vector z to maximize $\omega = e_1^T V_2 z$. Let take $z = Qe_1$, where Q is a Householder transformation such that

$$V_2 Q = \begin{pmatrix} \omega & 0 \\ y & V_2' \end{pmatrix}$$

Then a TLS solution of minimum norm is given by (8.7.8). If $\omega \neq 0$ there is no solution and the problem is nongeneric. By an argument similar to the case when $p = n$ this can only happen if $b \perp u_j$, $j = p : n$.

8.7.2 Conditioning of the TLS Problem

We now consider the conditioning of the total least squares problem and its relation to the least squares problem. We denote those solutions by x_{TLS} and x_{LS}, respectively.

In Sec. 7.1.6 we showed that the SVD of a matrix A is related to the eigenvalue problem for the symmetric matrix $A^T A$. From this it follows that in the generic case the TLS solution can also be characterized by

$$\begin{pmatrix} b^T \\ A^T \end{pmatrix} \begin{pmatrix} b & A \end{pmatrix} \begin{pmatrix} -1 \\ x \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} -1 \\ x \end{pmatrix}, \tag{8.7.11}$$

i.e. $\begin{pmatrix} -1 \\ x \end{pmatrix}$ is an eigenvector corresponding to the smallest eigenvalue $\lambda_{n+1} = \sigma_{n+1}^2$ of the matrix obtained by "squaring" $\begin{pmatrix} b & A \end{pmatrix}$. From the properties of the Rayleigh quotient of symmetric matrices (see Sec. 9.3.4) it follows that x_{TLS} is characterized by minimizing

$$\rho(x) = \frac{(b - Ax)^T (b - Ax)}{x^T x + 1} = \frac{\|b - Ax\|_2^2}{\|x\|_2^2 + 1}, \tag{8.7.12}$$

Thus, whereas the LS solution minimizes $\|b - Ax\|_2^2$ the TLS solution minimizes the "orthogonal distance" function $\rho(x)$ in (8.7.11).

From the last block row of (8.7.11) it follows that

$$(A^T A - \sigma_{n+1}^2 I) x_{TLS} = A^T b. \tag{8.7.13}$$

Note that if $\hat{\sigma}_n > \sigma_{n+1}$ then the matrix $(A^TA - \sigma_{n+1}^2 I)$ is symmetric positive definite, which ensures that the TLS problem has a unique solution. This can be compared with the corresponding normal equations for the least squares solution

$$A^TAx_{LS} = A^Tb. \tag{8.7.14}$$

In (8.7.13) a positive multiple of the unit matrix is *subtracted* from the matrix A^TA of normal equations. Thus, TLS can be considered as a *deregularizing* procedure. (Compare Sec. 8.4.2, where a multiple of the unit matrix was added to improve the conditioning.) Hence, the TLS solution is always *worse conditioned* than the LS problem, From a statistical point of view this can be interpreted as removing the bias by subtracting the error covariance matrix (estimated by $\sigma_{n+1}^2 I$ from the data covariance matrix A^TA. Subtracting (8.7.14) from (8.7.14) we get

$$x_{TLS} - x_{LS} = \sigma_{n+1}^2 (A^TA - \sigma_{n+1}^2 I)^{-1} x_{LS}.$$

Taking norms we obtain

$$\frac{\|x_{TLS} - x_{LS}\|_2}{\|x_{LS}\|_2} \leq \frac{\sigma_{n+1}^2}{\hat{\sigma}_n^2 - \sigma_{n+1}^2}.$$

It can be shown that an approximate condition number for the TLS solution is

$$\kappa_{TLS} \approx \frac{\hat{\sigma}_1}{\hat{\sigma}_n - \sigma_{n+1}} = \kappa(A) \frac{\hat{\sigma}_n}{\hat{\sigma}_n - \sigma_{n+1}}. \tag{8.7.15}$$

When $\hat{\sigma}_n - \sigma_{n+1} \ll \hat{\sigma}_n$ the TLS condition number can be much worse than for the LS problem.

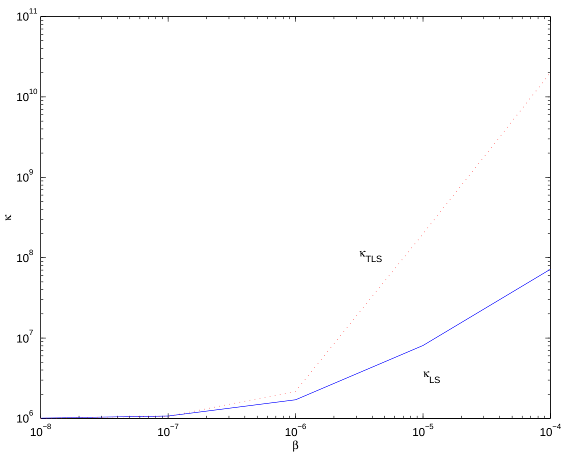

Figure 8.7.1. *Condition numbers κ_{LS} and κ_{TLS} as function of $\beta = \|r_{\rm LS}\|_2$.*

Example 8.7.2.

Consider the overdetermined system

$$\begin{pmatrix} \hat{\sigma}_1 & 0 \\ 0 & \hat{\sigma}_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \beta \end{pmatrix}. \tag{8.7.16}$$

Trivially, the LS solution is $x_{\text{LS}} = (c_1/\hat{\sigma}_1, c_2/\hat{\sigma}_2)^T$, $\|r_{\text{LS}}\|_2 = |\beta|$. If we take $\hat{\sigma}_1 = c_1 = 1$, $\hat{\sigma}_2 = c_2 = 10^{-6}$, then $x_{\text{LS}} = (1,\ 1)^T$ is independent of β, and hence does not reflect the ill-conditioning of A. However,

$$\kappa_{LS}(A, b) = \kappa(A)\left(1 + \frac{\|r_{\text{LS}}\|_2}{\|\hat{\sigma}_1 x_{\text{LS}}\|_2}\right)$$

will increase proportionally to β. The TLS solution is of similar size as the LS solution as long as $|\beta| \le \hat{\sigma}_2$. However, when $|\beta| \gg \hat{\sigma}_2$ then $\|x_{\text{TLS}}\|_2$ becomes large.

In Figure 8.7.1 the two condition numbers are plotted as a function of $\beta \in [10^{-8}, 10^{-4}]$. For $\beta > \hat{\sigma}_2$ the condition number κ_{TLS} grows proportionally to β^2. It can be verified that $\|x_{\text{TLS}}\|_2$ also grows proportionally to β^2.

Setting $c_1 = c_2 = 0$ gives $x_{LS} = 0$. If $|\beta| \ge \sigma_2(A)$, then $\sigma_2(A) = \sigma_3(A, b)$ and the TLS problem is nongeneric.

8.7.3 Some Generalized TLS Problems

We now consider the more general TLS problem with $d > 1$ right-hand sides

$$\min_{E,\ F} \|\,(\,E \quad F\,)\,\|_F, \qquad (A + E)X = B + F, \tag{8.7.17}$$

where $B \in \mathbf{R}^{m\times d}$. The consistency relations can be written

$$(\,B + F \quad A + E\,)\begin{pmatrix} -I_d \\ X \end{pmatrix} = 0,$$

Thus, we now seek perturbations $(E,\ F)$ that reduces the rank of the matrix $(\,B \quad A\,)$ by d. We call this a **multidimensional** TLS problem. As remarked before, for this problem to be meaningful the rows of the error matrix $(\,B + F \quad A + E\,)$ should be independently and identically distributed with zero mean and the same variance.

In contrast to the usual least squares problem, the multidimensional TLS problem is different from separately solving d one-dimensional TLS problems with right-hand sides $b_1, \dots, b_d$. This is because in the multidimensional problem we require that *the matrix A be similarly perturbed for all right-hand sides.* This should give an improved predicted power of the TLS solution.

The solution to the TLS problem with multiple right-hand sides can be expressed in terms of the SVD

$$(\,B \quad A\,) = U\Sigma V^T = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T, \tag{8.7.18}$$

where

$$\Sigma_1 = \operatorname{diag}(\sigma_1, \dots, \sigma_n), \quad \Sigma_2 = \operatorname{diag}(\sigma_{n+1}, \dots, \sigma_{n+d}),$$

and U and V partitioned conformally with $(\, B \quad A \,)$. Assuming that $\sigma_n > \sigma_{n+1}$, the minimizing perturbation is unique and given by the rank d matrix

$$(\, F \quad E \,) = -U_2\Sigma_2 V_2^T = -(\, B \quad A \,) V_2 V_2^T,$$

for which $\| (\, F \quad E \,) \|_F = \sum_{j=1}^d \sigma_{n+j}^2$ and $(\, B + F \quad A + E \,) V_2 = 0$. Assume that

$$V_2 = \begin{pmatrix} V_{12} \\ V_{22} \end{pmatrix}.$$

with $V_{12} \in \mathbf{R}^{d\times d}$ nonsingular. Then the solution to the TLS problem is unique and given by

$$X = -V_{22}V_{12}^{-1} \in \mathbf{R}^{n\times d}.$$

We show that if $\sigma_n(A) > \sigma_{n+1}(\, B \quad A \,)$, then V_{12} is nonsingular. From (8.7.18) it follows that $BV_{12} + AV_{22} = U_2\Sigma_2$. Now, suppose that V_{12} is singular. Then $V_{12}x = 0$ for some unit vector x. It follows that $U_2\Sigma_2 x = AV_{12}x$. From $V_2^T V_2 = V_{12}^T V_{12} + V_{22}^T V_{22} = I$ it follows that $V_{22}^T V_{22} x = x$ and hence $\|V_{22}x\|_2 = 1$. But then

$$\sigma_{n+1}(\, B \quad A \,) \ge \|U_2\Sigma_2 x\|_2 = \|AV_{12}x\|_2 \ge \sigma_n(A),$$

a contradiction. Hence, V_{12} is nonsingular.

From the above characterization it follows that the TLS solution satisfies

$$\begin{pmatrix} B^T \\ A^T \end{pmatrix} (\, B \quad A \,) \begin{pmatrix} -I_d \\ -X \end{pmatrix} = \begin{pmatrix} -I_d \\ X \end{pmatrix} C, \tag{8.7.19}$$

where

$$C = V_{22}\Sigma_2^2 V_{22}^{-1} \in \mathcal{R}^{d\times d}. \tag{8.7.20}$$

Note that C is symmetrizable but not symmetric! Multiplying (8.7.19) from the left with $(I_d \; X^T)$ gives

$$(B - AX)^T(B - AX) = (X^T X + I_d)C,$$

and if $(X^T X + I_d)$ is nonsingular,

$$C = (X^T X + I_d)^{-1}(B - AX)^T(B - AX), \tag{8.7.21}$$

The multidimensional TLS solution X_{TLS} minimizes $\|C\|_F$, which generalizes the result for $d = 1$.

The last block component of (8.7.19) reads

$$A^T AX - XC = A^T B,$$

which is a Sylvester equation for X. This has a unique solution if and only if $A^T A$ and C have no common eigenvalues. which is the case if $\hat{\sigma}_n > \sigma_{n+1}$.

Now assume that $\sigma_k > \sigma_{k+1} = \cdots = \sigma_{n+1}$, $k < n$, and set $V_2 = (v_{k+1}, \ldots, v_{n+d})$. Let Q be a product of Householder transformations such that

$$V_2 Q = \begin{pmatrix} \Gamma & 0 \\ Z & Y \end{pmatrix},$$

where $\Gamma \in \mathbf{R}^{d\times d}$ is lower triangular. If Γ is nonsingular, then the TLS solution of minimum norm is given by

$$X = -Z\Gamma^{-1}.$$

In many parameter estimation problems, some of the columns are known exactly. It is no restriction to assume that the error-free columns are in leading positions in A. In the multivariate version of this **mixed LS-TLS problem** one has a linear relation

$$(A_1, A_2 + E_2)X = B + F, \qquad A_1 \in \mathbf{R}^{m\times n_1},$$

where $A = (A_1,\ A_2) \in \mathbf{R}^{m\times n}$, $n = n_1 + n_2$. It is assumed that the rows of the errors $(E_2,\ F)$ are independently and identically distributed with zero mean and the same variance. The mixed LS-TLS problem can then be expressed

$$\min_{E_2,F} \|(E_2,\ F)\|_F, \qquad (A_1,\ A_2 + E_2)X = B + F. \tag{8.7.22}$$

When A_2 is empty, this reduces to solving an ordinary least squares problem. When A_1 is empty this is the standard TLS problem. Hence, this mixed problem includes both extreme cases.

The solution of the mixed LS-TLS problem can be obtained by first computing a QR factorization of A and then solving a TLS problem of reduced dimension.

Algorithm 8.13.
Mixed LS-TLS problem Let $A = (A_1,\ A_2) \in \mathbf{R}^{m\times n}$, $n = n_1 + n_2$, $m \geq n$, and $B \in \mathbf{R}^{m\times d}$. Assume that the columns of A_1 are linearly independent. Then the following algorithm solves the mixed LS-TLS problem (8.7.22).

Step 1. Compute the QR factorization

$$(A_1,\ A_2,\ B) = Q\begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

where Q is orthogonal, and $R_{11} \in \mathbf{R}^{n_1\times n_1}$, $R_{22} \in \mathbf{R}^{(n_2+d)\times(n_2+d)}$ are upper triangular. If $n_1 = n$, then the solution X is obtained by solving $R_{11}X = R_{12}$ (usual least squares); otherwise continue (solve a reduced TLS problem).

Step 2. Compute the SVD of R_{22}

$$R_{22} = U\Sigma V^T, \qquad \Sigma = \mathrm{diag}\,(\sigma_1, \ldots, \sigma_{n_2+d}),$$

where the singular values are ordered in decreasing order of magnitude.

Step 3a. Determine $k \leq n_2$ such that

$$\sigma_k > \sigma_{k+1} = \cdots = \sigma_{n_2+d} = 0,$$

and set $V_{22} = (v_{k+1}, \dots, v_{n_2+d})$. If $n_1 > 0$ then compute V_2 by back-substitution from

$$R_{11}V_{12} = -R_{12}V_{22}, \qquad V_2 = \begin{pmatrix} V_{12} \\ V_{22} \end{pmatrix},$$

else set $V_2 = V_{22}$.

Step 3b. Perform Householder transformations such that

$$V_2 Q = \begin{pmatrix} \Gamma & 0 \\ Z & Y \end{pmatrix},$$

where $\Gamma \in \mathbf{R}^{d\times d}$ is upper triangular. If Γ is nonsingular then the solution is

$$X = -Z\Gamma^{-1}.$$

Otherwise the TLS problem is nongeneric and has no solution.

Note that the QR factorization in the first step would be the first step in computing the SVD of A.

8.7.4 Bidiagonalization and TLS Problems.

One way to avoid the complications of nongeneric problems is to compute a regular core TLS problem by bidiagonalizing of the matrix $(\,b \quad A\,)$. Consider the TLS problem

$$\min_{E,r} \|(E,\ r)\|_F, \qquad (A+E)x = b + r.$$

It was shown in Sec. 8.4.6 that we can always find square orthogonal matrices $\tilde{U}_{k+1}$ and $\tilde{V}_k = P_1 P_2 \cdots P_k$, such that

$$\tilde{U}_{k+1}^T (\,b \quad A\tilde{V}_k\,) = \begin{pmatrix} \beta_1 e_1 & B_k & 0 \\ 0 & 0 & A_k \end{pmatrix}, \tag{8.7.23}$$

where

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} \in \mathbf{R}^{(k+1)\times k},$$

and

$$\beta_j \alpha_j \neq 0, \quad j = 1:k. \tag{8.7.24}$$

Setting $x = \tilde{V}_k \begin{pmatrix} y \\ z \end{pmatrix}$, the approximation problem $Ax \approx b$ then decomposes into the two subproblems

$$B_k y \approx \beta_1 e_1, \qquad A_k z \approx 0.$$

It seems reasonable to simply take $z = 0$, and separately solve the first subproblem, which is the minimally dimensioned **core subproblem**. Setting

$$V_k = \tilde{V}_k \begin{pmatrix} I_k \\ 0 \end{pmatrix}, \qquad U_{k+1} = \tilde{U}_{k+1} \begin{pmatrix} I_{k+1} \\ 0 \end{pmatrix},$$

it follows that

$$(b \quad AV_k) = U_{k+1} (\beta_1 e_1 \quad B_k).$$

If $x = V_k y \in \mathcal{R}(V_k)$ then

$$(A + E)x = (A + E)V_k y = (U_{k+1} B_k + EV_k)y = \beta_1 U_{k+1} e_1 + r,$$

Hence, the consistency relation $(A + E_k)x = b + r$ becomes

$$(B_k + F)y = \beta_1 e_1 + s, \quad F = U_{k+1}^T E V_k, \quad s = U_{k+1}^T r. \tag{8.7.25}$$

Using the orthogonality of U_{k+1} and V_k it follows that

$$\|(E, r)\|_F = \|(F, s)\|_F. \tag{8.7.26}$$

Hence, to minimize $\|(E, r)\|_F$ we should take y_k to be the solution to the TLS core subproblem

$$\min_{F,s} \|(F,\ s)\|_F, \qquad (B_k + F)y = \beta_1 e_1 + s. \tag{8.7.27}$$

From (8.7.24) and Theorem 8.4.2 it follows that the singular values of the matrix B_k are simple and that the right hand side βe_1 has nonzero components along each left singular vector. This TLS problem therefore must have a unique solution. Note that we can assume that $\beta_{k+1} \neq 0$, since otherwise the system is compatible.

To solve this subproblem we need to compute the SVD of the bidiagonal matrix

$$(\beta_1 e_1,\ B_k) = \begin{pmatrix} \beta_1 & \alpha_1 & & & \\ & \beta_2 & \alpha_2 & & \\ & & \beta_3 & \ddots & \\ & & & \ddots & \alpha_k \\ & & & & \beta_{k+1} \end{pmatrix} \in \mathbf{R}^{(k+1)\times(k+1)}. \tag{8.7.28}$$

The SVD of this matrix

$$(\beta_1 e_1,\ B_k) = P \mathrm{diag}\,(\sigma_1, \ldots, \sigma_{k+1}) Q^T, \quad P, Q \in \mathbf{R}^{(k+1)\times(k+1)}$$

can be computed, e.g., by the implicit QR-SVD algorithm; see Sec. 9.7.6. (Note that the first stage in this is a transformation to bidiagonal form, so the work in performing the reduction (8.7.23) has not been wasted!) Then with

$$q_{k+1} = Q e_{k+1} = \begin{pmatrix} \omega \\ z \end{pmatrix}.$$

Here it is always the case that $\omega \neq 0$ and the solution to the original TLS problem (8.7.27) equals

$$x_{TLS} = V_k y = -\omega^{-1} V_k z.$$

Further, the norm of the perturbation equals

$$\min_{E,r} \|(E, r)\|_F = \sigma_{k+1}.$$

8.7.5 Iteratively Reweighted Least Squares.

In some applications it might be more adequate to solve the problem

$$\min \|Ax - b\|_p \tag{8.7.29}$$

for some l_p-norm with $p \neq 2$. For $p = 1$ the solution may not be unique, while for $1 < p < \infty$ the problem (8.7.29) is strictly convex and hence has exactly one solution. Minimization in the l_1-norm or l_∞-norm is more complicated since the function $f(x) = \|Ax - b\|_p$ is not differentiable for $p = 1, \infty$.

Example 8.7.3. *To illustrate the effect of using a different norm we consider the problem of estimating the scalar x from m observations $b \in \mathbf{R}^m$. This is equivalent to minimizing $\|Ax - b\|_p$, with $A = e = (1, 1, \dots, 1)^T$. It is easily verified that if $b_1 \geq b_2 \geq \dots \geq b_m$, then the solution x_p for some different values p are*

$$\begin{aligned} x_1 &= b_{\frac{m+1}{2}}, \quad (m \text{ odd}) \\ x_2 &= \frac{1}{m}(b_1 + b_2 + \dots + b_m), \\ x_\infty &= \frac{1}{2}(b_1 + b_m). \end{aligned}$$

These estimates correspond to the median, mean, and midrange respectively. Note that the estimate x_1 is insensitive to the extreme values of b_i, while x_∞ only *depends on the extreme values. The l_∞ solution has the property that the absolute error in at least n equations equals the maximum error.*

The simple example above shows that the l_1 norm of the residual vector has the advantage of giving a solution that is **robust**, i.e., a small number of isolated large errors will usually not change the solution much. A similar effect is also achieved with p greater than but close to 1.

For solving the l_p norm problem when $1 < p < 3$, the **iteratively reweighted least squares** (IRLS) method (see Osborne [295, 1985]) can be used to reduce the problem to a sequence of weighted least squares problems.

We start by noting that, provided that $|r_i(x)| = |b - Ax|_i > 0$, $i = 1, \dots, m$, the problem (8.7.29) can be restated in the form $\min_x \psi(x)$, where

$$\psi(x) = \sum_{i=1}^{m} |r_i(x)|^p = \sum_{i=1}^{m} |r_i(x)|^{p-2} r_i(x)^2. \tag{8.7.30}$$

This can be interpreted as a weighted least squares problem

$$\min_x \|D(r)^{(p-2)/2}(b - Ax)\|_2, \qquad D(r) = \mathrm{diag}\,(|r|), \tag{8.7.31}$$

where diag $(|r|)$ denotes the diagonal matrix with ith component $|r_i|$.

The diagonal weight matrix $D(r)^{(p-2)/2}$ in (8.7.31) depends on the unknown solution x, but we can attempt to use the following iterative method.

Algorithm 8.14.

IRLS for l_p Approximation $1 < p < 2$

Let $x^{(0)}$ be an initial approximation such that $r_i^{(0)} = (b - Ax^{(0)})_i \neq 0$, $i = 1, \dots, n$.

for $k = 0, 1, 2, \dots$

$\quad r_i^{(k)} = (b - Ax^{(k)})_i$;

$\quad D_k = \mathrm{diag}\,\big((|r_i^{(k)}|)^{(p-2)/2}\big)$;

$\quad$ solve $\delta x^{(k)}$ from

$$\min_{\delta x} \big\|D_k(r^{(k)} - A\delta x)\big\|_2;$$

$\quad x^{(k+1)} = x^{(k)} + \delta x^{(k)}$;

end

Since $D_k b = D_k(r^{(k)} - Ax^{(k)})$, it follows that $x^{(k+1)}$ in IRLS solves $\min_x \|D_k(b - Ax)\|_2$, but the implementation above is to be preferred. It has been assumed that in the IRLS algorithm, at each iteration $r_i^{(k)} \neq 0$, $i = 1, \dots, n$. In practice this cannot be guaranteed, and it is customary to modify the algorithm so that

$$D_k = \mathrm{diag}\,\big((100ue + |r^{(k)}|)^{(p-2)/2}\big),$$

where u is the machine precision and $e^T = (1, \dots, 1)$ is the vector of all ones. Because the weight matrix D_k is not constant, the simplest implementations of IRLS recompute, e.g., the QR factorization of $D_k A$ in each step. It should be pointed out that the iterations can be carried out entirely in the r space without the x variables. Upon convergence to a residual vector r_{opt} the corresponding solution can be found by solving the consistent linear system $Ax = b - r_{\mathrm{opt}}$.

It can be shown that in the l_p case any fixed point of the IRLS iteration satisfies the necessary conditions for a minimum of $\psi(x)$. The IRLS method is convergent for $1 < p < 3$, and also for $p = 1$ provided that the l_1 approximation problem has a unique nondegenerate solution. However, the IRLS method can be extremely slow when p is close to unity.

Review Questions

8.1 Formulate the total least squares (TLS) problem. The solution of the TLS problem is related to a theorem on matrix approximation. Which?

Problems and Computer Exercises

8.1 Consider a TLS problem where $n = 1$ and

$$C = (A, b) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

Show that the unique minimizing ΔC gives

$$C + \Delta C = (A + E, b + r) = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$$

so the perturbed system is not compatible, but that an arbitrary small perturbation ϵ in the (2,1) element will give a compatible system with solution $x = 2/\epsilon$.

8.2 (a) Let $A \in \mathbf{R}^{m\times n}$, $m \geq n$, $b \in \mathbf{R}^m$, and consider the total least squares (TLS) problem. $\min_{E,r} \|(E, r)\|_F$, where $(A+E)x = b+r$. If we have the QR factorization

$$Q^T(A,\ b) = \begin{pmatrix} S \\ 0 \end{pmatrix}, \qquad S = \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}.$$

then the ordinary least squares solution is $x_{LS} = R^{-1}z$, $\|r\|_2 = \rho$.
Show that if a TLS solution x_{TLS} exists, then it holds

$$\begin{pmatrix} R^T & 0 \\ z^T & \rho \end{pmatrix} \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix} \begin{pmatrix} x_{TLS} \\ -1 \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} x_{TLS} \\ -1 \end{pmatrix},$$

where σ_{n+1} is the smallest singular value of $(A,\ b)$.

(b) Write a program using inverse iteration to compute x_{TLS}, i.e., for $k = 0, 1, 2, \ldots$, compute a sequence of vectors $x^{(k+1)}$ by

$$\begin{pmatrix} R^T & 0 \\ z^T & \rho \end{pmatrix} \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix} \begin{pmatrix} y^{(k+1)} \\ -\alpha \end{pmatrix} = \begin{pmatrix} x^{(k)} \\ -1 \end{pmatrix}, \qquad x^{(k+1)} = y^{(k+1)}/\alpha.$$

As starting vector use $x^{(0)} = x_{LS}$ on the assumption that x_{TLS} is a good approximation to x_{LS}. Will the above iteration always converge? Try to make it fail!

(c) Study the effect of scaling the right hand side in the TLS problem by making the substitution $z := \theta z$, $\rho := \theta\rho$. Plot $\|x_{TLS}(\theta) - x_{LS}\|_2$ as a function of θ and verify that when $\theta \to 0$, then $x_{TLS} \to x_{LS}$.

Hint For generating test problems it is suggested that you use the function qmult(A) from the MATLAB collection of test matrices by N. Higham to generate a matrix $C = (A,\ b) = Q_1 * D * Q_2^T$, where Q_1 and Q_2 are random real orthogonal matrices and D a given diagonal matrix. This allows you to generate problems where C has known singular values and vectors.

Notes and Further References

Modern numerical methods for solving least squares problems are surveyed in the two comprehensive monographs [257] and [40]. The latter contains a bibliography of 860 references, indicating the considerable research interest in these problems. Hansen [200] gives an excellent survey of numerical methods for the treatment of numerically rank deficient linear systems arising, for example, from discrete ill-posed problems.

Several of the great mathematicians at the turn of the 19th century worked on methods for solving overdetermined linear systems. Laplace in 1799 used the principle of minimizing the sum of absolute errors $|r_i|$, with the added conditions that the errors sum to zero. This leads to a solution x that satisfies at least n equations exactly. The method of least squares was first published as an algebraic procedure by Legendre 1805 in [260]. Gauss justified the least squares principle as a statistical procedure in [151], where he claimed to have used the method since 1795. This led to one of the most famous priority dispute in the history of mathematics. Gauss further developed the statistical aspects in 1821–1823. For an interesting accounts of the history of the invention of least squares, see Stiegler [350, 1981].

Because of its success in analyzing astronomical data the method of least squares rapidly became the method of choice when analyzing observation. Geodetic calculations was another early area of application of the least squares principle. In the last decade applications in control and signal processing has been a source of inspiration for developments in least squares calculations.

Section 8.1

The singular value decomposition was independently developed by E. Beltrami 1873 and C. Jordan 1874; see G. W. Stewart [344, 1993] for an interesting account of the early history of the SVD. The first stable algorithm for computing the SVD the singular value was developed by Golub, Reinsch, and Wilkinson in the late 1960's. Several other applications of the SVD to matrix approximation can be found in Golub and Van Loan [184, Sec. 12.4].

Complex Givens rotations and complex Householder transformations are treated in detail by Wilkinson [387, pp. 47–50]. Lehoucq [261, 1996] gives a comparison of different implementations of complex Householder transformations. The reliable construction of real and complex Givens rotations are considered in great detail in Bindel, Demmel and Kahan [34].

Theorem 8.1.3 can be generalized to to the semi-definite case, see Gulliksson and Wedin [192, Theorem 3.2]. A case when B is indefinite and nonsingular is considered in Sec. 8.6.2

The modern formulation of the pseudoinverse is due to Moore [283, 1920], Bjerhammar [36, 1951] and Penrose [307, 1953]. A good introduction to generalized inverses is given by Ben-Israel and Greville [28, 1976]. Generalized inverses should be used with caution since the notation tends to hide intrinsic computational difficulties associated with rank deficient matrices. A more complete and thorough treatment is given in the monograph by the same authors [29, 2003]. The use of generalized inverses in geodetic calculations is treated in Bjerhammar [37, 1973].

Section 8.2

Peters and Wilkinson [309, 1970] developed methods based on Gaussian elimination from a uniform standpoint and the excellent survey by Noble [291, 1976]. Sautter [330, 1978] gives a detailed analysis of stability and rounding errors of the LU algorithm for computing pseudo-inverse solutions.

How to find the optimal backward error for the linear least squares problem was an open problem for many years, until it was elegantly answered by Karlsson et al. [378]; see also [231]. Gu [187] gives several approximations to that are optimal up to a factor less than 2. Optimal backward perturbation bounds for underdetermined systems are derived in [352]. The extension of backward error bounds to the case of constrained least squares problems is discussed by Cox and Higham [78].

Section 8.3

The different computational variants of Gram–Schmidt have an interesting history. The "modified" Gram–Schmidt (MGS) algorithm was in fact already derived by Laplace in 1816 as an elimination method using weighted row sums. Laplace did not interpret his algorithm in terms of orthogonalization, nor did he use it for computing least squares solutions! Bienaymé in 1853 gave a similar derivation of a slightly more general algorithm; see Björck [39, 1994]. What is now called the "classical" Gram–Schmidt (CGS) algorithm first appeared explicitly in papers by Gram 1883 and Schmidt 1908. Schmidt treats the solution of linear systems with infinitely many unknowns and uses the orthogonalization as a theoretical tool rather than a computational procedure.

In the 1950's algorithms based on Gram–Schmidt orthogonalization were frequently used, although their numerical properties were not well understood at the time. Björck [38] analyzed the modified Gram–Schmidt algorithm and showed its stability for solving linear least squares problems.

The systematic use of orthogonal transformations to reduce matrices to simpler form was initiated by Givens [167] and Householder [218, 1958]. The application of these transformations to linear least squares is due to Golub [169, 1965], where it was shown how to compute a QR factorization of A using Householder transformations.

Section 8.4

A different approach to the subset selection problem has been given by de Hoog and Mattheij [90]. They consider choosing the square subset A_1 of rows (or columns), which maximizes $det(A_1)$.

Section 8.5

The QR algorithm for banded rectangular matrices was first given by Reid [318]. Rank-revealing QR (RRQR) decompositions have been studied by a number of authors. A good survey can be found in Hansen [200]. The URV and ULV decompositions were introduced by Stewart [343, 345].

Section 8.6

An early reference to the exchange operator is in network analysis; see the survey of Tsatsomeros [363]. J-orthogonal matrices also play a role in the solution of the generalized eigenvalue problem $Ax = \lambda Bx$; see Sec. 9.8. For a systematic study of

J-orthogonal matrices and their many applications we refer to Higham [212]. An error analysis of Chamber's algorithm is given by Bojanczyk et al. [46].

The systematic use of GQR as a basic conceptual and computational tool are explored by [299]. These generalized decompositions and their applications are discussed in [8]. Algorithms for computing the bidiagonal decomposition are due to Golub and Kahan [173, 1965]. The partial least squares (PLS) method, which has become a standard tool in chemometrics, goes back to Wold et al. [391].

The term "total least squares problem", which was coined by Golub and Van Loan [183], renewed the interest in the "errors in variable model". A thorough and rigorous treatment of the TLS problem is found in Van Huffel and Vandewalle [370]. The important role of the core problem for weighted TLS problems was discovered by Paige and Strakoš [303].

Chapter 9 of NUMERICAL METHODS AND COMPUTATION

Chapter 9

Matrix Eigenvalue Problems

The eigenvalue problem has a deceptively simple formulation, yet the determination of accurate solutions presents a wide variety of challenging problems.
—J. H. Wilkinson, The Algebraic Eigenvalue problem, 1965

9.1 Basic Properties

9.1.1 Introduction

Of central importance in the study of matrices $A \in \mathbf{C}^{n\times n}$ are the special vectors whose directions are not changed when multiplied by A. If

$$Ax = \lambda x, \quad x \neq 0, \tag{9.1.1}$$

the complex scalar λ is called an **eigenvalue** of A and x is an **eigenvector** of A. When an eigenvalue λ is known, the corresponding eigenvector(s) is obtained by solving the linear homogeneous system

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

Thus, λ is an eigenvalue of A only if $A - \lambda I$ is a singular matrix. Clearly, an eigenvector x is only determined up to a multiplicative constant $\alpha \neq 0$.

Eigenvalues and eigenvectors are a standard tool in the mathematical sciences and in scientific computing. Eigenvalues give information about the behavior of evolving systems governed by a matrix or operator. The problem of computing eigenvalues and eigenvectors of a matrix occurs in many settings in physics and engineering. Eigenvalues are useful in analyzing resonance, instability, and rates of growth or decay with applications to, e.g., vibrating systems, airplane wings, ships, buildings, bridges and molecules. Eigenvalue decompositions play also an important part in the analysis of many numerical methods. Further, singular values are closely related to an eigenvalues a symmetric matrix.

In this chapter we treat numerical methods for computing eigenvalues and eigenvectors of matrices. In the first three sections we briefly review the classical theory needed for the proper understanding of the numerical methods treated in the later sections. Sec. 9.1 gives a brief account of basic facts of the matrix eigenvalue problem, and the theory of canonical forms and matrix functions. Sec. 9.2 is devoted to the localization of eigenvalues and perturbation results for eigenvalues and eigenvectors.

Sec. 9.5 treats the Jacobi methods for the real symmetric eigenvalue problem and the SVD. These methods have advantages for parallel implementation and are potentially very accurate. The power method and its modifications are treated in Sec. 9.3. Transformation to condensed form described in Sec. 9.3 often is a preliminary step in solving the eigenvalue problem. Followed by the QR algorithm this constitutes the current method of choice for computing eigenvalues and eigenvectors of small to medium size matrices, see Sec. 9.4. This method can also be adopted to compute singular values and singular vectors although the numerical implementation is often far from trivial, see Sec. 9.4.

In Sec. 9.7 we briefly discuss some methods for solving the eigenvalue problem for large sparse matrices. Finally, in Sec. 9.8 we consider the generalized eigenvalue problem $Ax = \lambda Bx$, and the generalized SVD.

9.1.2 Theoretical Background

It follows from (9.1.1) that λ is an eigenvalue of A if and only if the system $(A - \lambda I)x = 0$ has a nontrivial solution $x \neq 0$, or equivalently if and only if the matrix $A - \lambda I$ is singular. Hence, the eigenvalues satisfy the **characteristic equation**

$$p_n(\lambda) = \det(A - \lambda I) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \ldots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0 \qquad (9.1.2)$$

The set $\lambda(A) = \{\lambda_i\}_{i=1}^n$ of all eigenvalues of A is called the **spectrum**[30] of A. The polynomial $p_n(\lambda) = \det(A - \lambda I)$ is called the **characteristic polynomial** of the matrix A. Expanding the determinant in (9.1.2) it follows that $p(\lambda)$ has the form

$$p_n(\lambda) = (a_{11} - \lambda)(a_{22} - \lambda) \cdots (a_{nn} - \lambda) + q(\lambda), \qquad (9.1.3)$$

$$= (-1)^n(\lambda^n - \xi_{n-1}\lambda^{n-1} - \cdots \xi_0). \qquad (9.1.4)$$

where $q(\lambda)$ has degree at most $n - 2$. Thus, by the fundamental theorem of algebra the matrix A has exactly n eigenvalues λ_i, $i = 1 : n$, counting multiple roots according to their multiplicities, and we can write

$$p(\lambda) = (\lambda_1 - \lambda)(\lambda_2 - \lambda) \cdots (\lambda_n - \lambda).$$

[30] From Latin verb *specere* meaning "to look".

Using the relation between roots and coefficients of an algebraic equation we obtain

$$p(0) = \lambda_1 \lambda_2 \cdots \lambda_n = \det(A), \tag{9.1.5}$$

Further, using the relation between roots and coefficients of an algebraic equation we obtain

$$\lambda_1 + \lambda_2 + \cdots + \lambda_n = \operatorname{trace}(A). \tag{9.1.6}$$

where $\operatorname{trace}(A) = a_{11} + a_{22} + \cdots + a_{nn}$ is the **trace** of the matrix A. This relation is useful for checking the accuracy of a computed spectrum.

Theorem 9.1.1.

Let $A \in \mathbf{C}^{n\times n}$. Then

$$\lambda(A^T) = \lambda(A), \quad \lambda(A^H) = \bar{\lambda}(A).$$

Proof. Since $\det(A^T - \lambda I)^T = \det(A - \lambda I)^T = \det(A - \lambda I)$ it follows that A^T and A have the same characteristic polynomial and thus same set of eigenvalues. For the second part note that $\det(A^H - \bar{\lambda} I) = \det(A - \lambda I)^H$ is zero if and only if $\det(A - \lambda I)$ is zero. □

By the above theorem, if λ is an eigenvalue of A then $\bar{\lambda}$ is an eigenvalue of A^H, i.e., $A^H y = \overline{\lambda} y$ for some vector $y \neq 0$, or equivalently

$$y^H A = \lambda y^H, \qquad y \neq 0. \tag{9.1.7}$$

Here y is called a **left** eigenvector of A, and consequently if $Ax = \lambda x$, x is also called a **right** eigenvector of A. For a Hermitian matrix $A^H = A$ and thus $\overline{\lambda} = \lambda$, i.e., λ is real. In this case the left and right eigenvectors can be chosen to coincide.

Theorem 9.1.2.

Let λ_i and λ_j be two distinct eigenvalues of $A \in \mathbf{C}^{n\times n}$, and let y_i and x_j be left and right eigenvectors corresponding to λ_i and λ_j respectively. Then $y_i^H x_j = 0$, i.e., y_i and x_j are orthogonal.

Proof. By definition we have

$$y_i^H A = \lambda_i y_i^H, \qquad A x_j = \lambda_j x_j.$$

Multiplying the first equation with x_j from the right and the second with y_i^H from the left and subtracting we obtain $(\lambda_i - \lambda_j) y_i^H x_j = 0$. Since $\lambda_i \neq \lambda_j$ the theorem follows. □

Definition 9.1.3.

Denote the eigenvalues of the matrix $A \in \mathbf{C}^{n\times n}$ by $\lambda_i|$, $i = 1 : n$. The **spectral radius** *of A is is the maximal absolute value of the eigenvalues of A*

$$\rho(A) = \max_i |\lambda_i|. \tag{9.1.8}$$

The **spectral abscissa** *is the maximal real part of the eigenvalues of A*

$$\alpha(A) = \max_i \Re\lambda_i. \tag{9.1.9}$$

Two matrices $A \in \mathbf{C}^{n\times n}$ and $\tilde{A} \in \mathbf{C}^{n\times n}$ are said to be **similar** if there is a square nonsingular matrix $S \in \mathbf{C}^{n\times n}$ such that

$$\tilde{A} = S^{-1}AS, \tag{9.1.10}$$

The transformation (9.1.10) is called a **similarity transformation** of A. Similarity of matrices is an equivalence transformation, i.e., if A is similar to B and B is similar to C then A is similar to C.

Theorem 9.1.4.

If A and B are similar, then A and B have the same characteristic polynomial, and hence the same eigenvalues. Further, if $B = S^{-1}AS$ and y is an eigenvector of B corresponding to λ then Sy is an eigenvector of A corresponding to λ.

Proof. We have

$$\begin{aligned}\det(B-\lambda I) &= \det(S^{-1}AS-\lambda I) = \det(S^{-1}(A-\lambda I)S)\\ &= \det\left(S^{-1}\right)\det(A-\lambda I)\det(S) = \det(A-\lambda I).\end{aligned}$$

Further, from $AS = SB$ it follows that $ASy = SBy = \lambda Sy$. $\square$

Similarity transformations corresponds to a change in the coordinate system. Similar matrices represent the same linear transformation in different coordinate systems.

Let the matrix $A \in \mathbf{R}^{n\times n}$ have the factorization $A = BC$, where $B \in \mathbf{R}^{n\times n}$ is invertible, and set $\tilde{A} = CB$. Then $\tilde{A}$ is similar to A since

$$A = BC = B^{-1}(BC)B = CB = \tilde{A}. \tag{9.1.11}$$

A slightly more general result is the following:

Lemma 9.1.5.

Let $A = BC \in \mathbf{C}^{n\times n}$, where $B \in \mathbf{C}^{n\times p}$ and $C \in \mathbf{C}^{p\times m}$, and set $\tilde{A} = CB \in \mathbf{R}^{p\times p}$. Then the nonzero eigenvalues of A and and $\tilde{A}$ are the same.

Proof. The result follows from the identty

$$S^{-1}\begin{pmatrix} BC & 0\\ C & 0\end{pmatrix} S = \begin{pmatrix} 0 & 0\\ C & CB\end{pmatrix}, \qquad S = \begin{pmatrix} I & B\\ 0 & I\end{pmatrix}.$$

This shows that the two block triangular matrices are similar and therefore have the same eigenvalues. $\square$

Many important algorithms for computing eigenvalues and eigenvectors use a *sequence of similarity transformations*

$$A_k = P_k^{-1} A_{k-1} P_k, \quad k = 1, 2, \ldots,$$

where $A_0 = A$, to transform the matrix A into a matrix of simpler form. The matrix A_k is similar to A and the eigenvectors x of A and y of A_k are related by $x = P_1 P_2 \cdots P_k y$. for example, if the matrix A can be transformed to triangular form, then its eigenvalues equal the diagonal elements.

Let $Ax_i = \lambda_i x_i$, $i = 1 : n$. It is easily verified that these n equations are equivalent to the single matrix equation

$$AX = X\Lambda, \qquad \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n),$$

where the columns of $X = (x_1, \ldots, x_n)$ are right eigenvectors of A.

From (9.1.14) it follows that $X^{-1}A = \Lambda X^{-1}$, which shows that *the rows of X^{-1} are left eigenvectors y_i^H*. We can also write $A = X\Lambda X^{-1} = X\Lambda Y^H$, or

$$A = \sum_{i=1}^{n} \lambda_i P_i, \quad P_i = x_i y_i^H. \tag{9.1.12}$$

Since $Y^H X = I$ it follows that the left and right eigenvectors are biorthogonal, $y_i^H x_j = 0$, $i \neq j$, and $y_i^H x_i = 1$. Hence, P_i is a projection ($P_i^2 = P_i$) and (9.1.12) is called the **spectral decomposition** of A. The decomposition (9.1.12) is essentially unique. If λ_{i_1} is an eigenvalue of multiplicity m and $\lambda_{i_1} = \lambda_{i_2} = \cdots = \lambda_{i_m}$, then the vectors $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$ can be chosen as any basis for the null space of $A - \lambda_{i_1} I$.

Suppose that for a matrix $X \in \mathbf{C}^{n\times k}$, $\operatorname{rank}(X) = k \leq n$, it holds that

$$AX = XB, \qquad B \in \mathbf{C}^{k\times k}.$$

Any vector $x \in \mathcal{R}(X)$ can be written $x = Xz$ for some vector $z \in \mathbf{C}^k$. Thus, $Ax = AXz = XBz \in \mathcal{R}(X)$ and $\mathcal{R}(X)$ is called a **right invariant subspace**. If $By = \lambda y$, it follows that

$$AXy = XBy = \lambda Xy,$$

and so *any eigenvalue λ of B is also an eigenvalue of A and Xy a corresponding eigenvector.*

Similarly, if $Y^H A = BY^H$, where $Y \in \mathbf{C}^{n\times k}$, $\operatorname{rank}(Y) = k \leq n$, then $\mathcal{R}(Y)$ is a **left invariant** subspace. If $v^H B = \lambda v^H$ it follows that

$$v^H Y^H A = v^H B Y^H = \lambda v^H Y^H,$$

and so λ is an eigenvalue of A and Yv is a left eigenvector.

Lemma 9.1.6.

Let $\mathcal{S}$ be a invariant subspace of $A \in \mathbf{C}^{n\times n}$ of dimension $k < n$. Let S be a nonsingular matrix whose first k columns are a basis for $\mathcal{S}$. Then

$$B = S^{-1}AS = \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}$$

is block upper triangular, and $B_{11} \in \mathbf{C}^{k\times k}$.

Proof. Let $S = (S_1, S_2)$, where S_1 is a basis for an invariant subspace. Then $AS_1 = S_1 B_{11}$, where $B_{11} \in \mathbf{C}^{k\times k}$. Then we have

$$S^{-1}AS = S^{-1}(AS_1, AS_2) = S^{-1}(S_1B_{11}, AS_2) = \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}. \tag{9.1.13}$$

□

The similarity transformation (9.1.13) reduces the eigenproblem for A into two smaller eigenvalue problems for B_{11} and B_{22}. Note that if S_1 has orthonormal columns, then $S = (Q_1, Q_2)$ in (9.1.13) can be chosen as a Unitary matrix.

When a basis S_1 for an invariant subspace of dimension k of A is known, then the remaining $n-k$ eigenvalues of A can be found from B_{22}. This process is called **deflation** and is a powerful tool for computation of eigenvalues and eigenvectors.

9.1.3 The Jordan Canonical Form

If the eigenvectors $x_1, x_2, \ldots, x_n$ of a matrix $A \in \mathbf{C}^{n\times n}$ are linearly independent, then the matrix of eigenvectors $X = (x_1, x_2, \ldots, x_n)$ is nonsingular and it holds that

$$X^{-1}AX = \Lambda; \tag{9.1.14}$$

that is, a similarity transformation by X transforms A to diagonal form. An important fact is that by the following theorem a matrix is diagonalizable if it has no multiple eigenvalues.

Theorem 9.1.7.

Let $x_1, \ldots, x_k$ be eigenvectors of $A \in \mathbf{C}^{n\times n}$ corresponding to distinct eigenvalues $\lambda_1, \ldots, \lambda_k$. Then the vectors $x_1, \ldots, x_k$ are linearly independent. In particular, if all the eigenvalues of a matrix A are distinct then A has a complete set of linearly independent eigenvectors.

Proof. Assume that only the vectors $x_1 \ldots, x_p$, $p < k$, are linearly independent and that $x_{p+1} = \gamma_1 x_1 + \cdots + \gamma_p x_p$. Then $Ax_{p+1} = \gamma_1 Ax_1 + \cdots + \gamma_p Ax_p$, or

$$\lambda_{p+1}x_{p+1} = \gamma_1\lambda_1 x_1 + \cdots + \gamma_p\lambda_p x_p.$$

It follows that $\sum_{i=1}^{p} \gamma_i(\lambda_i - \lambda_{p+1})x_i = 0$. Since $\gamma_i \neq 0$ for some i and $\lambda_i - \lambda_{p+1} \neq 0$ for all i, this contradicts the assumption of linear independence. Hence, we must have $p = k$ linearly independent vectors. □

A matrix A may not have a full set of n linearly independent eigenvectors. Let $\lambda_1, \cdots, \lambda_k$ be the distinct zeros of $p(\lambda)$ and let σ_i be the multiplicity of λ_i, $i = 1, ..., k$. The integer σ_i is called the **algebraic multiplicity** of the eigenvalue λ_i and

$$\sigma_1 + \sigma_2 + \cdots + \sigma_k = n.$$

To every distinct eigenvalue corresponds at least one eigenvector. All the eigenvectors corresponding to the eigenvalue λ_i form a linear subspace $L(\lambda_i)$ of $\mathbf{C}^n$ of dimension

$$\rho_i = n - \operatorname{rank}(A - \lambda_i I). \tag{9.1.15}$$

The integer ρ_i is called the **geometric multiplicity** of λ_i, and specifies the maximum number of linearly independent eigenvectors associated with λ_i. The eigenvectors are not in general uniquely determined.

Theorem 9.1.8.
The geometric and algebraic multiplicity of a matrix satisfy the inequality

$$\rho(\lambda) \le \sigma(\lambda). \tag{9.1.16}$$

Proof. Let $\bar\lambda$ be an eigenvalue with geometric multiplicity $\rho = \rho(\bar\lambda)$ and let $x_1, \dots, x_\rho$ be linearly independent eigenvectors associated with $\bar\lambda$. If we put $X_1 = (x_1, \dots, x_\rho)$ then we have $AX_1 = \bar\lambda X_1$. We now let $X_2 = (x_{\rho+1}, \cdots, x_n)$ consist of $n - \rho$ more vectors such that the matrix $X = (X_1, X_2)$ is nonsingular. Then it follows that the matrix $X^{-1}AX$ must have the form

$$X^{-1}AX = \begin{pmatrix} \bar\lambda I & B \\ 0 & C \end{pmatrix}$$

and hence the characteristic polynomial of A, or $X^{-1}AX$ is

$$p(\lambda) = (\bar\lambda - \lambda)^\rho \det(C - \lambda I).$$

Thus, the algebraic multiplicity of $\bar\lambda$ is at least equal to ρ. $\square$

If $\rho(\lambda) = \sigma(\lambda)$ then the eigenvalue λ is said to be **semisimple**; if $\rho(\lambda) < \sigma(\lambda)$ then λ is **defective**. A matrix with at least one defective eigenvalue is **defective**, otherwise it is **nondefective**. The eigenvectors of a nondefective matrix A span the space $\mathbf{C}^n$ and A is said to have a complete set of eigenvectors. A matrix is nondefective if and only if it is semisimple. We now state without proof the following fundamental **Jordan Canonical Form**.[31] For a proof based on the block diagonal decomposition in Theorem 9.1.16, see Fletcher and Sorensen [134, 1983].

Theorem 9.1.9 (*The Jordan Canonical Form*)**.**

If $A \in \mathbf{C}^{n\times n}$, then there is a nonsingular matrix $X \in \mathbf{C}^{n\times n}$, such that

$$X^{-1}AX = J = \operatorname{diag}\left(J_{m_1}(\lambda_1), \cdots, J_{m_t}(\lambda_t)\right), \tag{9.1.17}$$

[31] Marie Ennemond Camille Jordan (1838–1922), French mathematician, professor at École Polytechnique and Collége de France. Jordan made important contributions to finite group theory, linear and multilinear algebra as well as differential equations. His paper on the canonical form was published in 1870.

where $m_i \geq 1$, $i = 1 : t$, *and*

$$J_{m_i}(\lambda_i) = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix} = \lambda_i I + S \in \mathbf{C}^{m_i \times m_i}. \tag{9.1.18}$$

The numbers $m_1, \ldots, m_t$ *are unique and* $\sum_{i=1}^{t} m_i = n$.

To each Jordan block $J_{m_i}(\lambda_i)$ there corresponds exactly one eigenvector. Hence, the number of Jordan blocks corresponding to a multiple eigenvalue λ equals the geometric multiplicity of λ. The vectors $x_2, \ldots, x_{m_1}$, which satisfy

$$Ax_1 = \lambda_1 x_1, \quad Ax_{i+1} = \lambda_1 x_{i+1} + x_i, \quad i = 1 : m_1 - 1.$$

are called **principal vectors** of A.

The form (9.1.17) is called the Jordan canonical form of A, and is unique up to the ordering of the Jordan blocks. Note that the same eigenvalue may appear in several different Jordan blocks. A matrix for which this occurs is called **derogatory**. The Jordan canonical form has the advantage that it displays all eigenvalues and eigenvectors of A explicitly.

A serious disadvantage is that the Jordan canonical form is not in general a continuous function of the elements of A. For this reason the Jordan canonical form of a nondiagonalizable matrix may be very difficult to determine numerically.

Example 9.1.1.

Consider the matrices of the form

$$J_m(\lambda, \epsilon) = \begin{pmatrix} \lambda & 1 & & \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & \lambda \end{pmatrix} \in \mathbf{C}^{m \times m}.$$

The matrix $J_m(\lambda, 0)$ has an eigenvalue equal to λ of multiplicity m, and is in Jordan canonical form. For any $\epsilon > 0$ the matrix $J_m(\lambda, \epsilon)$ has m distinct eigenvalues μ_i, $i = 1 : m$, which are the roots of the equation $(\lambda - \mu)^m - (-1)^m \epsilon = 0$. Hence, $J_m(\lambda, \epsilon)$ is diagonalizable for any $\epsilon \neq 0$, and its eigenvalues λ_i satisfy$|\lambda_i - \lambda| = |\epsilon|^{1/m}$. For example, if $m = 10$ and $\epsilon = 10^{-10}$, then the perturbation is of size 0.1.

The minimal polynomial of A can be read off from its Jordan canonical form. Consider a Jordan block $J_m(\lambda) = \lambda I + N$ of order m and put $q(z) = (z - \lambda)^j$. Then we have $q(J_m(\lambda)) = N^j = 0$ for $j \geq m$. The minimal polynomial of a matrix A with the *distinct* eigenvalues $\lambda_1, \ldots, \lambda_k$ then has the form

$$q(z) = (z - \lambda_1)^{m_1} (z - \lambda_2)^{m_2} \cdots (z - \lambda_k)^{m_k}, \tag{9.1.19}$$

where m_j is the highest dimension of any Jordan box corresponding to the eigenvalue λ_j, $j = 1 : k$.

As a corollary we obtain **Cayley–Hamilton theorem**,[32] which states that the characteristic polynomial $p(z)$ of a matrix A satisfies $p(A) = 0$. The polynomials

$$\pi_i(z) = \det\big(zI - J_{m_i}(\lambda_i)\big) = (z - \lambda_i)^{m_i}$$

are called **elementary divisors** of A. They divide the characteristic polynomial of A. The elementary divisors of the matrix A are all linear if and only if the Jordan canonical form is diagonal.

We end with an approximation theorem due to Bellman, which sometimes makes it possible to avoid the complication of the Jordan canonical form.

Theorem 9.1.10.

Let $A \in \mathbf{C}^{n\times n}$ be a given matrix. Then for any $\epsilon > 0$ there exists a matrix B with $\|A - B\|_2 \le \epsilon$, such that B has n distinct eigenvalues. Hence, the class of diagonalizable matrices is dense in $\mathbf{C}^{n\times n}$.

Proof. Let $X^{-1}AX = J$ be the Jordan canonical form of A. Then, by a slight extension of Example 9.1.1 it follows that there is a matrix $J(\delta)$ with distinct eigenvalues such that $\|J - J(\delta)\|_2 = \delta$. (Show this!) Take $B = XJ(\delta)X^{-1}$. Then

$$\|A - B\|_2 \le \epsilon, \quad \epsilon = \delta\|X\|_2\|X^{-1}\|_2.$$

□

For any nonzero vector $v_1 = v$, define a sequence of vectors by

$$v_{k+1} = Av_k = A^k v_1. \tag{9.1.20}$$

Let v_{m+1} be the first of these vectors that can be expressed as a linear combination of the preceding ones. (Note that we must have $m \le n$.) Then for some polynomial p of degree m

$$p(\lambda) = c_0 + c_1\lambda + \cdots + \lambda^m$$

we have $p(A)v = 0$, i.e., p annihilates v. Since p is the polynomial of minimal degree that annihilates v it is called the **minimal polynomial** and m the **grade** of v with respect to A.

Of all vectors v there is at least one for which the degree is maximal, since for any vector $m \le n$. If v is such a vector and q its minimal polynomial, then it can be shown that $q(A)x = 0$ for any vector x, and hence

$$q(A) = \gamma_0 I + \gamma_1 A + \cdots + \gamma_{s-1}A^{s-1} + A^s = 0.$$

This polynomial p is the **minimal polynomial** for the matrix A, see Section 9.1.4.

[32] Arthur Cayley (1821–1895) English mathematician studied at Trinity College, Cambridge. Cayley worked as a lawyer for 14 years before in 1863 he was appointed Sadlerian Professor of Pure Mathematics at Cambridge. His most important work was in developing the algebra of matrices and in geometry and group theory. His work on matrices served as a foundation for quantum mechanics developed by Heisenberg in 1925.

9.1.4 The Schur Normal Form

Using similarity transformations it is possible to transform a matrix into one of several canonical forms, which reveal its eigenvalues and gives information about the eigenvectors. These canonical forms are useful also for extending analytical functions of one variable to matrix arguments.

The computationally most useful of the canonical forms is the triangular, or **Schur normal form** due to Schur [332, 1909].

Theorem 9.1.11 (*The Schur Normal Form*).

Given $A \in \mathbf{C}^{n\times n}$ there exists a unitary matrix $U \in \mathbf{C}^{n\times n}$ such that

$$U^H A U = T = D + N, \tag{9.1.21}$$

where T is upper triangular, N strictly upper triangular, $D = \mathrm{diag}\,(\lambda_1, \cdots, \lambda_n)$, and $\lambda_i, i = 1, ..., n$ are the eigenvalues of A. Furthermore, U can be chosen so that the eigenvalues appear in arbitrary order in D.

Proof. The proof is by induction on the order n of the matrix A. For $n = 1$ the theorem is trivially true. Assume the theorem holds for all matrices of order $n-1$. We will show that it holds for any matrix $A \in \mathbf{C}^{n\times n}$.

Let λ be an arbitrary eigenvalue of A. Then, $Ax = \lambda x$, for some $x \neq 0$ and we let $u_1 = x/\|x\|_2$. Then we can always find $U_2 \in \mathbf{C}^{n\times n-1}$ such that $U = (u_1,\ U_2)$ is a unitary matrix. Since $AU = A(u_1, U_2) = (\lambda u_1,\ AU_2)$ we have

$$U^H A U = \begin{pmatrix} u_1^H \\ U_2^H \end{pmatrix} AU = \begin{pmatrix} \lambda u_1^H u_1 & u_1^H A U_2 \\ \lambda U_2^H u_1 & U_2^H A U_2 \end{pmatrix} = \begin{pmatrix} \lambda & w^H \\ 0 & B \end{pmatrix}.$$

Here B is of order $n-1$ and by the induction hypothesis there exists a unitary matrix $\tilde{U}$ such that $\tilde{U}^H B \tilde{U} = \tilde{T}$. Then

$$\overline{U}^H A \overline{U} = T = \begin{pmatrix} \lambda & w^H \tilde{U} \\ 0 & \tilde{T} \end{pmatrix}, \quad \overline{U} = U \begin{pmatrix} 1 & 0 \\ 0 & \tilde{U} \end{pmatrix},$$

where $\overline{U}$ is unitary. From the above it is obvious that we can choose U to get the eigenvalues of A arbitrarily ordered on the diagonal of T. □

The advantage of the Schur normal form is that it can be obtained using a numerically stable unitary transformation. The eigenvalues of A are displayed on the diagonal. The columns in $U = (u_1, u_2, \ldots, u_n)$ are called **Schur vectors**. It is easy to verify that the nested sequence of subspaces

$$S_k = \mathrm{span}[u_1, \ldots, u_k], \qquad k = 1, \ldots, n,$$

are invariant subspaces. However, of the Schur vectors in general only u_1 is an eigenvector.

If the matrix A is real, we would like to restrict ourselves to real similarity transformations, since otherwise we introduce complex elements in $U^{-1}AU$. If A

has complex eigenvalues, then A obviously cannot be reduced to triangular form by a real orthogonal transformation. For a real matrix A the eigenvalues occur in complex conjugate pairs, and it is possible to reduce A to block triangular form T, with 1×1 and 2×2 diagonal blocks, in which the 2×2 blocks correspond to pairs of complex conjugate eigenvalues. T is then said to be in **quasi-triangular** form.

Theorem 9.1.12 (*The Real Schur Form***).**

Given $A \in \mathbf{R}^{n\times n}$ there exists a real orthogonal matrix $Q \in \mathbf{R}^{n\times n}$ such that

$$Q^T AQ = T = D + N, \tag{9.1.22}$$

where T is real block upper triangular, D is block diagonal with 1×1 and 2×2 blocks, and where all the 2×2 blocks have complex conjugate eigenvalues.

Proof. Let A have the complex eigenvalue $\lambda \neq \bar{\lambda}$ corresponding to the eigenvector x. Then, since $A\bar{x} = \bar{\lambda}\bar{x}$, $\bar{\lambda}$ is also an eigenvalue with eigenvector $\bar{x} \neq x$, and $\mathcal{R}(x, \bar{x})$ is an invariant subspace of dimension 2. Let

$$X_1 = (x_1, x_2), \quad x_1 = x + \bar{x}, \quad x_2 = i(x - \bar{x})$$

be a real basis for this invariant subspace. Then $AX_1 = X_1 M$ where $M \in \mathbf{R}^{2\times 2}$ has eigenvalues λ and $\bar{\lambda}$. Let $X_1 = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1 R$ be the QR decomposition of X_1. Then $AQ_1 R = Q_1 RM$ or $AQ_1 = Q_1 P$, where $P = RMR^{-1} \in \mathbf{R}^{2\times 2}$ is similar to M. Using (9.1.13) with $X = Q$, we find that

$$Q^T AQ = \begin{pmatrix} P & W^H \\ 0 & B \end{pmatrix}.$$

where P has eigenvalues λ and $\bar{\lambda}$. An induction argument completes the proof. $\square$

A matrix $A \in \mathbf{C^{n\times n}}$ is said to be **normal** if

$$A^H A = AA^H. \tag{9.1.23}$$

Theorem 9.1.13.

A matrix $A \in \mathbf{C}^{n\times n}$ is normal, $A^H A = AA^H$, if and only if A can be unitarily diagonalized, i.e., there exists a unitary matrix $U \in \mathbf{C}^{n\times n}$ such that

$$U^H AU = D = \operatorname{diag}(\lambda_1, \cdots, \lambda_n).$$

Proof. If A is normal then for unitary U so is $U^H AU$, since

$$(U^H AU)^H U^H AU = U^H (A^H A)U = U^H (AA^H)U = U^H AU(U^H AU)^H.$$

It follows that the upper triangular matrix

$$T = \begin{pmatrix} \lambda_1 & t_{12} & \dots & t_{1n} \\ & \lambda_2 & \dots & t_{2n} \\ & & \ddots & \vdots \\ & & & \lambda_n \end{pmatrix},$$

in the Schur normal form is normal, i.e. $T^H T = TT^H$. Equating the $(1,1)$-element on both sides of the equation $T^H T = TT^H$ we get

$$|\lambda_1|^2 = |\lambda_1|^2 + \sum_{j=2}^{n} |t_{1j}|^2,$$

and thus $t_{1j} = 0$, $j = 2 : n$. In the same way it can be shown that all the other nondiagonal elements in T vanishes, and so T is diagonal.

If on the other hand A is unitarily diagonalizable then we immediately have that

$$A^H A = UD^H DU^H = UDD^H U^H = AA^H.$$

□

Important classes of normal matrices in $\mathbf{C}^{n\times n}$ are:

- A is Hermitian if $A^H = A$.
- A is skew-Hermitian if $A^H = -A$.
- A is unitary if $A^H A^{-1}$.

For matrices $A \in \mathbf{R}^{n\times n}$ the corresponding terms are symmetric ($A^T = A$), skew-symmetric ($A^T = -A$), and orthogonal ($A^T = A^{-1}$).

If $A \in \mathbf{C}^{n\times n}$ and $u \neq 0$ is an eigenvector then $Au = \lambda u$. It follows that $x^H Ax = \lambda \|x\|_2^2$ or

$$\lambda = (u^H Au)/\|u\|_2^2.$$

From this it is easily shown that Hermitian matrices have real eigenvalues, skew-Hermitian matrices have zero or purely imaginary eigenvalues, and Unitary matrices have eigenvalues on the unit circle.

The following relationship between unitary and skew-symmetric matrices are called the Cayley parameterization.

Theorem 9.1.14.

If U is unitary and does not have -1 as an eigenvalue, then

$$U = (I + iH)(I - iH)^{-1},$$

where H is Hermitian and uniquely determined by the formula

$$H = i(I - U)(I + U)^{-1},$$

From Theorem 9.1.13 it follows in particular that any Hermitian matrix may be decomposed into

$$A = U\Lambda U^H = \sum_{i=1}^{n} \lambda_i u_i u_i^H. \tag{9.1.24}$$

with λ_i real. In the special case that A is real and symmetric we can take U to be real and orthogonal, $U = Q = (q_1, \dots, q_n)$, where q_i are orthonormal eigenvectors. Note that in (9.1.24) $u_i u_i^H$ is the unitary projection matrix that projects unitarily onto the eigenvector u_i. We can also write $A = \sum_j \lambda P_j$, where the sum is taken over the *distinct* eigenvalues of A, and P_j projects $\mathbf{C^n}$ unitarily onto the eigenspace belonging to λ_j. (This comes closer to the formulation given in functional analysis.)

Note that although U in the Schur normal form (9.1.21) is not unique, $\|N\|_F$ is independent of the choice of U, and

$$\Delta_F^2(A) \equiv \|N\|_F^2 = \|A\|_F^2 - \sum_{i=1}^{n} |\lambda_i|^2.$$

The quantity $\Delta_F(A)$ is called the **departure from normality** of A.

Let P_n be the permutation matrix,

$$P_n = \begin{pmatrix} 0 & & \cdots & & 1 \\ 1 & & & & \\ & 1 & & & \vdots \\ \vdots & & \ddots & & \\ 0 & & \cdots & 1 & 0 \end{pmatrix} \tag{9.1.25}$$

Let e_i be the ith unit vector. Then we have $P_n e_1 = e_2$, $P_n e_2 = e_3$,…,$P_n e_{n-1} = e_n$, and $P_n e_n = e_1$, Thus, $P_n^n e_i = e_i$, $i?1 : n$, and it follows that $P_n^n - I = 0$. Suppose now that $C = \sum_{k=0}^{n-1} c_k P^k = 0$,

$$C = \begin{pmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_3 \\ \vdots & \vdots & \vdots & \dots & \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots & c_0 \end{pmatrix}. \tag{9.1.26}$$

Then $c_0 = c_1 = \cdots c_{n-1} = 0$ and thus no polynomial of degree $n-1$ in P vanishes. Hence, $\varphi(\lambda) = \lambda^n - 1$ is the characteristic polynomial and the eigenvalues of P_n are the n roots of unity

$$\omega_j = e^{-2\pi j/n}, \quad j = 0 : n-1.$$

The corresponding eigenvectors are

$$x_j = (1, \omega_j, \dots, \omega_j^{n-1})^T, \quad j = 0 : n-1.$$

A matrix $C \in \mathbf{R}^{n\times n}$ of the form (9.1.26) is called a **circulant matrix**. Each column in C is a cyclic down-shifted version of the previous column. Since any circulant matrix is a polynomial in the matrix P_n, $C = q(P_n)$ the circulant has the same eigenvectors as P_n and its eigenvalues are equal to

$$\lambda_i = q(\omega_j), \quad j = 0 : n-1,$$

Let the matrix A have the block triangular form

$$A = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix}, \tag{9.1.27}$$

where B and D are square. Suppose that we wish to reduce A to **block diagonal** form by a similarity transformation of the form

$$P = \begin{pmatrix} I & Q \\ 0 & I \end{pmatrix}, \qquad P^{-1} = \begin{pmatrix} I & -Q \\ 0 & I \end{pmatrix}.$$

This gives the result

$$P^{-1}AP = \begin{pmatrix} I & -Q \\ 0 & I \end{pmatrix} \begin{pmatrix} B & C \\ 0 & D \end{pmatrix} \begin{pmatrix} I & Q \\ 0 & I \end{pmatrix} = \begin{pmatrix} B & C - QD + BQ \\ 0 & D \end{pmatrix}.$$

The result is a block diagonal matrix if and only if $BQ - QD = -C$. This equation, which is a linear equation in the elements of Q, is called **Sylvester's equation**[33]

We will investigate the existence and uniqueness of solutions to the general Sylvester equation

$$AX - XB = C, \quad X \in \mathbf{R}^{n\times m}, \tag{9.1.28}$$

where $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{m\times m}$. We prove the following result.

Theorem 9.1.15.

The matrix equation (9.1.28) *has a unique solution if and only if*

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

Proof. From Theorem 9.1.11 follows the existence of the Schur decompositions

$$U_1^H A U_1 = S, \qquad U_2^H B U_2 = T,$$

where S and T are upper triangular and U_1 and U_2 are unitary matrices. Using these decompositions (9.1.28) can be reduced to

$$SY - YT = F, \qquad Y = U_1^H X U_2, \quad F = U_1^H C U_2.$$

[33] James Joseph Sylvester English mathematician (1814–1893). Because of his Jewish faith Sylvester had trouble in finding an adequate research position in England. His most productive period was in 1877–1884, when he held a chair at Johns Hopkins university in USA. He considered the homogeneous case in 1884.

The kth column of this equation is

$$Sy_k - \begin{pmatrix} y_1 & y_2 & \cdots & y_k \end{pmatrix} \begin{pmatrix} t_{1k} \\ t_{2k} \\ \vdots \\ t_{kk} \end{pmatrix} = f_k, \quad k = 1 : n. \tag{9.1.29}$$

For $k = 1$ this equation gives

$$Sy_1 - t_{11}y_1 = (S - t_{11}I)y_1 = d_1.$$

Here t_{11} is an eigenvalue of T and hence is *not* an eigenvalue of S. Therefore, the triangular matrix $S - t_{11}I$ is not singular and we can solve for y_1. Now suppose that we have found $y_1, \ldots, y_{k-1}$. From the kth column of the system

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

Here the right hand side is known and, by the argument above, the triangular matrix $S - t_{kk}I$ nonsingular. Hence, it can be solved for y_k. The proof now follows by induction. □

If we have an algorithm for computing the Schur decompositions this proof gives an algorithm for solving the Sylvester equation. It involves solving m triangular equations and requires $O(mn^2)$ operations; see Bartels and Stewart [25] and Golub et al. [177].

An important special case of (9.1.28) is the **Lyapunov equation**

$$AX + XA^H = C. \tag{9.1.30}$$

Here $B = -A^H$, and hence by Theorem 9.1.15 this equation has a unique solution if and only if the eigenvalues of A satisfy $\lambda_i + \bar{\lambda}_j \neq 0$ for all i and j. Further, if $C^H = C$ the solution X is Hermitian. In particular, if all eigenvalues of A have negative real part, then all eigenvalues of $-A^H$ have positive real part, and the assumption is satisfied; see Hammarling [197].

We have seen that a given block triangular matrix (9.1.27) can be transformed by a similarity transformation to block diagonal form provided that B and C have disjoint spectra. The importance of this construction is that it can be applied recursively.

If A is not normal, then the matrix T in its Schur normal form cannot be diagonal. To transform T to a form closer to a diagonal matrix we have to use *non-unitary similarities*. By Theorem 9.1.11 we can order the eigenvalues so that in the Schur normal form

$$D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n), \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

We now show how to obtain the following block diagonal form:

Theorem 9.1.16 (Block Diagonal Decomposition)**.**

Let the distinct eigenvalues of A be $\lambda_1, \cdots, \lambda_k$, and in the Schur normal form let $D = \operatorname{diag}(D_1, \ldots, D_k)$, $\quad D_i = \lambda_i I, \quad i = 1 : k$. Then there exists a nonsingular matrix Z such that

$$Z^{-1} U^H A U Z = Z^{-1} T Z = \operatorname{diag}(\lambda_1 I + N_1, \cdots, \lambda_k I + N_k),$$

where N_i, $i = 1 : k$ are strictly upper triangular. In particular, if the matrix A has n distinct eigenvalues the matrix D diagonal.

Proof. Consider first the matrix $T = \begin{pmatrix} \lambda_1 & t \\ 0 & \lambda_2 \end{pmatrix} \in \mathbf{C}^{2\times 2}$, where $\lambda_1 \neq \lambda_2$. Perform the similarity transformation

$$M^{-1} T M = \begin{pmatrix} 1 & -m \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 & t \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda_1 & m(\lambda_1 - \lambda_2) + t \\ 0 & \lambda_2 \end{pmatrix}.$$

where M is an upper triangular elementary elimination matrix; see Sec. 7.2.2. By taking $m = t/(\lambda_2 - \lambda_1)$, we can annihilate the off-diagonal element in T.

In the general case let t_{ij} be an element in T outside the block diagonal. Let M_{ij} be a matrix which differs from the unit matrix only in the (i, j)th element, which is equal to m_{ij}. Then as above we can choose m_{ij} so that the element (i, j) is annihilated by the similarity transformation $M_{ij}^{-1} T M_{ij}$. Since T is upper triangular this transformation will not affect any already annihilated off-diagonal elements in T with indices (i', j') if $j' - i' < j - i$. Hence, we can annihilate all elements t_{ij} outside the block diagonal in this way, starting with the elements on the diagonal closest to the main diagonal and working outwards. For example, in a case with 3 blocks of orders $2, 2, 1$ the elements are eliminated in the order

$$\begin{pmatrix} \times & \times & 2 & 3 & 4 \\ & \times & 1 & 2 & 3 \\ & & \times & \times & 2 \\ & & & \times & 1 \\ & & & & \times \end{pmatrix}.$$

Further details of the proof is left to the reader. $\square$

9.1.5 Nonnegative Matrices

Theorem 9.1.17.

Assume that the matrix A can be reduced by a permutation to the block upper triangular form

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ 0 & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A_{NN} \end{pmatrix}, \tag{9.1.31}$$

where each diagonal block A_{ii} is square. Then $\lambda(A) = \bigcup_{i=1}^{N} \lambda(A_{ii})$, where $\lambda(A)$ denotes the spectrum of A. In particular, the eigenvalues of a triangular matrix are its diagonal elements.

Non-negative matrices arise in many applications and play an important role in, e.g., queuing theory, stochastic processes, and input-output analysis.

A matrix $A \in \mathbf{R}^{m\times n}$ is called **nonnegative** if $a_{ij} \geq 0$ for each i and j and **positive** if $a_{ij} > 0$ for each i and j. Similarly, a vector $x \in \mathbf{R}^n$ is called nonnegative if $x_i \geq 0$ and positive if $x_i > 0$, $i = 1 : n$. If A and B are nonnegative, then so is their sum $A + B$ and product AB. Hence, nonnegative matrices form a convex set. Further, if If A is nonnegative, then so is A^k for all $k \geq 0$.

If A and B are two $m \times n$ matrices and

$$a_{ij} \leq b_{ij}, \quad i = 1 : m, \quad j = 1 : n,$$

then we write $A \leq B$. The binary relation "$\leq$" then defines a partial ordering of the matrices in $\mathbf{R}^{m\times n}$. This ordeing is transitive since if $A \leq B$ and $B \leq C$ then $A \leq C$. Further useful properties are the following:

Lemma 9.1.18. *Let A,B, and C be nonnegative $n \times n$ matrices with $A \leq B$. Then*

$$AC \leq BC \qquad CA \leq CB, \qquad A^k \leq B^k, \quad \forall k \geq 0.$$

Definition 9.1.19.

A nonsingular matrix $A \in \mathbf{R}^{n\times n}$ is said to be an M-matrix if it has the following properties:

1. $a_{ii} > 0$ *for* $i = 1 : n$.
2. $a_{ij} \leq 0$ *for* $i \neq j$, $i, j = 1 : n$.
3. $A^{-1} \geq 0$.

Lemma 9.1.20. *Let $A \in \mathbf{R}^{n\times n}$ be a square nonnegative matrix and let $s = Ae$, $e = (1\ 1\ \cdots\ 1)^T$ be the vector of row sums of A. Then*

$$\min_i s_i \leq \rho(A) \leq \max_i s_i = \|A\|_1. \tag{9.1.32}$$

The class of nonnegative square matrices have remarkable spectral properties. These were discovered in (1907) by Perron[34] for positive matrices and amplified and generalized for nonnegative irreducible (see Def. 7.8.4) matrices by Frobenius (1912).

[34]Oskar Perron (1880–1975) German mathematician held positions at Heidelberg and Munich. His work covered a vide ranage of topics. He also wrote important textbooks on continued fractions, algebra, and non-Euclidean geometry.

Theorem 9.1.21 (*Perron–Frobenius Theorem*).

Let $A \in \mathbf{R}^{n\times n}$ be a nonnegative irreducible matrix, Then

(i) *A has a real positive simple eigenvalue equal to $\rho(A)$;*

(ii) *To $\rho(A)$ corresponds an eigenvector $x > 0$.*

(iii) *$\rho(A)$ increases when any entry of A increases.*

(iv) *The eigenvalues of modulus $\rho(A)$ are all simple. If there are m eigenvalues of modulus ρ, they must be of the form*

$$\lambda_k = \rho e^{2k\pi i/m}, \quad k = 0 : m-1.$$

If $m = 1$, the the matrix is called **primitive**.

(v) *If $m > 1$, then there exists a permutation matrix P such that*

$$PAP^T = \begin{pmatrix} 0 & A_{12} & 0 & \cdots & 0 \\ 0 & 0 & A_{23} & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \cdots & A_{m-1,m} \\ A_{m1} & 0 & 0 & \cdots & 0 \end{pmatrix},$$

where the zero blocks down the diagonal are square. Such a matrix is called **cyclic** *of index $m > 1$.*

Since the proof of the Perron–Frobenius theorem is too long to be given here we refer to, e.g., Varga [375, Sec. 2.1–2.2] or Berman and Plemmons [31, pp. 27,32].

The Perron–Frobenius theorem is an important tool for analyzing the Markov chains and the asymptotic convergence of stationary iterative methods; see Sec. 10.1.3.

Review Questions

1.1 How are the eigenvalues and eigenvectors of A affected by a similarity transformation?

1.2 What is meant by a (right) invariant subspace of A? Describe how a basis for an invariant subspace can be used to construct a similarity transformation of A to block triangular form. How does such a transformation simplify the computation of the eigenvalues of A?

1.3 What is meant by the algebraic multiplicity and the geometric multiplicity of an eigenvalue of A? When is a matrix said to be defective?

1.4 What is the Schur normal form of a matrix $A \in \mathbf{C}^{n\times n}$?

(b)What is meant by a normal matrix? How does the Schur form simplify for a normal matrix?

1.5 Show that if A and B are normal and $AB = BA$, then AB is normal.

1.6 How can the class of matrices which are diagonalizable by unitary transformations be characterized?

1.7 What is meant by a defective eigenvalue? Give a simple example of a matrix with a defective eigenvalue.

1.8 Give an example of a matrix, for which the minimal polynomial has a lower degree than the characteristic polynomial. Is the characteristic polynomial always divisible by the minimal polynomial?

1.9 Prove the Cayley–Hamilton theorem for a diagonalizable matrix. Then generalize to an arbitrary matrix, either as in the text or by using Bellman's approximation theorem, (Theorem 9.1.15).

Problems

1.1 A matrix $A \in \mathbf{R}^{n\times n}$ is called nilpotent if $A^k = 0$ for some $k > 0$. Show that a nilpotent matrix can only have 0 as an eigenvalue.

1.2 Show that if λ is an eigenvalue of a unitary matrix U then $|\lambda| = 1$.

1.3 (a) Let $A = xy^T$, where x and y are vectors in $\mathbf{R}^n$, $n \geq 2$. Show that 0 is an eigenvalue of A with multiplicity at least $n-1$, and that the remaining eigenvalue is $\lambda = y^T x$.

(b) What are the eigenvalues of a Householder reflector $P = I - 2uu^T$, $\|u\|_2 = 1$?

1.4 What are the eigenvalues of a Givens' rotation

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}?$$

When are the eigenvalues real?

1.6 Let $A \in \mathbf{C}^{n\times n}$ be an Hermitian matrix, λ an eigenvalue of A, and z the corresponding eigenvector. Let $A = S + iK$, $z = x + iy$, where S, K, x, y are real. Show that λ is a double eigenvalue of the real symmetric matrix

$$\begin{pmatrix} S & -K \\ K & S \end{pmatrix} \in \mathbf{R}^{2n\times 2n},$$

and determine two corresponding eigenvectors.

1.7 Show that the matrix

$$K_n = \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

has the characteristic polynomial

$$p(\lambda) = (-1)^n(\lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n).$$

K_n is called the **companion matrix** of $p(\lambda)$. Determine the eigenvectors of K_n corresponding to an eigenvalue λ, and show that there is only one eigenvector even when λ is a multiple eigenvalue.

Remark: The term companion matrix is sometimes used for slightly different matrices, where the coefficients of the polynomial appear, e.g., in the last row or in the last column.

1.8 Draw the graphs $G(A)$, $G(B)$ and $G(C)$, where

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Show that A and C are irreducible but B is reducible.

1.9 Prove the Cayley–Hamilton theorem for a diagonalizable matrix. Then generalize to an arbitrary matrix, either as in the text or by using Bellman's approximation theorem, (Theorem 9.1.15).

1.10 Find a similarity transformation $X^{-1}AX$ that diagonalizes the matrix

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1+\epsilon \end{pmatrix}, \quad \epsilon > 0.$$

How does the transformation X behave as ϵ tends to zero?

1.11 Show that the Sylvester equation (9.1.28) can be written as the linear system

$$(I_m \otimes A - B^T \otimes I_n)\mathrm{vec}(X) = \mathrm{vec}(C), \tag{9.1.33}$$

where $\otimes$ denotes the Kronecker product and $\mathrm{vec}(X)$ is the column vector obtained by stacking the column of X on top of each other.

1.12 Show that the eigenvalues λ_i of a matrix A satisfy the inequalities

$$\sigma_{\min}(A) \le \min_i |\lambda_i| \le \max_i |\lambda_i| \sigma_{\max}(A).$$

Hint: Use the fact that the singular values of A and its Schur decomposition $Q^TAQ = \mathrm{diag}\,(\lambda_i) + N$ are the same.

1.13 Show that Sylvester's equation (9.1.28) can be written as an equation in standard matrix-vector form,

$$\big((I \otimes A) + (-B^T \otimes I)\big)x = c,$$

where the vectors $x, c \in \mathbf{R}^{nm}$ are obtained from $X = (x_1, \ldots, x_m)$ and $C = (c_1, \ldots, c_m)$ by

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}, \qquad c = \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix}.$$

Then use (9.1.20) to give an independent proof that Sylvester's equation has a unique solution if and only if $\lambda_i - \mu_j \neq 0$, $i = 1 : n$, $j = 1 : m$.

9.2 Perturbation Theory and Eigenvalue Bounds

Methods for computing eigenvalues and eigenvectors are subject to roundoff errors. The best we can demand of an algorithm in general is that it yields approximate eigenvalues of a matrix A that are the exact eigenvalues of a slightly perturbed matrix $A + E$. In order to estimate the error in the computed result we need to know the effects of the perturbation E on the eigenvalues and eigenvectors of A. Such results are derived in this section.

9.2.1 Gerschgorin's Theorems

In 1931 the Russian mathematician published a seminal paper [158] on how to obtain estimates of all eigenvalues of a complex matrix. His results can be used both to locate eigenvalues and to derive perturbation results.

Theorem 9.2.1 (*Gerschgorin's Theorem*).

All the eigenvalues of the matrix $A \in \mathbf{C}^{n\times n}$ *lie in the union of the* **Gerschgorin disks** *in the complex plane*

$$\mathcal{D}_i = \{z \mid |z - a_{ii}| \le r_i\}, \qquad r_i = \sum_{j=1, j\ne i}^{n} |a_{ij}|, \qquad i = 1 : n. \tag{9.2.1}$$

Proof. If λ is an eigenvalue there is an eigenvector $x \ne 0$ such that $Ax = \lambda x$, or

$$(\lambda - a_{ii})x_i = \sum_{j=1, j\ne i}^{n} a_{ij}x_j, \quad i = 1 : n.$$

Choose i so that $|x_i| = \|x\|_\infty$. Then

$$|\lambda - a_{ii}| \le \sum_{j=1, j\ne i}^{n} \frac{|a_{ij}||x_j|}{|x_i|} \le r_i. \tag{9.2.2}$$

□

The Gerschgorin theorem is very useful for getting crude estimates for eigenvalues of matrices, and can also be used to get accurate estimates for the eigenvalues of a nearly diagonal matrix. Since A and A^T have the same eigenvalues we can, in the non-Hermitian case, obtain more information about the location of the eigenvalues simply by applying the theorem to A^T .

From (9.2.2) it follows that if the ith component of the eigenvector is maximal, then λ lies in the ith disk. Otherwise the Gerschgorin theorem does not say in *which* disks the eigenvalues lie. Sometimes it is possible to decide this as the following theorem shows.

Theorem 9.2.2.

If the union $\mathcal{M}$ of k Gerschgorin disks $\mathcal{D}_i$ is disjoint from the remaining disks, then $\mathcal{M}$ contains precisely k eigenvalues of A.

Proof. Consider for $t \in [0,1]$ the family of matrices

$$A(t) = tA + (1-t)D_A, \quad D_A = \operatorname{diag}(a_{ii}).$$

The coefficients in the characteristic polynomial are continuous functions of t, and hence the eigenvalues $\lambda(t)$ of $A(t)$ are also continuous functions of t. Since $A(0) = D_A$ and $A(1) = A$ we have $\lambda_i(0) = a_{ii}$ and $\lambda_i(1) = \lambda_i$. For $t = 0$ there are exactly k eigenvalues in $\mathcal{M}$. For reasons of continuity an eigenvalue $\lambda_i(t)$ cannot jump to a subset that does not have a continuous connection with a_{ii} for $t = 1$. Therefore, k eigenvalues of $A = A(1)$ lie also in $\mathcal{M}$. $\square$

Example 9.2.1.

The matrix

$$A = \begin{pmatrix} 2 & -0.1 & 0.05 \\ 0.1 & 1 & -0.2 \\ 0.05 & -0.1 & 1 \end{pmatrix},$$

with eigenvalues $\lambda_1 = 0.8634$, $\lambda_2 = 1.1438$, $\lambda_3 = 1.9928$, has the Gerschgorin disks

$$\mathcal{D}_1 = \{z \mid |z-2| \le 0.15\}; \quad \mathcal{D}_2 = \{z \mid |z-1| \le 0.3\}; \quad \mathcal{D}_3 = \{z \mid |z-1| \le 0.15\}.$$

Since the disk $\mathcal{D}_1$ is disjoint from the rest of the disks, it must contain precisely one eigenvalue of A. The remaining two eigenvalues must lie in $\mathcal{D}_2 \cup \mathcal{D}_3 = \mathcal{D}_2$.

There is another useful sharpening of Gerschgorin's Theorem in case the matrix A is irreducible, cf. Def. 7.8.4.

Theorem 9.2.3.

If A is irreducible then each eigenvalue λ lies in the **interior** *of the union of the Gerschgorin disks, unless it lies on the boundary of* **all** *Gerschgorin disks.*

Proof. If λ lies on the boundary of the union of the Gerschgorin disks, then we have

$$|\lambda - a_{ii}| \ge r_i, \quad \forall i. \tag{9.2.3}$$

Let x be a corresponding eigenvector and assume that $|x_{i_1}| = \|x\|_\infty$. Then from the proof of Theorem 9.2.1 and (9.2.3) it follows that $|\lambda - a_{i_1 i_1}| = r_{i_1}$. But (9.2.2) implies that equality can only hold here if for any $a_{i_1 j} \ne 0$ it holds that $|x_j| = \|x\|_\infty$. If we assume that $a_{i_1,i_2} \ne 0$ then it follows that $|\lambda - a_{i_2 i_2}| = r_{i_2}$. But since A is irreducible for any $j \ne i$ there is a path $i = i_1, i_2, \ldots, i_p = j$. It follows that λ must lie on the boundary of all Gerschgorin disks. $\square$

Example 9.2.2. Consider the real, symmetric matrix

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbf{R}^{n\times n}.$$

Its Gerschgorin disks are

$$|z-2| \le 2, \quad i = 2:n-1, \qquad |z-2| \le 1, \quad i = 1, n,$$

and it follows that all eigenvalues of A satisfy $\lambda \ge 0$. Since zero is on the boundary of the union of these disks, but *not* on the boundary of all disks, zero cannot be an eigenvalue of A. Hence, all eigenvalues are *strictly* positive and A is positive definite.

9.2.2 Perturbation Theorems

In the rest of this section we consider the sensitivity of eigenvalue and eigenvectors to perturbations.

Theorem 9.2.4 (*Bauer–Fike's Theorem***).**

Let the matrix $A \in \mathbf{C}^{n\times n}$ be diagonalizable, $X^{-1}AX = D = diag(\lambda_1, \cdots, \lambda_n)$, and let μ be an eigenvalue to $A+E$. Then for any p-norm

$$\min_{1\le i\le n} |\mu - \lambda_i| \le \kappa_p(X)\|E\|_p. \tag{9.2.4}$$

where $\kappa_p(X) = \|X^{-1}\|_p\,\|X\|_p$ is the condition number of the eigenvector matrix.

Proof. We can assume that μ is not an eigenvalue of A, since otherwise (9.2.4) holds trivially. Since μ is an eigenvalue of $A+E$ the matrix $A+E-\mu I$ is singular and so is also

$$X^{-1}(A+E-\mu I)X = (D-\mu I) + X^{-1}EX.$$

Then there is a vector $z \ne 0$ such that

$$(D-\mu I)z = -X^{-1}EXz.$$

Solving for z and taking norms we obtain

$$\|z\|_p \le \kappa_p(X)\|(D-\mu I)^{-1}\|_p\|E\|_p\|z\|_p.$$

The theorem follows by dividing by $\|z\|_p$ and using the fact that for any p-norm $\|(D-\mu I)^{-1}\|_p = 1/\min\limits_{1\le i\le n}|\lambda_i - \mu|$. □

The Bauer–Fike theorem shows that $\kappa_p(X)$ is an upper bound for the condition number of the eigenvalues of a *diagonalizable matrix* A. In particular, if A is normal we know from the Schur Canonical Form (Theorem 9.1.11) that we can take $X = U$ to be a unitary matrix. Then we have $\kappa_2(X) = 1$, which shows the important result that *the eigenvalues of a normal matrix are perfectly conditioned, also if they have multiplicity greater than one.* On the other hand, for a matrix A which is close to a defective matrix the eigenvalues can be very ill-conditioned, see Example 9.1.1, and the following example.

Example 9.2.3.

Consider the matrix $A = \begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix}$, $0 < \epsilon$ with eigenvector matrix

$$X = \begin{pmatrix} 1 & 1 \\ \sqrt{\epsilon} & -\sqrt{\epsilon} \end{pmatrix}, \qquad X^{-1} = \frac{0.5}{\sqrt{\epsilon}} \begin{pmatrix} \sqrt{\epsilon} & 1 \\ \sqrt{\epsilon} & -1 \end{pmatrix}.$$

If $\epsilon \ll 1$ then

$$\kappa_\infty(X) = \|X^{-1}\|_\infty \|X\|_\infty = \frac{1}{\sqrt{\epsilon}} + 1 \gg 1.$$

Note that in the limit when $\epsilon \to 0$ the matrix A is not diagonalizable.

In general a matrix may have a mixture of well-conditioned and ill-conditioned eigenvalues. Therefore, it is useful to have perturbation estimates for the individual eigenvalues of a matrix A. We now derive first order estimates for simple eigenvalues and corresponding eigenvectors.

Theorem 9.2.5.

Let λ_j be a simple eigenvalue of A and let x_j and y_j be the corresponding right and left eigenvector of A,

$$Ax_j = \lambda_j x_j, \qquad y_j^H A = \lambda_j y_j^H.$$

Then for sufficiently small ϵ the matrix $A + \epsilon E$ has a simple eigenvalue $\lambda_j(\epsilon)$ such that

$$\lambda_j(\epsilon) = \lambda_j + \epsilon \frac{y_j^H E x_j}{y_j^H x_j} + O(\epsilon^2). \tag{9.2.5}$$

Proof. Since λ_j is a simple eigenvalue there is a $\delta > 0$ such that the disk $\mathcal{D} = \{\mu \| |\mu - \lambda_j| < \delta\}$ does not contain any eigenvalues of A other than λ_j. Then using Theorem 9.2.2 it follows that for sufficiently small values of ϵ the matrix $A + \epsilon E$ has a simple eigenvalue $\lambda_j(\epsilon)$ in $\mathcal{D}$. If we denote a corresponding eigenvector $x_j(\epsilon)$ then

$$(A + \epsilon E)x_j(\epsilon) = \lambda_j(\epsilon)x_j(\epsilon).$$

Using results from function theory, it can be shown that $\lambda_j(\epsilon)$ and $x_j(\epsilon)$ are analytic functions of ϵ for $\epsilon < \epsilon_0$. Differentiating with respect to ϵ and putting $\epsilon = 0$ we get

$$(A - \lambda_j I)x_j'(0) + Ex_j = \lambda_j'(0)x_j. \tag{9.2.6}$$

Since $y_j^H(A - \lambda_j I) = 0$ we can eliminate $x_j'(0)$ by multiplying this equation with y_j^H and solve for $\lambda_j'(0) = y_j^H E x_j / y_j^H x_j$. □

If $\|E\|_2 = 1$ we have $|y_j^H E x_j| \le \|x_j\|_2 \|y_j\|_2$ and E can always be chosen so that equality holds. If we also normalize so that $\|x_j\|_2 = \|y_j\|_2 = 1$, then $1/s(\lambda_j)$, where

$$s(\lambda_j) = |y_j^H x_j| \tag{9.2.7}$$

can be taken as *the condition number of the simple eigenvalue* λ_j. Note that $s(\lambda_j) = \cos\theta(x_j, y_j)$, where $\theta(x_j, y_j)$ is the acute angle between the left and right eigenvector corresponding to λ_j. If A is a normal matrix we get $s(\lambda_j) = 1$.

The above theorem shows that for perturbations in A of order ϵ, a simple eigenvalue λ of A will be perturbed by an amount approximately equal to $\epsilon/s(\lambda)$. If λ is a defective eigenvalue, then there is no similar result. *Indeed, if the largest Jordan block corresponding to λ is of order k, then perturbations to λ of order $\epsilon^{1/k}$ can be expected.* Note that for a Jordan box we have $x = e_1$ and $y = e_m$ and so $s(\lambda) = 0$ in (9.2.7).

Example 9.2.4.

Consider the perturbed diagonal matrix

$$A + \epsilon E = \begin{pmatrix} 1 & \epsilon & 2\epsilon \\ \epsilon & 2 & \epsilon \\ \epsilon & 2\epsilon & 2 \end{pmatrix}.$$

Here A is diagonal with left and right eigenvector equal to $x_i = y_i = e_i$. Thus, $y_i^H E x_i = e_{ii} = 0$ and the first order term in the perturbation of the simple eigenvalues are zero. For $\epsilon = 10^{-3}$ the eigenvalues of $A + E$ are

$$0.999997, \quad 1.998586, \quad 2.001417.$$

Hence, the perturbation in the simple eigenvalue λ_1 is of order 10^{-6}. Note that the Bauer–Fike theorem would predict perturbations of order 10^{-3} for all three eigenvalues.

We now consider the perturbation of an eigenvector x_j corresponding to a simple eigenvalue λ_j. Assume that the matrix A is diagonalizable and that $x_1, \ldots, x_n$ are linearly independent eigenvectors. Then we can write

$$x_j(\epsilon) = x_j + \epsilon x_j'(0) + O(\epsilon^2), \qquad x_j'(0) = \sum_{k \ne j} c_{kj} x_k,$$

where we have normalized $x_j(\epsilon)$ to have unit component along x_j. Substituting the expansion of $x_j'(0)$ into (9.2.6) we get

$$\sum_{k \ne j} c_{kj}(\lambda_k - \lambda_j) x_k + E x_j = \lambda_j'(0) x_j.$$

Multiplying by y_i^H and using $y_i^H x_j = 0$, $i \neq j$, we obtain

$$c_{ij} = \frac{y_i^H E x_j}{(\lambda_j - \lambda_i) y_i^H x_i}, \quad i \neq j. \tag{9.2.8}$$

Hence, the sensitivity of the eigenvectors depend also on the separation $\delta_j = \min_{i \neq j} |\lambda_i - \lambda_j|$ between λ_j and the rest of the eigenvalues of A. If several eigenvectors corresponds to a multiple eigenvalue these are not uniquely determined, which is consistent with this result. Note that even if the individual eigenvectors are sensitive to perturbations it may be that an invariant subspace containing these eigenvectors is well determined.

To measure the accuracy of computed invariant subspaces we need to introduce the largest angle between two subspaces.

Definition 9.2.6. *Let $\mathcal{X}$ and $\mathcal{Y} = \mathcal{R}(Y)$ be two subspaces of $\mathbf{C}^n$ of dimension k. Define the largest angle between these subspaces to be*

$$\theta_{\max}(\mathcal{X}, \mathcal{Y}) = \max_{\substack{x \in \mathcal{X} \\ \|x\|_2 = 1}} \min_{\substack{y \in \mathcal{Y} \\ \|y\|_2 = 1}} \theta(x, y). \tag{9.2.9}$$

where $\theta(x, y)$ is the acute angle between x and y.

The quantity $\sin \theta_{\max}(\mathcal{X}, \mathcal{Y})$ defines a distance between the two subspaces $\mathcal{X}$ and $\mathcal{Y}$. If X and Y are orthonormal matrices such that $\mathcal{X} = \mathcal{R}(X)$ and $\mathcal{Y} = \mathcal{R}(Y)$, then it can be shown (see Golub and Van Loan [184]) that

$$\theta(\mathcal{X}, \mathcal{Y}) = \arccos \sigma_{min}(X^H Y). \tag{9.2.10}$$

9.2.3 Hermitian Matrices

We have seen that the eigenvalues of Hermitian, and real symmetric matrices are all real, and from Theorem 9.2.5 it follows that these eigenvalues are perfectly conditioned. For this class of matrices it is possible to get more informative perturbation bounds, than those given above. In this section we give several classical theorems. They are all related to each other, and the interlace theorem dates back to Cauchy, 1829. We assume in the following that the eigenvalues of A have been ordered in decreasing order $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

In the particular case of a Hermitian matrix the extreme eigenvalues λ_1 and λ_n can be characterized by

$$\lambda_1 = \max_{\substack{x \in C^n \\ x \neq 0}} \rho(x), \quad \lambda_n = \min_{\substack{x \in C^n \\ x \neq 0}} \rho(x).$$

The following theorem gives an important extremal characterization also of the intermediate eigenvalues of a Hermitian matrix.

Theorem 9.2.7 (*Fischer's Theorem*).

Let the Hermitian matrix A have eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ ordered so that $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n$. Then

$$\lambda_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x\in\mathcal{S}\\ x\neq 0}} \frac{x^H A x}{x^H x} \tag{9.2.11}$$

$$= \min_{\dim(\mathcal{S})=n-i+1} \max_{\substack{x\in\mathcal{S}\\ x\neq 0}} \frac{x^H A x}{x^H x}. \tag{9.2.12}$$

where $\mathcal{S}$ denotes a subspace of $\mathbf{C}^n$.

Proof. See Stewart [340, 1973, p. 314]. □

The formulas (9.2.11) and (9.2.12) are called the max-min and the min-max characterization, respectively. They can be used to establish an important relation between the eigenvalues of two Hermitian matrices A and B, and their sum $C = A + B$.

Theorem 9.2.8.

Let $\alpha_1 \ge \alpha_2 \ge \cdots \ge \alpha_n$, $\beta_1 \ge \beta_2 \ge \cdots \ge \beta_n$, and $\gamma_1 \ge \gamma_2 \ge \cdots \ge \gamma_n$ be the eigenvalues of the Hermitian matrices A, B, and $C = A + B$. Then

$$\alpha_i + \beta_1 \ge \gamma_i \ge \alpha_i + \beta_n, \quad i = 1 : n. \tag{9.2.13}$$

Proof. Let $x_1, x_2, \dots, x_n$ be an orthonormal system of eigenvectors of A corresponding to $\alpha_1 \ge \alpha_2 \ge \cdots \ge \alpha_n$, and let $\mathcal{S}$ be the subspace of $\mathbf{C}^n$ spanned by $x_1, \dots, x_i$. Then by Fischer's theorem

$$\gamma_i \ge \min_{\substack{x\in\mathcal{S}\\ x\neq 0}} \frac{x^H C x}{x^H x} \ge \min_{\substack{x\in\mathcal{S}\\ x\neq 0}} \frac{x^H A x}{x^H x} + \min_{\substack{x\in\mathcal{S}\\ x\neq 0}} \frac{x^H B x}{x^H x} = \alpha_i + \min_{\substack{x\in\mathcal{S}\\ x\neq 0}} \frac{x^H B x}{x^H x} \ge \alpha_i + \beta_n.$$

This is the last inequality of (9.2.12). The first equality follows by applying this result to $A = C + (-B)$. □

The theorem implies that when B is added to A all of its eigenvalues are changed by an amount which lies between the smallest and greatest eigenvalues of B. If the matrix rank$(B) < n$, the result can be sharpened, see Parlett [305, Sec. 10-3]. An important case is when $B = \pm zz^T$ is a rank one matrix. Then B has only one nonzero eigenvalue equal to $\rho = \pm\|z\|_2^2$. In this case the perturbed eigenvalues will satisfy the relations

$$\lambda_i' - \lambda_i = m_i\rho, \quad 0 \le m_i, \quad \sum m_i = 1. \tag{9.2.14}$$

Hence, all eigenvalues are shifted by an amount which lies between zero and ρ.

An important application is to get bounds for the eigenvalues λ_i' of $A+E$, where A and E are Hermitian matrices. Usually the eigenvalues of E are not known, but from

$$\max\{|\lambda_1(E)|, |\lambda_n(E)|\} = \rho(E) = \|E\|_2$$

it follows that

$$|\lambda_i - \lambda_i'| \le \|E\|_2. \tag{9.2.15}$$

Note that this result also holds for *large perturbations*.

A related result is the **Wielandt–Hoffman theorem** which states that

$$\sqrt{\textstyle\sum_{i=1}^{n} |\lambda_i - \lambda_i'|^2} \le \|E\|_F. \tag{9.2.16}$$

An elementary proof of this result is given by Wilkinson [387, Sec. 2.48].

Another important result that follows from Fischer's Theorem is the following theorem, due to Cauchy, which relates the eigenvalues of a principal submatrix to the eigenvalues of the original matrix.

Theorem 9.2.9 (*Interlacing Property*)**.**

Let A_{n-1} be a principal submatrix of order $n-1$ of a Hermitian matrix $A_n \in \mathbf{C}^{n\times n}$, Then, the eigenvalues of A_{n-1}, $\mu_1 \ge \mu_2 \ge \cdots \ge \mu_{n-1}$ interlace the eigenvalues $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n$ of A_n, that is

$$\lambda_i \ge \mu_i \ge \lambda_{i+1}, \quad i = 1 : n-1. \tag{9.2.17}$$

Proof. Without loss of generality we assume that A_{n-1} is the leading principal submatrix of A,

$$A_n = \begin{pmatrix} A_{n-1} & a^H \\ a & \alpha \end{pmatrix}.$$

Consider the subspace of vectors $\mathcal{S}' = \{x \in \mathbf{C}^n, x \perp e_n\}$. Then with $x \in \mathcal{S}'$ we have $x^H A_n x = (x')^H A_{n-1} x'$, where $x^H = ((x')^H, 0)$. Using the minimax characterization (9.2.11) of the eigenvalue λ_i it follows that

$$\lambda_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x\in\mathcal{S}\\ x\ne 0}} \frac{x^H A_n x}{x^H x} \ge \max_{\substack{dim(\mathcal{S})=i\\ \mathcal{S}\perp e_n}} \min_{\substack{x\in\mathcal{S}\\ x\ne 0}} \frac{x^H A_n x}{x^H x} = \mu_i.$$

The proof of the second inequality $\mu_i \ge \lambda_{i+1}$ is obtained by a similar argument applied to $-A_n$. $\square$

Since any principal submatrix of a Hermitian matrix also is Hermitian, this theorem can be used recursively to get relations between the eigenvalues of A_{n-1} and A_{n-2}, A_{n-2} and A_{n-3}, etc.

It is sometimes desirable to determine the eigenvalues of a diagonal matrix modifies by a symmetric matrix of rank one.

Theorem 9.2.10.

Let $D = \operatorname{diag}(d_i)$ and $z = (z_1, \dots, z_n)^T$. If $\lambda \neq d_i$, $i = 1 : n$, then the eigenvalues of $D + \mu z z^T$ are the roots of the **secular equation**

$$\psi(\lambda) = 1 + \mu \sum_{i=1}^{n} \frac{z_i^2}{d_i - \lambda} = 0. \tag{9.2.18}$$

The eigenvalues λ_i interlace the elements d_i so that if $\mu \geq 0$ then

$$d_1 \leq \lambda_1 \leq d_2 \leq \lambda_2 \leq \cdots \leq d_n \leq \lambda_n \leq d_n + \mu \|z\|_2^2. \tag{9.2.19}$$

The eigenvector x_i corresponding to λ_i satisfies $(D - \lambda_i)x_i + \mu z(z^T x_i) = 0$ and hence if $\lambda_i \neq d_j$, $j = 1 : n$,

$$x_i = (D - \lambda_i)^{-1} z / \|(D - \lambda_i)^{-1} z\|_2,$$

is a unit eigenvector.

Proof. By the assumption the matrix $(D - \lambda I)$ is nonsingular, and hence

$$\det(D + \mu z z^T - \lambda I) = \det\big((D - \lambda I)\big) \det\big(I + (D - \lambda I)^{-1} \mu z z^T)\big).$$

Since $\det(D - \lambda I) \neq 0$ we conclude, using the identity $\det(I + xy^T) = 1 + y^T x$, that the eigenvalues satisfy

$$\det\big(I + (D - \lambda I)^{-1} \mu z z^T)\big) = 0,$$

which gives (9.2.18). The interlacing property (9.2.19) follows form Fischer's Theorem 9.2.8. $\square$

For an application of these results, see Sec. 9.5.4.

The following perturbation result due to Demmel and Kahan [96] shows the remarkable fact that all singular values of a bidiagonal matrix are determined to *full relative precision independent of their magnitudes.*

Theorem 9.2.11.

Let $B \in \mathbf{R}^{n \times n}$ be a bidiagonal matrix with singular values $\sigma_1 \geq \cdots \geq \sigma_n$. Let $|\delta B| \leq \omega |B|$, and let $\bar{\sigma}_1 \geq \cdots \geq \bar{\sigma}_n$ be the singular values of $\bar{B} = B + \delta B$. Then if $\eta = (2n - 1)\omega < 1$,

$$|\bar{\sigma}_i - \sigma_i| \leq \frac{\eta}{1 - \eta} |\sigma_i|, \tag{9.2.20}$$

$$\max\{\sin\theta(u_i, \tilde{u}_i), \sin\theta(v_i, \tilde{v}_i)\} \leq \frac{\sqrt{2}\eta(1 + \eta)}{\mathrm{relgap}_i - \eta}, \tag{9.2.21}$$

$i = 1 : n$, where the **relative gap** *between singular values is*

$$\mathrm{relgap}_i = \min_{j \neq i} \frac{|\sigma_i - \sigma_j|}{\sigma_i + \sigma_j}. \tag{9.2.22}$$

9.2.4 The Rayleigh Quotient and Residual Bounds

We make the following definition.

Definition 9.2.12.

The **Rayleigh quotient** *of a nonzero vector* $x \in \mathbf{C}^n$ *is the (complex) scalar*

$$\rho(x) = \rho(A, x) = \frac{x^H A x}{x^H x}. \tag{9.2.23}$$

The Rayleigh quotient plays an important role in the computation of eigenvalues and eigenvectors. The Rayleigh quotient is a homogeneous function of x, $\rho(\alpha x) = \rho(x)$ for all scalar $\alpha \neq 0$.

Definition 9.2.13.

The **field of values** *of a matrix* A *is the set of all possible Rayleigh quotients*

$$F(A) = \{\rho(A, x) \mid x \in C^n\}.$$

For any unitary matrix U we have $F(U^H A U) = F(A)$. From the Schur canonical form it follows that there is no restriction in assuming A to be upper triangular, and, if normal, then diagonal. Hence, for a normal matrix A

$$\rho(x) = \sum_{i=1}^{n} \lambda_i |\xi_i|^2 \Big/ \sum_{i=1}^{n} |\xi_i|^2,$$

that is any point in $F(A)$ is a weighted mean of the eigenvalues of A. Thus, for a normal matrix the field of values coincides with the convex hull of the eigenvalues. In the special case of a Hermitian matrix the field of values equals the segment $[\lambda_1, \lambda_n]$ of the real axis.

In general the field of values of a matrix A may contain complex values even if its eigenvalues are real. However, the field of values will always contain the convex hull of the eigenvalues.

Let x and A be given and consider the problem

$$\min_{\mu} \|Ax - \mu x\|_2^2.$$

This is a linear least squares problem for the unknown μ. The normal equations are $x^H x \mu = x^H A x$. Hence, the minimum is attained for $\rho(x)$, the Rayleigh quotient of x.

When A is Hermitian the gradient of $\frac{1}{2}\rho(x)$ is

$$\frac{1}{2}\nabla\rho(x) = \frac{Ax}{x^H x} - \frac{x^H A x}{(x^H x)^2} x = \frac{1}{x^H x}(Ax - \rho x),$$

and hence the Rayleigh quotient $\rho(x)$ is stationary if and only if x is an eigenvector of A.

Suppose we have computed by some method an approximate eigenvalue/eigenvector pair (σ, v) to a matrix A. In the following we derive some error bounds depending on the **residual vector**

$$r = Av - \sigma v.$$

Since $r = 0$ if (σ, v) are an exact eigenpair it is reasonable to assume that the size of the residual r measures the accuracy of v and σ. We show a simple backward error bound:

Theorem 9.2.14.

Let $\bar{\lambda}$ and $\bar{x}$, $\|\bar{x}\|_2 = 1$, be a given approximate eigenpair of $A \in \mathbf{C}^{n\times n}$, and $r = A\bar{x} - \bar{\lambda}\bar{x}$ be the corresponding residual vector. Then $\bar{\lambda}$ and $\bar{x}$ is an exact eigenpair of the matrix $A + E$, where

$$E = -r\bar{x}^H, \qquad \|E\|_2 = \|r\|_2. \tag{9.2.24}$$

Proof. We have $(A + E)\bar{x} = (A - r\bar{x}^H/\bar{x}^H\bar{x})\bar{x} = A\bar{x} - r = \bar{\lambda}\bar{x}$. □

It follows that given an approximate eigenvector $\bar{x}$ a good eigenvalue approximation is the Rayleigh quotient $\rho(\bar{x})$, since this choice minimizes the error bound in Theorem 9.2.14.

By combining Theorems 9.2.4 and 9.2.14 we obtain for a Hermitian matrix A the very useful a posteriori error bound

Corollary 9.2.15. *Let A be a Hermitian matrix. For any $\bar{\lambda}$ and any unit vector $\bar{x}$ there is an eigenvalue of λ of A such that*

$$|\lambda - \bar{\lambda}| \le \|r\|_2, \qquad r = A\bar{x} - \bar{\lambda}\bar{x}. \tag{9.2.25}$$

For a fixed $\bar{x}$, the error bound is minimized by taking $\bar{\lambda} = \bar{x}^T A\bar{x}$.

This shows that $(\bar{\lambda}, \bar{x})$ ($\|\bar{x}\|_2 = 1$) is a numerically acceptable eigenpair of the Hermitian matrix A if $\|A\bar{x} - \lambda\bar{x}\|_2$ is of order machine precision.

For a Hermitian matrix A, the Rayleigh quotient $\rho(x)$ may be a far more accurate approximate eigenvalue than x is an approximate eigenvector. The following theorem shows that if an eigenvector is known to precision ϵ, the Rayleigh quotient approximates the corresponding eigenvalue to precision ϵ^2.

Theorem 9.2.16.

Let the Hermitian matrix A have eigenvalues $\lambda_1, \ldots, \lambda_n$ and orthonormal eigenvectors $x_1, \ldots, x_n$. If the vector $x = \sum_{i=1}^n \xi_i x_i$, satisfies

$$\|x - \xi_1 x_1\|_2 \le \epsilon\|x\|_2. \tag{9.2.26}$$

then

$$|\rho(x) - \lambda_1| \le 2\|A\|_2\epsilon^2. \tag{9.2.27}$$

Proof. Writing $Ax = \sum_{i=1}^n \xi_i\lambda_i x_i$, the Rayleigh quotient becomes

$$\rho(x) = \sum_{i=1}^n |\xi_i|^2\lambda_i \Big/ \sum_{i=1}^n |\xi_i|^2 = \lambda_1 + \sum_{i=2}^n |\xi_i|^2(\lambda_i - \lambda_1) \Big/ \sum_{i=1}^n |\xi_i|^2.$$

Using (9.2.26) we get $|\rho(x)-\lambda_1| \le \max_i |\lambda_i - \lambda_1|\epsilon^2$. Since the matrix A is Hermitian we have $|\lambda_i| \le \sigma_1(A) = \|A\|_2$, $i = 1 : n$, and the theorem follows. □

Stronger error bounds can be obtained if $\sigma = \rho(v)$ is known to be well separated from all eigenvalues except λ.

Theorem 9.2.17.

Let A be a Hermitian matrix with eigenvalues $\lambda(A) = \{\lambda_1, \dots, \lambda_n\}$, x a unit vector and $\rho(x)$ its Rayleigh quotient. Let $Az = \lambda_\rho z$, where λ_ρ is the eigenvalue of A closest to $\rho(x)$. Define

$$\operatorname{gap}(\rho) = \min_{\lambda \in \lambda(A)} |\lambda - \rho|, \quad \lambda \ne \lambda_\rho. \tag{9.2.28}$$

Then it holds that

$$|\lambda_\rho - \rho(x)| \le \|Ax - x\rho\|_2^2 / \operatorname{gap}(\rho), \tag{9.2.29}$$

$$\sin\theta(x, z) \le \|Ax - x\rho\|_2 / \operatorname{gap}(\rho). \tag{9.2.30}$$

Proof. See Parlett [305, Sec. 11.7]. □

Example 9.2.5.

With $x = (1,\ 0)^T$ and

$$A = \begin{pmatrix} 1 & \epsilon \\ \epsilon & 0 \end{pmatrix}, \text{ we get } \rho = 1, \quad Ax - x\rho = \begin{pmatrix} 0 \\ \epsilon \end{pmatrix}.$$

From Corollary 9.2.15 we get $|\lambda - 1| \le \epsilon$, whereas Theorem 9.2.17 gives the improved bound $|\lambda - 1| \le \epsilon^2/(1-\epsilon^2)$.

Often gap(σ) is not known and the bounds in Theorem 9.2.17 are only theoretical. In some methods, e.g., the method of spectrum slicing (see Sec. 9.3.2) an interval around σ can be determined which contain no eigenvalues of A.

9.2.5 Residual Bounds for SVD

The singular values of a matrix $A \in \mathbf{R}^{m\times n}$ equal the positive square roots of the eigenvalues of the symmetric matrix A^TA and AA^T. Another very useful relationship between the SVD of $A = U\Sigma V^T$ and a symmetric eigenvalue was given in Theorem 7.3.2. If A is square, then[35]

$$C = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} U & U \\ V & -V \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} U & U \\ V & -V \end{pmatrix}^T \tag{9.2.31}$$

[35] This assumption is no restriction since we can always adjoin zero rows (columns) to make A square.

Using these relationships the theory developed for the symmetric (Hermitian) eigenvalue problem in Secs. 9.2.3–9.2.4 applies also to the singular value decomposition. For example, Theorems 8.3.3–8.3.5 are straightforward applications of Theorems 9.2.7–9.2.9.

We now consider applications of the Rayleigh quotient and residual error bounds given in Section 9.2.4. If u, v are unit vectors the Rayleigh quotient of C is

$$\rho(u,v) = \frac{1}{\sqrt{2}}(u^T, v^T)\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} u \\ v \end{pmatrix} = u^T A v, \tag{9.2.32}$$

From Corollary 9.2.15 we obtain the following error bound.

Theorem 9.2.18. *For any scalar α and unit vectors u, v there is a singular value σ of A such that*

$$|\sigma - \alpha| \le \frac{1}{\sqrt{2}}\left\| \begin{pmatrix} Av - u\alpha \\ A^T u - v\alpha \end{pmatrix} \right\|_2. \tag{9.2.33}$$

For fixed u, v this error bound is minimized by taking $\alpha = u^T A v$.

The following theorem is an application to Theorem 9.2.17.

Theorem 9.2.19.

Let A have singular values σ_i, $i = 1 : n$. Let u and v be unit vectors, $\rho = u^T A v$ the corresponding Rayleigh quotient, and

$$\delta = \frac{1}{\sqrt{2}}\left\| \begin{pmatrix} Av - u\rho \\ A^T u - v\rho \end{pmatrix} \right\|_2$$

the residual norm. If σ_s is the closest singular value to ρ and $Au_s = \sigma_s v_s$, then

$$|\sigma_s - \rho(x)| \le \delta^2/\text{gap}\,(\rho), \tag{9.2.34}$$

$$\max\{\sin\theta(u_s, u), \sin\theta(v_s, v)\} \le \delta/\text{gap}\,(\rho). \tag{9.2.35}$$

where

$$\text{gap}\,(\rho) = \min_{i \ne s} |\sigma_i - \rho|. \tag{9.2.36}$$

Review Questions

2.1 State Gerschgorin's Theorem, and discuss how it can be sharpened.

2.2 Discuss the sensitivity to perturbations of eigenvalues and eigenvectors of a Hermitian matrix A.

2.3 Suppose that $(\bar{\lambda}, \bar{x})$ is an approximate eigenpair of A. Give a backward error bound. What can you say of the error in $\bar{\lambda}$ if A is Hermitian?

2.4 (a) Tell the minimax and maximin properties of the eigenvalues (of what kind of matrices?), and the related properties of the singular values (of what kind of matrices?).

(b) Show how the theorems in (a) can be used for deriving an interlacing property for the eigenvalues of a matrix in $\mathbf{R}^{n\times n}$ (of what kind?) and the eigenvalues of its principal submatrix in $\mathbf{R}^{(n-1)\times(n-1)}$.

Problems

2.1 An important problem is to decide if all the eigenvalues of a matrix A have negative real part. Such a matrix is called **stable**. Show that if

$$\mathrm{Re}(a_{ii}) + r_i \le 0, \quad \forall i,$$

and $\mathrm{Re}(a_{ii}) + r_i < 0$ for at least one i, then the matrix A is stable if A is irreducible.

2.2 Suppose that the matrix A is real, and all Gerschgorin discs of A are distinct. Show that from Theorem 9.2.2 it follows that all eigenvalues of A are real.

2.3 Show that all eigenvalues to a matrix A lie in the union of the disks

$$|z - a_{ii}| \le \frac{1}{d_i} \sum_{j=1, j\neq i}^{n} d_j |a_{ij}|, \qquad i = 1:n,$$

where d_i, $i = 1:n$ are given positive scale factors.

Hint: Use the fact that the eigenvalues are invariant under similarity transformations.

2.4 Let $A \in \mathbf{C}^{n\times n}$, and assume that $\epsilon = \max_{i\neq j} |a_{ij}|$ is small. Choose the diagonal matrix $D = \mathrm{diag}\,(\mu, 1, \ldots, 1)$ so that the first Gerschgorin disk of DAD^{-1} is as small as possible, without overlapping the other disks. Show that if the diagonal elements of A are distinct then

$$\mu = \frac{\epsilon}{\delta} + O(\epsilon^2), \quad \delta = \min_{i\neq 1} |a_{ii} - a_{11}|,$$

and hence the first Gerschgorin disk is given by

$$|\lambda - a_{11}| \le r_1, \quad r_1 \le (n-1)\epsilon^2/\delta + O(\epsilon^3).$$

2.5 Compute the eigenvalues of B and A, where

$$B = \begin{pmatrix} 0 & \epsilon \\ \epsilon & 0 \end{pmatrix}, \qquad A = \begin{pmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Show that they interlace.

2.6 Use a suitable diagonal similarity and Gerschgorin's theorem to show that the eigenvalues of the tridiagonal matrix

$$T = \begin{pmatrix} a & b_2 & & & \\ c_2 & a & b_3 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & a & b_n \\ & & & c_n & a \end{pmatrix}.$$

satisfy the inequality

$$|\lambda - a| < 2\sqrt{\max_i |b_i| \max_i |c_i|}.$$

2.7 Let A and B be square Hermitian matrices and

$$H = \begin{pmatrix} A & C \\ C^H & B \end{pmatrix}.$$

Show that for every eigenvalue $\lambda(B)$ of B there is an eigenvalue $\lambda(H)$ of H such that

$$|\lambda(H) - \lambda(B)| \le (\|C^H C\|_2)^{1/2}.$$

Hint: Use the estimate (9.2.25).

9.3 The Power Method

9.3.1 Iteration with a Single Vector

One of the oldest methods for computing eigenvalues and eigenvectors of a matrix is the **power method**. For a long time the power method was the only alternative for finding the eigenvalues of a general non-Hermitian matrix. It is still one of the few practical methods when the matrix A is very large and sparse. Although it is otherwise no longer much used in its basic form for computing eigenvalues it is central to the convergence analysis of many currently used algorithms. A variant of the power method is also a standard method for computing eigenvectors when an accurate approximation to the corresponding eigenvalue is known.

For a given matrix $A \in \mathbf{C}^{n\times n}$ and starting vector $q \neq 0$ the power method forms the sequence of vectors q, Aq, A^2q, $A^3q, \ldots$, using the recursion

$$q_0 = q, \qquad q_k = Aq_{k-1}, \quad k = 1, 2, 3, \ldots.$$

Note that this only involves matrix vector multiplications and that the matrix powers A^k are never computed.

To simplify the analysis we assume that the matrix A is semisimple, that is, it has a set of linearly independent eigenvectors $x_1, x_2, \ldots, x_n$ associated with the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. We further assume in the following that the eigenvalues are ordered so that

$$|\lambda_1| \ge |\lambda_2| \ge \cdots \ge |\lambda_n|.$$

Then the initial vector q_0 can be expanded along the eigenvectors x_i of A, $q_0 = \sum_{j=1}^{n} \alpha_j x_j$, and we have

$$q_k = \sum_{j=1}^{n} \lambda_j^k \alpha_j x_j = \lambda_1^k \Big(\alpha_1 x_1 + \sum_{j=2}^{n} \Big(\frac{\lambda_j}{\lambda_1}\Big)^k \alpha_j x_j \Big), \quad k = 1, 2, \cdots$$

If λ_1 is a unique eigenvalue of maximum magnitude, $|\lambda_1| > |\lambda_2|$, we say that λ_1 is a **dominant eigenvalue**. If $\alpha_1 \neq 0$, then

$$\frac{1}{\lambda_1^k} q_k = \alpha_1 x_1 + O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right), \tag{9.3.1}$$

and up to a factor λ_1^k the vector q_k will converge to a limit vector which is an eigenvector associated with the dominating eigenvalue λ_1. The *rate of convergence is linear and equals* $|\lambda_2|/|\lambda_1|$. One can show that this result holds also when A is not diagonalizable by writing q_0 as a linear combination of the vectors associated with the Jordan (or Schur) canonical form of A, see Theorems 9.1.9 and 9.1.11.

In practice the vectors q_k have to be normalized in order to avoid overflow or underflow. Hence, assume that $\|q_0\|_\infty = 1$, and modify the initial recursion as follows:

$$\hat{q}_k = A q_{k-1}, \quad \mu_k = \|\hat{q}_k\|_\infty, \quad q_k = \hat{q}_k / \mu_k, \quad k = 1, 2, \dots. \tag{9.3.2}$$

Then under the above assumptions

$$q_k = \frac{1}{\gamma_k} A^k q_0, \quad \gamma_k = \mu_1 \cdots \mu_k,$$

and q_k converges to a normalized eigenvector x_1. From (9.3.1) it follows that

$$\hat{q}_k = \lambda_1 q_{k-1} + O\big(|\lambda_2/\lambda_1|^k\big), \tag{9.3.3}$$

and $\lim_{k\to\infty} \mu_k = |\lambda_1|$. Convergence is slow when $|\lambda_2| \approx |\lambda_1|$, but can be accelerated by Aitken extrapolation; see Volume I, Sec. 3.4.2.

For a real symmetric matrix A the eigenvalues are real and the eigenvectors can be chosen to be real and orthogonal. Using (9.3.1) it follows that the Rayleigh quotients

$$\rho(q_{k-1}) = q_{k-1}^T A q_{k-1} = q_{k-1}^T \hat{q}_k.$$

will converge twice as fast as μ_k,

$$\lambda_1 = \rho(q_{k-1}) + O\big(|\lambda_2/\lambda_1|^{2k}\big), \tag{9.3.4}$$

Example 9.3.1.

The eigenvalues of the matrix

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 4 \end{pmatrix}$$

are (4.732051, 3, 1.267949), correct to 6 decimals. If we take $q_0 = (1, 1, 1)^T$ then we obtain the Rayleigh quotients ρ_k and errors $e_k = \lambda_1 - \rho_k$ are given in the table below. The ratios of successive errors converge to $(\lambda_2/\lambda_1)^2 = 0.4019$.

k	ρ_k	e_k	e_k/e_{k-1}
1	4.333333	0.398718	
2	4.627119	0.104932	0.263
3	4.694118	0.037933	0.361
4	4.717023	0.015027	0.396
5	4.729620	0.006041	0.402

Convergence of the power method can be shown only for *almost all starting vectors* since it depends on the assumption that $\alpha_1 \neq 0$, However, when $\alpha_1 = 0$, rounding errors will tend to introduce a small component along x_1 in Aq_0 and, in practice, the method converges also in this case.

Convergence of the power method can also be shown under the weaker assumption that $\lambda_1 = \lambda_2 = \cdots = \lambda_r$, and

$$|\lambda_r| > |\lambda_{r+1}| \geq \cdots \geq |\lambda_n|.$$

In this case the iterates will converge to *one particular vector in the invariant subspace* span$[x_1, \ldots, x_r]$. The limit vector will depend upon the initial vector.

If the eigenvalue λ of largest magnitude of a real matrix is complex, then the complex conjugate $\bar{\lambda}$ must also be an eigenvalue. A modification of the power method for this particular case is given in Fröberg [142, p. 194]. A modification for the case when there are two dominant eigenvalues of opposite sign, $\lambda_1 = -\lambda_2$ is given in Problem 9.3.2.

An attractive feature of the power method is that the matrix A is not explicitly needed. It suffices to be able to form the matrix times vector product Aq for any given vector q. It may not even be necessary to explicitly store the nonzero elements of A. Therefore, the power method can be very useful for computing the dominant eigenvalue and the associated eigenvector when the matrix A is very large.

A simple modification of the power method is to apply the power method to $(A - pI)$, where p is a **shift of origin**. Suppose that A and all its eigenvalues λ_i are real and that $\lambda_1 > \lambda_2 > \cdots > \lambda_n$. Then for the shifted matrix either $\lambda_1 - p$ or $\lambda_n - p$ is the dominant eigenvalue. For convergence to x_1 the shift $p = \frac{1}{2}(\lambda_2 + \lambda_n)$ is optimal. The rate of convergence is then governed by

$$\frac{\lambda_2 - p}{\lambda_1 - p} = \frac{\lambda_2 - \lambda_n}{2\lambda_1 - (\lambda_2 + \lambda_n)}.$$

Similarly, for convergence to x_n the shift $p = \frac{1}{2}(\lambda_1 + \lambda_{n-1})$ is optimal. Unfortunately the improvement in convergence using this device is often quite small.

So far we have neglected the effect of rounding errors in the power method. These errors will limit the accuracy which can be attained. If we include rounding errors we will in (9.3.2) compute

$$\mu_k q_k = Aq_{k-1} + f_k,$$

where f_k is a small error vector. If Aq_{k-1} is computed using floating point arithmetic we have (see ??)

$$fl(Aq_{k-1}) = (A + F_k)q_{k-1}, \quad |F_k| < \gamma_n |A|.$$

We can not guarantee any progress after having reached vector q_k, which is an exact eigenvector of some matrix $(A + G)$.

9.3.2 Deflation of Eigenproblems

The simple power method can be used for computing several eigenvalues and the associated eigenvectors by combining it with **deflation**. By that we mean a method that given an eigenvector x_1 and the corresponding eigenvalue λ_1 computes a matrix A_1 such that $\lambda(A) = \lambda_1 \cup \lambda(A_1)$. A way to construct such a matrix A_1 in a stable way was indicated in Sec. 9.1, see (9.1.13). However, this method has the drawback that even if A is sparse the matrix A_1 will in general be dense.

The following simple method for deflation is due to Hotelling. Suppose an eigenpair (λ_1, x_1), $\|x_1\|_2 = 1$, of a symmetric matrix A is known. If we define $A_1 = A - \lambda_1 x_1 x_1^H$, then from the orthogonality of the eigenvectors x_i we have

$$A_1 x_i = Ax_i - \lambda_1 x_1 (x_1^T x_i) = \begin{cases} 0, & \text{if } i = 1; \\ \lambda_i x_i, & \text{if } i \neq 1. \end{cases}$$

Hence, the eigenvalues of A_1 are $0, \lambda_2, \ldots, \lambda_n$ with corresponding eigenvectors equal to $x_1, x_2, \ldots, x_n$. The power method can now be applied to A_1 to determine the dominating eigenvalue of A_1. Note that $A_1 = A - \lambda_1 x_1 x_1^T = (I - x_1 x_1^T)A = P_1 A$, where P_1 is an orthogonal projection.

When A is unsymmetric there is a corresponding deflation technique. Here it is necessary to have the left eigenvector y_1^T as well as the right x_1. If these are normalized so that $y_1^T x_1 = 1$, then we define A_1 by $A_1 = A - \lambda_1 x_1 y_1^T$. From the biorthogonality of the x_i and y_i we have

$$A_1 x_i = Ax_i - \lambda_1 x_1 (y_1^T x_i) = \begin{cases} 0, & \text{if } i = 1; \\ \lambda_i x_i, & \text{if } i \neq 1. \end{cases}$$

In practice an important advantage of this scheme is that it is not necessary to form the matrix A_1 explicitly. The power method, as well as many other methods, only requires that an operation of the form $y = A_1 x$ can be performed. This operation can be performed as

$$A_1 x = Ax - \lambda_1 x_1 (y_1^T x) = Ax - \tau x_1, \qquad \tau = \lambda_1 (y_1^T x).$$

Hence, it suffices to have the vectors x_1, y_1 available as well as a procedure for computing Ax for a given vector x. Obviously this deflation procedure can be performed repeatedly, to obtain $A_2, A_3, \ldots$.

This deflation procedure has to be used with caution, since errors will accumulate. This can be disastrous in the nonsymmetric case, when the eigenvalues may be badly conditioned.

9.3.3 Spectral Transformation and Inverse Iteration

The simple power method has the drawback that convergence may be arbitrarily slow or may not happen at all. To overcome this difficulty we can use a **spectral transformation**,, which we now describe. Let A have an eigenvalues λ_j with eigenvector x_j. Recall that if $p(x)$ and $q(x)$ are two polynomials such that $q(A)$ is nonsingular, then $r(\lambda_j)$ is an eigenvalue and x_j an eigenvector of $r(A) = (q(A))^{-1}p(A)$.

As a simple application assume that A is nonsingular and take $r(x) = x^{-1}$. Then $r(A) = A^{-1}$ has eigenvalues $1/\lambda_j$. Hence, (9.3.3) shows that if the eigenvalues of A satisfy

$$|\lambda_1| \geq \cdots \geq |\lambda_{n-1}| > |\lambda_n|$$

and the power method is applied to A^{-1}, q_k will converge to the eigenvector x_n of A corresponding to λ_n. This is called **inverse iteration** and was introduced in 1944 by Wielandt [385].

The power method can also be applied to the matrix $(A - sI)^{-1}$, where s is a chosen **shift** of the spectrum. The corresponding iteration can be written

$$(A - sI)\hat{q}_k = q_{k-1}, \quad q_k = \hat{q}_k/\|\hat{q}_k\|_2, \qquad k = 1, 2, \ldots. \tag{9.3.5}$$

The eigenvalues of $(A - sI)^{-1}$ equal

$$\mu_j = (\lambda_j - s)^{-1}. \tag{9.3.6}$$

Note that there is no need to explicitly invert $A - sI$. Instead we compute a triangular factorization of $A - sI$, and in each step of (9.3.5) solve two triangular systems

$$L(U\hat{q}_k) = Pq_{k-1}, \quad P(A - sI) = LU.$$

For a dense matrix A one step of the iteration (9.3.6) is therefore no more costly than one step of the simple power method. However, if the matrix is sparse the total number of nonzero elements in L and U may be much larger than in A. Note that if A is positive definite (or diagonally dominant) this property is in general not shared by the shifted matrix $(A - sI)$. Hence, in general partial pivoting must be employed.

If s is chosen sufficiently close to an eigenvalue λ_i, so that $|\lambda_i - s| \ll |\lambda_j - s|$, $\lambda_i \neq \lambda_j$ then $(\lambda_i - s)^{-1}$ is a dominating eigenvalue of B,

$$|\lambda_i - s|^{-1} \gg |\lambda_j - s|^{-1}, \quad \lambda_i \neq \lambda_j. \tag{9.3.7}$$

Then q_k will converge fast to the eigenvector x_i, and an approximation $\bar{\lambda}_i$ to the eigenvalue λ_i of A is obtained from the Rayleigh quotient

$$\frac{1}{\lambda_i - s} \approx q_{k-1}^T (A - sI)^{-1} q_{k-1} = q_{k-1}^T \hat{q}_k,$$

where $\hat{q_k}$ satisfies $(A - sI)\hat{q_k} = q_{k-1}$. Thus,

$$\bar{\lambda}_i = s + 1/(q_{k-1}^T \hat{q}_k). \tag{9.3.8}$$

It is possible to choose a different shift in each iteration step, at the cost of repeating the LU factorization. This more general iteration is closely related to Newton's method applied to the extended nonlinear system $(A - \lambda x) = 0$, where we require that the eigenvector is normalized so that $e_m^T x = 1$, where e_m is a unit vector, i.e. the mth component of x is normalized to one. This corresponds to a system of $(n+1)$ equations

$$(A - \lambda I)x = 0, 1 - e_m^T x = 0,$$

for the $n+1$ unknowns x, λ; see [308].

An a posteriori bound for the error in the approximate eigenvalue $\bar{\lambda}_i$ of A can be obtained from the residual corresponding to $(\bar{\lambda}_i, \hat{q}_k)$, which equals

$$r_k = A\hat{q}_k - \Big(s + 1/(q_{k-1}^T \hat{q}_k)\Big)\hat{q}_k = q_{k-1} - \hat{q}_k/(q_{k-1}^T \hat{q}_k).$$

Then, by Theorem 9.2.14, $(\bar{\lambda}_i, \hat{q}_k)$ is an exact eigenpair of a matrix $A + E$, where $\|E\|_2 \le \|r_k\|_2/\|\hat{q}_k\|_2$. If A is real symmetric then the error in the approximative eigenvalue $\hat{\lambda}_i$ of A is bounded by $\|r_k\|_2/\|\hat{q}_k\|_2$.

9.3.4 Eigenvectors by Inverse Iteration

After extensive developments by Wilkinson and others inverse iteration has become the method of choice for computing the associated eigenvector to an eigenvalue λ_i, for which an accurate approximation already is known. Often just *one step* of inverse iteration suffices.

Inverse iteration will in general converge faster the closer μ is to λ_i. However, if μ equals λ_i up to machine precision then $A - \mu I$ in (9.3.5) is numerically singular. It was long believed that inverse iteration was doomed to failure when μ was chosen too close to an eigenvalue. Fortunately this is not the case!

If Gaussian elimination with partial pivoting is used the computed factorization of $(A - \mu I)$ will satisfy

$$P(A + E - \mu I) = \bar{L}\bar{U},$$

where $\|E\|_2/\|A\|_2 = f(n)O(u)$, and u is the unit roundoff and $f(n)$ a modest function of n (see Theorem 6.6.5). Since the rounding errors in the solution of the triangular systems usually are negligible the computed q_k will nearly satisfy

$$(A + E - \mu I)\hat{q}_k = q_{k-1}.$$

This shows that the inverse power method will give an approximation to an eigenvector of a slightly perturbed matrix $A + E$, independent of the ill-conditioning of $(A - \mu I)$.

To decide when a computed vector is a numerically acceptable eigenvector corresponding to an approximate eigenvalue we can apply the simple a posteriori error bound in Theorem 9.2.14 to inverse iteration. By (9.3.5) q_{k-1} is the residual vector corresponding to the approximate eigenpair $(\mu, \hat{q}_k)$. Hence, where u is the unit roundoff, $\hat{q}_k$ is a numerically acceptable eigenvector if

$$\|q_{k-1}\|_2/\|\hat{q}_k\|_2 \le u\|A\|_2. \tag{9.3.9}$$

Example 9.3.2.

The matrix $A = \begin{pmatrix} 1 & 1 \\ 0.1 & 1.1 \end{pmatrix}$ has a simple eigenvalue $\lambda_1 = 0.7298438$ and the corresponding normalized eigenvector is $x_1 = (0.9653911, -0.2608064)^T$. We take $\mu = 0.7298$ to be an approximation to λ_1, and perform one step of inverse iteration, starting with $q_0 = (1, 0)^T$ we get

$$A - \mu I = LU = \begin{pmatrix} 1 & 0 \\ 0.37009623 & 1 \end{pmatrix} \begin{pmatrix} 0.2702 & 1 \\ 0 & 0.0001038 \end{pmatrix}$$

and $\hat{q}_1 = 10^4(1.3202568, -0.3566334)^T$, $q_1 = (0.9653989, -0.2607777)^T$, which agrees with the correct eigenvector to more than four decimals. From the backward error bound it follows that 0.7298 and q_1 is an exact eigenpair to a matrix $A + E$, where $\|E\|_2 \le 1/\|\hat{q}_1\|_2 = 0.73122 \cdot 10^{-4}$.

Inverse iteration is a useful algorithm for calculation of specified eigenvectors corresponding to well separated eigenvalues for dense matrices. In order to save work in the triangular factorizations the matrix is usually first reduced to Hessenberg or real tridiagonal form, by the methods described in Sec. 9.5.

It is quite tricky to develop inverse iteration into a reliable algorithm for the case when the eigenvalues are not well separated. When A is symmetric and eigenvectors corresponding to multiple or very close eigenvalues are required, special steps have to be taken to ensure orthogonality of the eigenvectors. In the nonsymmetric case the situation can be worse in particular if the eigenvalue is defective or very ill-conditioned. Then, unless a suitable initial vector is used inverse iteration may not produce a numerically acceptable eigenvector. Often a random vector with elements from a uniform distribution in $[-1, 1]$ will work.

Example 9.3.3.

The matrix

$$A = \begin{pmatrix} 1+\epsilon & 1 \\ \epsilon & 1+\epsilon \end{pmatrix}$$

has eigenvalues $\lambda = (1+\epsilon) \pm \sqrt{\epsilon}$. Assume that $|\epsilon| \approx u$, where u is the machine precision. Then the eigenpair $\lambda = 1$, $x = (1, 0)^T$ is a numerically acceptable eigenpair of A, since it is exact for the matrix $A + E$, where

$$E = -\begin{pmatrix} \epsilon & 0 \\ \epsilon & \epsilon \end{pmatrix}, \qquad \|E\|_2 < \sqrt{3}u.$$

If we perform one step of inverse iteration starting from the acceptable eigenvector $q_0 = (1, 0)^T$ then we get

$$\hat{q}_1 = \frac{1}{1-\epsilon} \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

No growth occurred and it can be shown that $(1, q_1)$ *is not* an acceptable eigenpair of A. If we carry out one more step of inverse iteration we will again get an acceptable eigenvector!

Equation (9.2.24) gives an expression for the backward error E of the computed eigenpair. An error bound can then be obtained by applying the perturbation analysis of Sec. 9.2. In the Hermitian case the eigenvalues are perfectly conditioned, and the error bound equals $\|E\|_2$. In general the sensitivity of an eigenvalue λ is determined by $1/s(\lambda) = 1/|y^H x|$, where x and y are right and left unit eigenvector corresponding to λ; see Sec. 9.2.2. If the power method is applied also to A^H (or in inverse iteration to $(A^H - \mu I)^{-1}$) we can generate an approximation to y and hence estimate $s(\lambda)$.

9.3.5 Rayleigh Quotient Iteration

A natural variation of the inverse power method is to vary the shift μ in each iteration. The previous analysis suggests choosing a shift equal to the Rayleigh quotient of the current eigenvector approximation. This leads to the **Rayleigh Quotient Iteration (RQI)**:

Let q_0, $\|q_0\|_2 = 1$, be a given starting vector, and for $k = 1, 2, \dots$, compute

$$\big(A - \rho(q_{k-1})I\big)\hat{q}_k = q_{k-1}, \quad \rho(q_{k-1}) = q_{k-1}^T A q_{k-1}, \tag{9.3.10}$$

and set $q_k = \hat{q}_k/\|\hat{q}_k\|_2$. Here $\rho(q_{k-1})$ is the Rayleigh quotient of q_{k-1}.

RQI can be used to improve a given approximate eigenvector. It can also be used to find an eigenvector of A starting from any unit vector q_0, but then we cannot say to which eigenvector $\{q_k\}$ will converge. There is also a possibility that some unfortunate choice of starting vector will lead to endless cycling. However, it can be shown that such cycles are unstable under perturbations so this will not occur in practice.

In the RQI a new triangular factorization must be computed of the matrix $A - \rho(q_{k-1})I$ for each iteration step, which makes this algorithm much more expensive than ordinary inverse iteration. However, if the matrix A is, for example, of Hessenberg (or tridiagonal) form the extra cost is small. If the RQI converges towards an eigenvector corresponding to a *simple* eigenvalue then it can be shown that convergence is quadratic. More precisely, it can be shown that

$$\eta_k \le c_k \eta_{k-1}^2, \quad \eta_k = \|Aq_k - \rho(q_k)q_k\|_2,$$

where c_k changes only slowly; see Stewart [340, 1973, Sec. 7.2].

If the matrix A is real and symmetric (or Hermitian), then the situation is even more satisfactory because of the result in Theorem 9.2.16. This theorem says that if an eigenvector is known to precision ϵ, the Rayleigh quotient approximates the corresponding eigenvalue to precision ϵ^2. This leads to *cubic convergence* for the RQI for real symmetric (or Hermitian) matrices. Also, in this case it is no longer necessary to assume that the iteration converges to an eigenvector corresponding to a simple eigenvalue. Indeed, it can be shown that the for Hermitian matrices RQI has *global convergence*, i.e., it converges from *any starting vector* q_0. A key fact in the proof is that *the norm of the residuals always decrease*, $\eta_{k+1} \le \eta_k$, for all k; see Parlett [305, Sec. 4.8].

9.3.6 Subspace Iteration

A natural generalization of the power method is to iterate *simultaneously* with several vectors. Let $Z_0 = S = (s_1, \dots, s_p) \in \mathbf{R}^{n\times p}$, be an initial matrix of rank $p > 1$. If we compute a sequence of matrices $\{Z_k\}$, from

$$Z_k = AZ_{k-1}, \qquad k = 1, 2, \dots, \tag{9.3.11}$$

then it holds

$$Z_k = A^k S = (A^k s_1, \dots, A^k s_p). \tag{9.3.12}$$

In applications A is often a very large sparse matrix and $p \ll n$.

At first it is not clear that we gain much by iterating with several vectors. If A has a dominant eigenvalue λ_1 *all the columns* of Z_k will converge to a scalar multiple of the dominant eigenvector x_1. Hence, Z_k will be close to a matrix of numerical rank one.

We first note that we are really computing a sequence of subspaces. If $\mathcal{S} = \operatorname{span}(S)$ the iteration produces the subspaces $A^k\mathcal{S} = \operatorname{span}(A^k S)$. Hence, the problem is that the basis $A^k s_1, \dots, A^k s_p$ of this subspace becomes more and more ill-conditioned. This can be avoided by be maintaining orthogonality between the columns as follows: Starting with a matrix Q_0 with orthogonal columns we compute

$$Z_k = AQ_{k-1} = Q_k R_k, \qquad k = 1, 2, \dots, \tag{9.3.13}$$

where $Q_k R_k$ is the QR decomposition of Z_k. Here Q_k can be computed, e.g., by Gram-Schmidt orthogonalization of Z_k. The iteration (9.3.13) is also called **orthogonal iteration**. Note that R_k plays the rule of a normalizing matrix. We have $Q_1 = Z_1 R_1^{-1} = AQ_0 R_1^{-1}$. Similarly, it can be shown by induction that

$$Q_k = A^k Q_0 (R_k \cdots R_1)^{-1}. \tag{9.3.14}$$

It is important to note that if $Z_0 = Q_0$, then both iterations (9.3.11) and (9.3.13) will generate the same sequence of subspaces. $\mathcal{R}(A^k Q_0) = \mathcal{R}(Q_k)$. However, in orthogonal iteration an orthogonal bases for the subspace is calculated at each iteration. (Since the iteration (9.3.11) is less costly it is sometimes preferable to perform the orthogonalization in (9.3.13) only occasionally when needed.)

The method of orthogonal iteration overcomes several of the disadvantages of the power method. In particular, it allows us to determine a dominant invariant subspace of a multiple eigenvalue.

Assume that the eigenvalues of A satisfy

$$|\lambda_1| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n| \tag{9.3.15}$$

and let

$$\begin{pmatrix} U_1^H \\ U_2^H \end{pmatrix} A(U_1 \; U_2) = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}, \tag{9.3.16}$$

be a Schur decomposition of A, where

$$\operatorname{diag}(T_{11}) = (\lambda_1, \cdots, \lambda_p)^H.$$

Then the subspace $\mathcal{U}_1 = \mathcal{R}(U_1)$ is a **dominant** invariant subspace of A. It can be shown that almost always the subspaces $\mathcal{R}(Q_k)$ in orthogonal iteration (9.3.13) converge to $\mathcal{U}_1$ when $k \to \infty$.

Theorem 9.3.1.

Let $\mathcal{U}_1 = \mathcal{R}(U_1)$ be a dominant invariant subspace of A defined in (9.3.16). *Let $\mathcal{S}$ be a p-dimensional subspace of $\mathbf{C}^n$ such that $\mathcal{S} \cap \mathcal{U}_1^{\perp} = \{0\}$. Then there exists a constant C such that*

$$\theta_{\max}(A^k\mathcal{S}, \mathcal{U}_1) \le C|\lambda_{p+1}/\lambda_p|^k.$$

where $\theta_{\max}(\mathcal{X}, \mathcal{Y})$ denotes the largest angle between the two subspaces (see Definition 9.2.6).

Proof. See Golub and Van Loan [184, pp. 333]. $\square$

If we perform subspace iteration on p vectors, we are simultaneously performing subspace iteration on a nested sequence of subspaces

$$\operatorname{span}(s_1), \quad \operatorname{span}(s_1, s_2), \ldots, \quad \operatorname{span}(s_1, s_2, \ldots, s_p).$$

This is also true for orthogonal iteration since this property is not changed by the orthogonalization procedure. Hence, Theorem 9.3.1 shows that whenever $|\lambda_{q+1}/\lambda_q|$ is small for some $q \le p$, the convergence to the corresponding dominant invariant subspace of dimension q will be fast.

We now show that there is a duality between direct and inverse subspace iteration.

Lemma 9.3.2. (Watkins [1982])

Let $\mathcal{S}$ and $\mathcal{S}^{\perp}$ be orthogonal complementary subspaces of $\mathbf{C}^n$. Then for all integers k the spaces $A^k\mathcal{S}$ and $(A^H)^{-k}\mathcal{S}^{\perp}$ are also orthogonal.

Proof. Let $x \perp y \in \mathbf{C}^n$. Then $(A^k x)^H (A^H)^{-k} y = x^H y = 0$ and thus $A^k x \perp (A^H)^{-k} y$. $\square$

This duality property means that the two sequences

$$S, AS, A^2S, \ldots, \qquad S^{\perp}, (A^H)^{-1}S^{\perp}, (A^H)^{-2}S^{\perp}, \ldots$$

are equivalent in that they yield orthogonal complements! This result will be important in Section 9.4.1 for the understanding of the QR algorithm.

Approximations to eigenvalues of A can be obtained from eigenvalues of the sequence of matrices

$$B_k = Q_k^T A Q_k = Q_k^T Z_{k+1} \in \mathbf{R}^{p \times p}. \tag{9.3.17}$$

Note that B_k is a generalized Rayleigh quotient, see Sec. 9.7– 9.7.1. Finally, both direct and inverse orthogonal iteration can be performed using a sequence of shifted matrices $A - \mu_k I$, $k = 0, 1, 2, \ldots$.

Review Questions

3.1 Describe the power method and its variants. Name at least one important application of the shifted inverse power method.

3.2 If the Rayleigh Quotient Iteration converges to a simple eigenvalue of a general matrix A, what is the asymptotic rate of convergence? If A is Hermitian, what can you say then?

3.3 Describe how the power method can be generalized to simultaneously iterating with several starting vector.

Problems

3.1 Let $A \in \mathbf{R}^{n\times n}$ be a symmetric matrix with eigenvalues satisfying $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_{n-1} > \lambda_n$. Show that the choice $\mu = (\lambda_2 + \lambda_n)/2$ gives fastest convergence towards the eigenvector corresponding to λ_1 in the power method applied to $A - \mu I$. What is this rate of convergence?

3.2 The matrix A has one real eigenvalue $\lambda = \lambda_1$ and another $\lambda = -\lambda_1$. All remaining eigenvalues satisfy $|\lambda| < |\lambda_1|$. Generalize the simple power method so that it can be used for this case.

3.3 (a) Compute the residual vector corresponding to the last eigenpair obtained in Example 9.3.1, and give the corresponding backward error estimate.

(b) Perform Aitken extrapolation on the Rayleigh quotient approximations in Example 9.3.1 to compute an improved estimate of λ_1.

3.4 The symmetric matrix

$$A = \begin{pmatrix} 14 & 7 & 6 & 9 \\ 7 & 9 & 4 & 6 \\ 6 & 4 & 9 & 7 \\ 9 & 6 & 7 & 15 \end{pmatrix}$$

has an eigenvalue $\lambda \approx 4$. Compute an improved estimate of λ with one step of inverse iteration using the factorization $A - 4I = LDL^T$.

3.5 For a symmetric matrix $A \in \mathbf{R}^{n\times n}$ it holds that $\sigma_i = |\lambda_i|$, $i = 1 : n$. Compute with inverse iteration using the starting vector $x = (1, -2,\ 1)^T$ the smallest singular value of the matrix

$$A = \begin{pmatrix} 1/5 & 1/6 & 1/7 \\ 1/6 & 1/7 & 1/8 \\ 1/7 & 1/8 & 1/9 \end{pmatrix}$$

with at least two significant digits.

3.6 The matrix

$$A = \begin{pmatrix} 1 & 1 \\ \epsilon & 1+\epsilon \end{pmatrix}$$

has two simple eigenvalues close to 1 if $\epsilon > 0$. For $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$ first compute the smallest eigenvalue to six decimals, and then perform inverse iteration to determine the corresponding eigenvectors. Try as starting vectors both $x = (1, 0)^T$ and $x = (0, 1)^T$.

9.4 The QR Algorithm

9.4.1 The Basic QR Algorithm

Suppose that the LU factorization of $A = LU$ is computed and then the factors are multiplied in reverse order. This performs the similarity transformation

$$L^{-1}AL = L^{-1}(LU)L = UL,$$

cf. Lemma 9.1.5. In the LR algorithm[36] this process is iterated. Setting $A_1 = A$, and compute

$$A_k = L_kU_k, \qquad A_{k+1} = U_kL_k, \quad k = 1, 2, \dots \tag{9.4.1}$$

The **LR algorithm** is due to H. Rutishauser [325, 1958] and is related to a more general algorithm, the qd algorithm. This can be used to find poles of rational functions or zeros of polynomials; see Volume I, Sec. 3.5.5.

Repeated application of (9.4.1) gives

$$A_k = L_{k-1}^{-1} \cdots L_2^{-1}L_1^{-1}A_1L_1L_2 \cdots L_{k-1}. \tag{9.4.2}$$

or

$$L_1L_2 \cdots L_{k-1}A_k = A_1L_1L_2 \cdots L_{k-1}. \tag{9.4.3}$$

The two matrices defined by

$$T_k = L_1 \cdots L_{k-1}L_k, \qquad U_k = U_kU_{k-1} \cdots U_1, \tag{9.4.4}$$

are lower and upper triangular respectively. Forming the product T_kU_k and using (9.4.3) we have

$$\begin{aligned} T_kU_k &= L_1 \cdots L_{k-1}(L_kU_k)U_{k-1} \cdots U_1 \\ &= L_1 \cdots L_{k-1}A_kU_{k-1} \cdots U_1 \\ &= A_1L_1 \cdots L_{k-1}U_{k-1} \cdots U_1. \end{aligned}$$

Repeating this we obtain the basic relation

$$T_kU_k = A^k. \tag{9.4.5}$$

which shows the close relationship between the LR algorithm and the power method.

Under certain restrictions it can be shown that the matrix A_k converges to an upper triangular matrix U_∞. The eigenvalues of A then lie on the diagonal of U_∞. To establish this result several assumptions need to be made. It has to be assumed

[36] In German the LU factorization is called the LR factorization, where L and R stands for "links" and "recht".

that the LU factorization exists at every stage. This is not the case for the simple matrix

$$A = \begin{pmatrix} 0 & 1 \\ -3 & 4 \end{pmatrix},$$

with eigenvalues 1 and 3. We could equally well work with the shifted matrix $A+I$, for which the LR algorithm converges. However, there are other problems with the LR algorithm, which make a robust implementation difficult.

To avoid the problems with the LR algorithm it seems natural to devise a similar algorithm using *orthogonal* similarity transformations. This leads to the QR algorithm, published independently by Francis [140, 1961] and Kublanovskaya [247, 1961]. However, Francis paper also contained algorithmic developments needed for the practical implementation. The QR algorithm is still the most important algorithm for computing eigenvalues and eigenvectors of matrices.[37] For symmetric (Hermitian) matrices alternative algorithms have been developed that can compete with the QR algorithm in terms of speed and accuracy; see Sec. 9.5.

In the QR algorithm a sequence of matrices $A_{k+1} = Q_k^T A_k Q_k$ similar to to $A_1 = A$ are computed by

$$A_k = Q_k R_k, \qquad A_{k+1} = R_k Q_k, \quad k = 1, 2, \ldots, \tag{9.4.6}$$

where Q_k is orthogonal and R_k is upper triangular. That is, in the kth step the QR decomposition of A_k is computed and then the factors are multiplied in reverse order giving A_{k+1}.

The successive iterates of the QR algorithm satisfy relations similar to those derived for the LR algorithm. If we define

$$P_k = Q_1 Q_2 \cdots Q_k, \qquad U_k = R_k \cdots R_2 R_1,$$

where P_k is orthogonal and U_k is upper triangular, then by repeated applications of (9.4.6) it follows that

$$A_{k+1} = P_k^T A P_k. \tag{9.4.7}$$

Further, we have

$$\begin{aligned} P_k U_k &= Q_1 \cdots Q_{k-1} (Q_k R_k) R_{k-1} \cdots R_1 &(9.4.8)\\ &= Q_1 \cdots Q_{k-1} A_k R_{k-1} \cdots R_1 &(9.4.9)\\ &= A_1 Q_1 \cdots Q_{k-1} R_{k-1} \cdots R_1. &(9.4.10)\end{aligned}$$

Repeating this gives

$$P_k U_k = A^k. \tag{9.4.11}$$

We now show that in general the QR iteration is related to orthogonal iteration. Given an orthogonal matrix $\tilde{Q}_0 \in \mathbf{R}^{n\times n}$, orthogonal iteration computes a sequence $\tilde{Q}_1$, $\tilde{Q}_2, \ldots$, where

$$Z_k = A\tilde{Q}_k, \qquad Z_k = \tilde{Q}_{k+1} R_k. \qquad k = 0, 1, \ldots \tag{9.4.12}$$

[37]The QR algorithm was chosen as one of the 10 algorithms with most influence on scientific computing in the 20th century by the editors of the journal Computing in Science and Engineering.

The related sequence of matrices $B_k = \tilde{Q}_k^T A \tilde{Q}_k = \tilde{Q}_k^T Z_k$ similar to A can be computed directly. Using (9.4.12) we have $B_k = (\tilde{Q}_k^T \tilde{Q}_{k+1})R_k$, which is the QR decomposition of B_k, and

$$B_{k+1} = (\tilde{Q}_{k+1}^T A)\tilde{Q}_{k+1} = (\tilde{Q}_{k+1}^T A\tilde{Q}_k)\tilde{Q}_k^T \tilde{Q}_{k+1} = R_k(\tilde{Q}_k^T \tilde{Q}_{k+1}).$$

Hence, B_{k+1} is obtained by multiplying the QR factors of B_k in reverse order, which is just one step of QR iteration! If, in particular we take $\tilde{Q}_0 = I$ then $B_0 = A_0$, and it follows that $B_k = A_k$, $k = 0, 1, 2, \ldots$, where A_k is generated by the QR iteration (9.4.6). From the definition of B_k and (9.4.6) we have $\tilde{Q}_k = P_{k-1}$, and (compare (9.3.4))

$$A^k = \tilde{Q}_k \tilde{R}_k, \quad \tilde{R}_k = R_k \cdots R_2 R_1. \tag{9.4.13}$$

From this we can conclude that the first p columns of $\tilde{Q}_k$ form an orthogonal basis for the space spanned by the first p columns of A^k, i.e., $A^k(e_1, \ldots, e_p)$.

In the QR algorithm subspace iteration takes place on the subspaces spanned by the unit vectors $(e_1, \ldots, e_p)$, $p = 1 : n$. It is important for the understanding of the QR algorithm to recall that therefore, according to Theorem 9.3.1, also inverse iteration by $(A^H)^{-1}$ takes place on the orthogonal complements, i.e., the subspaces spanned by $(e_{p+1}, \ldots, e_n)$, $p = 0 : n-1$. Note that this means that in the QR algorithm direct iteration is taking place in the top left corner of A, and inverse iteration in the lower right corner. (For the QL algorithm this is reversed, see below.)

Assume that the eigenvalues of A satisfy $|\lambda_p| > |\lambda_{p+1}|$, and let (9.3.16) be a corresponding Schur decomposition. Let $P_k = (P_{k1},\ P_{k2})$, $P_{k1} \in \mathbf{R}^{n\times p}$, be defined by (9.4.6). Then by Theorem 9.3.1 with linear rate of convergence equal to $|\lambda_{p+1}/\lambda_p|$

$$\mathcal{R}(P_{k1}) \to \mathcal{R}(U_1).$$

where U_1 spans the dominant invariant subspace of dimension p of A. It follows that A_k will tend to reducible form

$$A_k = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} + O\left((|\lambda_{p+1}/\lambda_p|)^k \right).$$

This result can be used to show that under rather general conditions A_k will tend to an upper triangular matrix R whose diagonal elements then are the eigenvalues of A.

Theorem 9.4.1.

If the eigenvalues of A satisfy $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, then the matrices A_k generated by the QR algorithm will tend to upper triangular form. The lower triangular elements $a_{ij}^{(k)}$, $i > j$, converge to zero with linear rate equal to $|\lambda_i/\lambda_j|$.

Proof. A proof can be based on the convergence properties of orthogonal iteration; see Watkins [379]. □

If the product P_k, $k = 1, 2, \ldots$, of the transformations are accumulated the eigenvectors may then be found by calculating the eigenvectors of the final triangular matrix and then transforming them back.

In practice, to speed up convergence, a shifted version of the QR algorithm is used, where

$$A_k - \tau_k I = Q_k R_k, \qquad R_k Q_k + \tau_k I = A_{k+1}, \qquad k = 0, 1, 2, \dots. \tag{9.4.14}$$

and τ_k is a **shift**. It is easily verified that since the shift is restored at the end of the step it holds that $A_{k+1} = Q_k^T A_k Q_k$,

If τ_k approximates a simple eigenvalue λ_j of A, then in general $|\lambda_i - \tau_k| \gg |\lambda_j - \tau_k|$ for $i \neq j$. By the result above the off-diagonal elements in the *last* row of $\tilde{A}_k$ will approach zero very fast.The relationship between the shifted QR algorithm and the power method is expressed in the next theorem.

Theorem 9.4.2.

Let Q_j and R_j, $j = 0 : k$, be computed by the QR algorithm (9.4.14)*. Then it holds that*

$$(A - \tau_k I) \cdots (A - \tau_1 I)(A - \tau_0 I) = P_k U_k, \tag{9.4.15}$$

where

$$P_k = Q_0 Q_1 \cdots Q_k, \qquad U_k = R_k R_{k-1} \cdots R_0. \tag{9.4.16}$$

Proof. For $k = 0$ the relation (9.4.15) is just the definition of Q_0 and R_0. Assume now that the relation is true for $k-1$. From $A_{k+1} = Q_k^T A_k Q_k$ and using the orthogonality of P_k

$$A_{k+1} - \tau_k I = P_k^T (A - \tau_k I) P_k. \tag{9.4.17}$$

Hence, $R_k = (A_{k+1} - \tau_k I) Q_k^T = P_k^T (A - \tau_k I) P_k Q_k^T = P_k^T (A - \tau_k I) P_{k-1}$. Post-Multiplying this equation by U_{k-1} we get

$$R_k U_{k-1} = U_k = P_k^T (A - \tau_k I) P_{k-1} U_{k-1},$$

and thus $P_k U_k = (A - \tau_k I) P_{k-1} U_{k-1}$. Using the inductive hypothesis the theorem follows. □

A variant called the QL algorithm is based on the iteration

$$A_k = Q_k L_k, \qquad L_k Q_k = A_{k+1}, \qquad k = 0, 1, 2, \dots, \tag{9.4.18}$$

where L_k is *lower* triangular, and is merely a reorganization of the QR algorithm. Let J be a permutation matrix such that JA reverses the rows of A. Then AJ reverses the columns of A and hence JAJ reverses both rows and columns. If R is upper triangular then JRJ is lower triangular. It follows that if $A = QR$ is the QR decomposition then $JAJ = (JQJ)(JRJ)$ is the QL decomposition of JAJ. It follows that the QR algorithm applied to A is the same as the QL algorithm applied to JAJ. The convergence theory is therefore the same for both algorithms. However, in the QL algorithm inverse iteration is taking place in the top left corner of A, and direct iteration in the lower right corner.

An important case where the choice of either the OR or QL algorithm should be preferred is when the matrix A is *graded*, see Sec. 9.5.1. If the large elements occur in the lower right corner then the QL algorithm is more stable. (Note that then the reduction to tridiagonal form should be done from bottom up; see the remark in Sec. 9.5.1.) Of course, the same effect can be achieved by explicitly reversing the ordering of the rows and columns.

9.4.2 Reduction to Hessenberg Form

For a dense matrix the cost for one QR iteration is $4n^3/3$ flops, which is too much to make it a practical algorithm. The work in the QR algorithm is greatly reduced if the matrix $A \in \mathbf{R}^{n\times n}$ is first reduced to Hessenberg form by an orthogonal similarity transformation

$$Q^T AQ = H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,n-1} & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2,n-1} & h_{2n} \\ & h_{32} & \ddots & \vdots & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & h_{n,n-1} & h_{nn} \end{pmatrix}. \tag{9.4.19}$$

The Hessenberg form is preserved by the QR iteration as we now show. Let H_k be upper Hessenberg and set

$$H_k - \tau_k I = Q_k R_k, \qquad R_k Q_k + \tau_k I = H_{k+1}, \quad k = 0, 1, 2, \dots. \tag{9.4.20}$$

First note that the addition or subtraction of $\tau_k I$ does not affect the Hessenberg form. If R_k is nonsingular then $Q_k = (H_k - \tau_k I)R_k^{-1}$ is a product of an upper Hessenberg matrix and an upper triangular matrix, and therefore again a Hessenberg matrix (cf. Problem 7.4.1). Hence, $R_k Q_k$ and H_{k+1} are again of upper Hessenberg form. The cost of the QR step (9.4.20) is reduced to only $4n^2$ flops.

We set $A = A^{(1)}$ and compute $A^{(2)}, \dots, A^{(n-1)} = H$, where

$$A^{(k+1)} = P_k A^{(k)} P_k, \quad k = 1 : n-2.$$

Here P_k is a Householder reflection,

$$P_k = I - \frac{1}{\gamma_k} u_k u_k^T, \qquad \gamma_k = \frac{1}{2}\|u_k\|_2^2 \tag{9.4.21}$$

which is chosen to zero the elements in column k below the first subdiagonal. The first k elements in u_k are zero. In the first step

$$A^{(2)} = P_1 A P_1 = \begin{pmatrix} h_{11} & h_{12} & \tilde a_{13} & \dots & \tilde a_{1n} \\ h_{21} & h_{22} & \tilde a_{23} & \dots & \tilde a_{2n} \\ 0 & \tilde a_{32} & \tilde a_{33} & \dots & \tilde a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \tilde a_{n2} & \tilde a_{n3} & \dots & \tilde a_{nn} \end{pmatrix},$$

where P_1 is chosen so that $P_1 A$ has zeros in the first column in the positions shown above. These zeros are not destroyed by the post-multiplication $(P_1 A)P_1$, which only affects the $n-1$ last columns. All later steps are similar.

We note that each of the Householder matrices P_j, $j = 1 : n$, satisfy $P_j e_1 = e_1$, and therefore we have $Qe_1 = e_1$. However, it is easy to modify the algorithm so that the first column of Q is proportional to any given nonzero vector z. Let P_0 be a Householder reflector such that $P_0 e_1 = \beta z$, $\beta = 1/\|z\|_2$. Then, taking $Q = P_0 P_1 \cdots P_{n-2}$ we have $Qe_1 = P_0 e_1 \beta z$.

Note that P_k is completely specified by u_k and γ_k, and that the required products of the form $P_k A$ and AP_k, can be computed in $4(n-k)^2$ flops by rank one update

$$P_k A = A - u_k (A^T u_k)^T/\gamma_k, \qquad AP_k = A - (Au_k)u_k^T/\gamma_k.$$

A simple operation count shows that this reduction requires

$$4\sum_{k=1}^{n}(k^2 + nk) = 10n^3/3 \text{ flops}.$$

Assembling the matrix $Q = Q_0 Q_1 \cdots Q_{n-2}$ adds another $4n^3/3$ flops. As described the reduction to Hessenberg form involves level 2 operations. Dongarra, Hammarling and Sorensen [111] have shown how to speed up the reduction by introduce level 3 operations.

The reduction by Householder transformations is stable in the sense that the computed $\bar{H}$ can be shown to be the *exact result* of an orthogonal similarity transformation of a matrix $A + E$, where

$$\|E\|_F \leq cn^2 u \|A\|_F, \tag{9.4.22}$$

and c is a constant of order unity. Moreover if we use the information stored to generate the product $U = P_1 P_2 \cdots P_{n-2}$ then the computed result is close to the matrix U that reduces $A + E$. This will guarantee that the eigenvalues and transformed eigenvectors of $\bar{H}$ are accurate approximations to those of a matrix close to A.

However, it should be noted that *this does not imply that the computed $\bar{H}$ will be close to the matrix H corresponding to the exact reduction of A.* Even the same algorithm run on two computers with different floating point arithmetic may produce very different matrices $\bar{H}$. Behavior of this kind, named **irrelevant instability** by B. N. Parlett, unfortunately continue to cause much unnecessary concern! The backward stability of the reduction ensures that each matrix will be similar to A to working precision and will yield approximate eigenvalues to as much absolute accuracy as is warranted.

Definition 9.4.3.

An upper Hessenberg matrix is called **unreduced** *if all its subdiagonal elements are nonzero.*

If $H \in \mathbf{R}^{n\times n}$ is an unreduced Hessenberg matrix, then $\operatorname{rank}(H) \geq n - 1$, and that therefore if H has a multiple eigenvalue it must be defective. In the following we assume that H is unreduced. This is no restriction because if H has a zero subdiagonal entry, then it can be partitioned into the block-diagonal form

$$H = \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix}.$$

Then the eigenvalue problem for H **decouples** into two eigenproblems for the Hessenberg matrices H_{11} and H_{22}. If these are not unreduced, then the eigenvalue problem can be split into even smaller pieces.

The following important theorem states that for an unreduced Hessenberg matrix the decomposition (9.4.19) is uniquely determined by the first column in Q, i.e., $q_1 = Qe_1$,

Theorem 9.4.4 (*Implicit Q Theorem*).

Given $A, H, Q \in \mathbf{R}^{n\times n}$, where $Q = (q_1, \dots, q_n)$ is orthogonal and $H = Q^T AQ$ is upper Hessenberg with positive subdiagonal elements. Then H and Q are uniquely determined by the first column q_1 in Q.

Proof. Assume we have already computed $q_1, \dots, q_k$ and the first $k-1$ columns in H. (Since q_1 is known this assumption is valid for $k = 1$.) Equating the kth columns in

$$QH = (q_1, q_2, \dots, q_n)H = A(q_1, q_2, \dots, q_n) = AQ$$

we obtain the equation

$$h_{1,k}q_1 + \cdots + h_{k,k}q_k + h_{k+1,k}q_{k+1} = Aq_k, \quad k = 1 : n-1.$$

Multiplying this by q_i^T and using the orthogonality of Q, we obtain

$$h_{ik} = q_i^T Aq_k, \quad i = 1 : k.$$

Since H is unreduced $h_{k+1,k} \neq 0$, and therefore q_{k+1} and $h_{k+1,k}$ are determined (up to a trivial factor of ± 1) by

$$q_{k+1} = h_{k+1,k}^{-1}\Big(Aq_k - \sum_{i=1}^{k} h_{ik}q_i\Big),$$

and the condition that $\|q_{k+1}\|_2 = 1$. □

The proof of the above theorem is constructive and gives an alternative algorithm for generating Q and H known as Arnoldi's process; see Sec. 9.7.5. This algorithm has the property that only matrix–vector products Aq_k are required, which makes the process attractive when A is large and sparse. The drawback is that roundoff errors will cause a loss of orthogonality in the generated vectors $q_1, q_2, q_3, \dots$, which has to be taken into account.

9.4.3 The Hessenberg QR Algorithm

In the **explicit-shift** QR algorithm we first form the matrix $H_k - \tau_k I$, and then apply a sequence of Givens rotations, $G_{j,j+1}$, $j = 1 : n-1$ (see (7.4.14)) so that

$$G_{n-1,n} \cdots G_{23} G_{12} (H_k - \tau_k I) = Q_k^T (H_k - \tau_k I) = R_k,$$

becomes upper triangular. At a typical step ($n = 6$, $j = 3$) the partially reduced matrix has the form

$$\begin{pmatrix} \rho_{11} & \times & \times & \times & \times & \times \\ & \rho_{22} & \times & \times & \times & \times \\ & & \nu_{33} & \times & \times & \times \\ & & h_{43} & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

The rotation $G_{3,4}$ is now chosen so that the element h_{43} is annihilated, which carries the reduction one step further. To form H_{k+1} we must now compute

$$R_k Q_k + \tau_k I = R_k G_{12}^T G_{23}^T \cdots G_{n-1,n}^T + \tau_k I.$$

The product $R_k G_{12}^T$ will affect only the first two columns of R_k, which are replaced by linear combinations of one another. This will add a nonzero element in the $(2, 1)$ position. The rotation G_{23}^T will similarly affect the second and third columns in $R_k G_{12}^T$, and adds a nonzero element in the $(3, 2)$ position. The final result is a Hessenberg matrix.

If the shift τ is chosen as an exact eigenvalue of H, then $H - \tau I = QR$ has a zero eigenvalue and thus is singular. Since Q is orthogonal R must be singular. Moreover, if H is unreduced then the first $n-1$ columns of $H - \tau I$ are independent and therefore the *last* diagonal element r_{nn} must vanish. Hence, the last row in RQ is zero, and the elements in the last row of $H' = RQ + \tau I$ are $h'_{n,n-1} = 0$ and $h'_{nn} = \tau$,

The above result shows that if the shift is equal to an eigenvalue τ then the QR algorithm converges in one step to this eigenvalue. This indicates that τ should be chosen as an approximation to an eigenvalue λ. Then $h_{n,n-1}$ will converge to zero at least with linear rate equal to $|\lambda - \tau| / \min_{\lambda' \neq \lambda} |\lambda' - \tau|$. The choice

$$\tau = h_{nn} = e_n^T H e_n$$

is called the **Rayleigh quotient shift**, since it can be shown to produce the same sequence of shifts as the RQI starting with the vector $q_0 = e_n$. With this shift convergence is therefore *asymptotically quadratic*.

If H is real with complex eigenvalues, then we obviously cannot converge to a complex eigenvalue using only real shifts. We could shift by the eigenvalue of

$$C = \begin{pmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{n,n} \end{pmatrix}, \tag{9.4.23}$$

closest to $h_{n,n}$, although this has the disadvantage of introducing complex arithmetic even when A is real. A way to avoid this will be described later.

A important question is when to stop the iterations and accept an eigenvalue approximation. If

$$|h_{n,n-1}| \leq \epsilon(|h_{n-1,n-1}| + |h_{n,n}|),$$

where ϵ is a small constant times the unit roundoff, we set $h_{n,n-1} = 0$ and accept h_{nn} as an eigenvalue. This criterion can be justified since it corresponds to a small backward error. In practice the size of *all* subdiagonal elements should be monitored. Whenever

$$|h_{i,i-1}| \leq \epsilon(|h_{i-1,i-1}| + |h_{i,i}|),$$

for some $i < n$, we set $|h_{i,i-1}|$ and continue to work on smaller subproblems. This is important for the efficiency of the algorithm, since the work is proportional to the square of the dimension of the Hessenberg matrix. An empirical observation is that on the average less than two QR iterations per eigenvalue are required.

When the shift is explicitly subtracted from the diagonal elements this may introduce large relative errors in any eigenvalue much smaller than the shift. We now describe an **implicit-shift** QR-algorithm, which avoids this type of error. This is based on Theorem9.4.4, which says that the matrix H_{k+1} in a QR iteration (9.4.20) *is essentially uniquely defined by the first column in* Q_k*, provided it is unreduced.*

In the following, for simplicity, we drop the iteration index and write (9.4.20) as

$$H - \tau I = QR, \qquad H' = RQ + \tau I. \tag{9.4.24}$$

To apply Theorem 9.4.4 to the QR algorithm we must find the first column $q_1 = Qe_1$. From $H - \tau I = QR$ with R upper triangular it follows that

$$h_1 = (H - \tau I)e_1 = Q(Re_1) = r_{11}Qe_1 = r_{11}q_1.$$

If we choose a Givens rotation G_{12} so that

$$G_{12}^T h_1 = \pm\|h_1\|_2 e_1, \quad h_1 = (h_{11} - \tau, h_{21}, 0, \cdots, 0)^T,$$

then $G_{12}e_1$ is proportional to q_1, and (take $n = 6$)

$$G_{12}^T H = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times, \end{pmatrix} \quad G_{12}^T H G_{12} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

The multiplication from the right with G_{12} has introduced a nonzero element in the $(3, 1)$ position. To preserve the Hessenberg form a rotation G_{23} is chosen to zero this element.

$$G_{23}^T G_{12}^T H G_{12} G_{23} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & + & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

The result of this transformation is to push the element outside the Hessenberg form to position $(4, 2)$. We continue to chase the element $+$ down the diagonal, with rotations $G_{34}, \dots, G_{n-1,n}$ until it disappears below the nth row. Then we have obtained a Hessenberg matrix $Q^T HQ$, where the first column in Q equals

$$Qe_1 = G_{12}G_{23} \cdots G_{n-1,n}e_1 = G_{12}e_1.$$

From Theorem 9.4.4 it follows that the computed Hessenberg matrix is indeed H'. Note that the information of the shift τ is contained in G_{12}, and the shift is not explicitly subtracted from the other diagonal elements. The cost of one QR iteration is $4n^2$ flops.

The implicit QR algorithm can be generalized to perform $p > 1$ shifts $\tau_1, \dots, \tau_p$ in one step. These can be chosen to approximate several eigenvalues of A. By Theorem 9.4.4 Q is determined by its first column, which is proportional to the vector

$$QRe_1 = (H - \tau_p I) \cdots (H - \tau_1 I)e_1 = z_p.$$

The vector $z_1 = (H - \tau_1 I)e_1$ will have all of its entries except the first two equal to zero. The next $z_2 = (H - \tau_2 I)z_1$ is a linear combination of the first two columns of a Hessenberg matrix and therefore has all except its first three elements equal to zero. By induction it follows that z_p will have all but its first $p + 1$ elements equal to zero. The QR algorithm then starts with a Householder reflection P_0 such that $P_0 z_p = \beta e_1$. When this is applied from the right $P_0 A P_0$ it will create a "bulge" of $p(p + 1)/2p$ elements outside the Hessenberg form. The QR step is completed by chasing this bulge down the diagonal until it disappears.

To avoid complex arithmetic when H is real one can *adopt the implicit-shift QR algorithm to compute the real Schur form* in Theorem 9.1.12, where R is quasi-triangular with 1×1 and 2×2 diagonal blocks. For real matrices this will save a factor of 2–4 over using complex arithmetic. Let τ_1 and τ_2 be the eigenvalues of the matrix C in (9.4.23), and consider a double implicit QR iterations using these shifts. Proceeding as above we compute

$$QRe_1 = (H - \tau_2 I)(H - \tau_1 I)e_1 = (H^2 - (\tau_1 + \tau_2)H + \tau_1\tau_2 I)e_1.$$

where $(\tau_1 + \tau_2)$ and $\tau_1\tau_2$ are real. Taking out a factor $h_{21} \neq 0$ this can be written $h_{21}(p, q, r, 0, \dots, 0)^T$, where

$$\begin{aligned} p &= (h_{11}^2 - (\tau_1 + \tau_2)h_{11} + \tau_1\tau_2)/h_{21} + h_{12}, \\ q &= h_{11} + h_{22} - (\tau_1 + \tau_2), \qquad r = h_{32}. \end{aligned} \tag{9.4.25}$$

Note that we do not even have to compute τ_1 and τ_2, since we have $\tau_1 + \tau_2 = h_{n-1,n-1} + h_{n,n}$, and $\tau_1\tau_2 = \det(C)$. Substituting this into (9.4.25), and grouping terms to reduce roundoff errors, we get

$$\begin{aligned} p &= [(h_{nn} - h_{11})(h_{n-1,n-1} - h_{11}) - h_{n,n-1}h_{n-1,n}]/h_{21} + h_{12} \\ q &= (h_{22} - h_{11}) - (h_{nn} - h_{11}) - (h_{n-1,n-1} - h_{11}), \qquad r = h_{32}. \end{aligned}$$

The double QR step iteration can now be implemented by a chasing algorithm. We first choose rotations G_{23} and G_{12} so that $G_1^T g_1 = G_{12}^T G_{23}^T g_1 = \pm\|g_1\|_2 e_1$, and carry out a similarity transformation

$$G_1^T H = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}, \quad G_1^T H G_1 = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ + & + & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

To preserve the Hessenberg form we then choose the transformation $G_2 = G_{34} G_{23}$ to zero out the two elements $+$ in the first column. Then

$$G_2^T G_1^T H G_1 G_2 = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & + & \times & \times & \times & \times \\ & + & + & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

Note that this step is similar to the first step. The "bulge" of $+$ elements has now shifted one step down along the diagonal, and we continue to chase these elements until they disappear below the last row. We have then completed one double step of the implicit QR algorithm.

If we only want the eigenvalues, then it is not necessary to save the sequence of orthogonal transformations in the QR algorithm. Storing the rotations can be avoided by alternating pre-multiplications and post-multiplications. For example, once we have formed $G_{23} G_{12} H_k$ the first two columns do not enter in the remaining steps and we can perform the post-multiplication with G_{12}^T. In the next step we compute $(G_{34}((G_{23} G_{12} H_k) G_{12}^T)) G_{23}^T$, and so on.

Suppose the QR algorithm has converged to the final upper triangular matrix T. Then we have

$$P^T H P = T,$$

where P is the product of all Givens rotations used in the QR algorithm. The eigenvectors z_i, $i = 1 : n$ of T satisfy $T z_i = \lambda_i z_i$, $z_1 = e_1$, and z_i is a linear combination of $e_1, \dots, e_i$. The nonzero components of z_i can then be computed by back-substitution

$$z_{ii} = 1, \quad z_{ji} = -\Big(\sum_{k=j+1}^{i} t_{jk} z_{ki} \Big) \Big/ (\lambda_j - \lambda_i), \quad j = i-1 : (-1) : 1. \tag{9.4.26}$$

The eigenvectors of H are then given by $P z_i$, $i = 1 : n$. Finally $H = Q^T A Q$ has been obtained by reducing a matrix A to Hessenberg form as described in Sec. 9.4.2, then the eigenvectors of A can be computed from

$$x_i = Q P z_i, \quad i = 1 : n. \tag{9.4.27}$$

When only a few selected eigenvectors are wanted, then a more efficient way is to compute these by using inverse iteration. However, if more than a quarter of the eigenvectors are required, it is better to use the procedure outlined above.

It must be remembered that the matrix A may be defective, in which case there is no complete set of eigenvectors. In practice it is very difficult to take this into account, since with any procedure that involves rounding errors one cannot demonstrate that a matrix is defective. Usually one therefore should attempt to find a complete set of eigenvectors. If the matrix is nearly defective this will often be evident, in that corresponding computed eigenvectors will be almost parallel.

From the real Schur form $Q^T AQ = T$ computed by the QR algorithm, we get information about some of the invariant subspaces of A. If

$$T = \begin{pmatrix} T_{11} & T_{12} \\ & T_{22} \end{pmatrix}, \quad Q = (Q_1 \quad Q_2),$$

and $\lambda(T_{11}) \cap \lambda(T_{22}) = 0$, then Q_1 is an orthogonal basis for the unique invariant subspace associated with $\lambda(T_{11})$. However, this observation is useful only if we want the invariant subspace corresponding to a set of eigenvalues appearing at the top of the diagonal in T. Fortunately, it is easy to modify the real Schur decomposition so that an arbitrary set of eigenvalues are permuted to the top position. Clearly we can achieve this by performing a sequence of transformations, where in each step we interchange two nearby eigenvalues in the Schur form. Thus, we only need to consider the 2×2 case,

$$Q^T AQ = T = \begin{pmatrix} \lambda_1 & h_{12} \\ 0 & \lambda_2 \end{pmatrix}, \quad \lambda_1 \neq \lambda_2.$$

To reverse the order of the eigenvalues we note that $Tx = \lambda_2 x$ where

$$x = \begin{pmatrix} h_{12} \\ \lambda_2 - \lambda_1 \end{pmatrix}.$$

Let G^T be a Givens rotation such that $G^T x = \gamma e_1$. Then $G^T TG(G^T x) = \lambda_2 G^T x$, i.e. $G^T x$ is an eigenvector of $\hat{T} = GTG^T$. It follows that $\hat{T} e_1 = \lambda_2 e_1$ and $\hat{T}$ must have the form

$$\hat{Q}^T A\hat{Q} = \hat{T} = \begin{pmatrix} \lambda_2 & \pm h_{12} \\ 0 & \lambda_1 \end{pmatrix},$$

where $\hat{Q} = QG$.

9.4.4 Balancing an Unsymmetric Matrix

By (9.4.22) computed eigenvalues will usually have errors at least of order $u\|A\|_F$. Therefore, it is desirable to precede the eigenvalue calculation by a diagonal similarity transformation $\tilde{A} = D^{-1}AD$ which reduces the Frobenius norm. (Note that only the off-diagonal elements are effected by such a transformation.) This can be achieved by **balancing** the matrix A.

Definition 9.4.5.
A matrix $A \in \mathbf{R}^{n\times n}$ is said to be balanced in the norm $\|\cdot\|_p$ if

$$\|a_{:,i}\|_p = \|a_{i,:}\|_p, \quad i = 1:n,$$

where $a_{:,i}$ denotes the ith column and $a_{:,i}$ the ith row of A.

There are classes of matrices which do not need balancing; for example normal matrices are already balanced for $p = 2$. An iterative algorithm for balancing a matrix has been given by Osborne [294], which for any (real or complex) irreducible matrix A and $p = 2$ converges to a balanced matrix $\tilde{A}$. For a discussion and an implementation, see Contribution II/11 in [390] and Parlett and Reinsch [306]. More recent work on balancing a matrix has been done by Knight and Ruiz [240]

Example 9.4.1.
As an example consider the matrix

$$A = \begin{pmatrix} 1 & 0 & 10^{-4} \\ 1 & 1 & 10^4 \\ 10^4 & 10^2 & 1 \end{pmatrix}.$$

With $D = \mathrm{diag}\,(100, 1, 0.01)$ we get

$$B = DAD^{-1} = \begin{pmatrix} 1 & 0 & 1 \\ 10^{-2} & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

The Frobenius norm has been reduced from $\|A\|_F \approx 10^4$ to $\|B\|_F \approx 2.6$.

We describe a slightly simplified balancing algorithm. Let A_0 be the off-diagonal part of A. Note that a diagonal similarity leaves the main diagonal of A unchanged. Fromm A_0 a sequence of matrices $\{A_k\}$, $k = 1, 2, \dots$ is formed. The matrix A_k differs from A_{k-1} only in the ith row and column, where i is given by $i - 1 \equiv k - 1 \bmod n$. That is the rows and columns are modified cyclically in the natural order. In step k, let

$$\alpha_k = \|a_{:,i}\|_p, \qquad \beta_k = \|a_{i,:}\|_p.$$

Usually the matrix is balanced in the 1-norm since this requires fewer multiplications than the 2-norm. Assume that $\alpha_k\beta_k \neq 0$ and set

$$\bar{D}_k = I + \gamma_k e_i e_i^T, \quad \gamma_k = \alpha_k/\beta_k,$$

and $D_k = \bar{D}_k D_{k-1}$. Then the matrix

$$A_k = \bar{D}_k A_{k-1} \bar{D}_k^{-1} = D_k A_0 D_k^{-1}$$

will be balanced in its ith row and columns.

The above iterative process will under some conditions converge to a balanced matrix. However, convergence is linear and can be slow.

Note that there are classes of matrices which do not need balancing. For example, normal matrices are already balanced in the 2-norm. Also, there is no need to balance the matrix if an eigenvalue algorithm is to be used which is invariant under scaling as, e.g., some vector iterations.

Review Questions

4.1 (a) Describe how an arbitrary square matrix can be reduced to Hessenberg form by a sequence of orthogonal similarity transformations.

(b) Show that the Hessenberg form is preserved by the QR algorithm.

4.2 What is meant by a graded matrix, and what precautions need to be taken when transforming such a matrix to condensed form?

4.3 If one step of the QR algorithm is performed on A with a shift τ equal to an eigenvalue of A, what can you say about the result? Describe how the shift usually is chosen in the QR algorithm applied to a real symmetric tridiagonal matrix.

4.4 What are the advantages of the implicit shift version of the QR algorithm for a real Hessenberg matrix H?

4.5 Suppose the eigenvalues to a Hessenberg matrix have been computed using the QR algorithm. How are the eigenvectors best computed (a) if all eigenvectors are needed; (b) if only a few eigenvectors are needed.

4.6 What is meant by balancing a matrix $A \in \mathbf{R}^{n\times n}$?. Why can it be advantageous to balance a matrix before computing its eigenvalues?

Problems

4.1 (a) Let L and U be the bidiagonal matrices (take $n = 4$)

$$L = \begin{pmatrix} 1 & & & \\ e_2 & 1 & & \\ & e_3 & 1 & \\ & & e_4 & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} q_1 & 1 & & \\ & q_2 & 1 & \\ & & q_3 & 1 \\ & & & q_4 \end{pmatrix}.$$

Consider the matrix equation

$$\hat{L}\hat{U} = UL,$$

where $\hat{L} = (\hat{l}_{ij})$ and $\hat{U} = (\hat{u}_{ij})$ are two new bidiagonal matrices of the same form. Show that both LU and $\hat{U}\hat{L}$ are tridiagonal matrices with all super-diagonal elements equal to one.

(b) Show that, setting $e_1 = \hat{e}_5 = 0$, the remaining nonzero elements in $\hat{L}$ and $\hat{U}$ are determined by the relations

$$\hat{e}_m + \hat{q}_{m-1} = e_m + q_m, \qquad \hat{e}_m\hat{q}_m = e_m q_{m+1}$$

which are the rhombus rules in Rutishauser's qd algorithm.

4.2 Let A be the matrix in Example 9.4.1. Apply the balancing procedure described in Sec. 9.4.4 to A. Use the 1-norm and terminate the iterations when the matrix is balanced to a tolerance equal to 0.01. How much is the Frobenius norm reduced?

4.3 The reduction to Hessenberg form can also be achieved by using elementary elimination matrices of the form

$$L_j = I + m_j e_j^T, \quad m_j = (0, \dots, 0, m_{j+1,j}, \dots, m_{n,j})^T.$$

Only the elements *below* the main diagonal in the jth column differ from the unit matrix. If a matrix A is pre-multiplied by L_j we get

$$L_j A = (I + m_j e_j^T)A = A + m_j(e_j^T A) = A + m_j a_j^T,$$

i.e., multiples of the row a_j^T are *added* to the last $n-j$ rows of A. The similarity transformation $L_j A L_j^{-1} = \tilde{A} L_j^{-1}$ is completed by post-multiplying

$$\tilde{A} L_j^{-1} = \tilde{A}(I - m_j e_j^T) = \tilde{A} - (\tilde{A} m_j) e_j^T.$$

Show that in this operation a linear combination $\tilde{A} m_j$ of the last $n-j$ columns is *subtracted* from the jth column of $\tilde{A}$.

9.5 Hermitian Eigenvalue Algorithms

9.5.1 Reduction to Symmetric Tridiagonal Form

Before applying the QR algorithm, the first step is the orthogonal reduction to Hessenberg form. If we carry out this reduction for a real symmetric matrix A, then

$$H^T = (Q^T A Q)^T = Q^T A^T Q = H.$$

It follows that H is a *real symmetric tridiagonal matrix*, which we write

$$Q^T A Q = T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}. \tag{9.5.1}$$

As in the unsymmetric case, if T is unreduced, that is, $\beta_k \neq 0$, $k = 2:n$, then the decomposition is uniquely determined by $q_1 = Qe_1$. If T has a zero subdiagonal element, then it decomposes into block diagonal form

$$T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix}$$

with symmetric tridiagonal blocks T_1 and T_2 and the eigenproblem splits into smaller pieces.

It is important to take advantage of symmetry to save storage and operations. In the kth step of the orthogonal reduction we compute $A^{(k+1)} = P_k A^{(k)} P_k$, where P_k is again chosen to zero the last $n-k-1$ elements in the kth column. By symmetry the corresponding elements in the kth row will be zeroed by the post-multiplication P_k. However, the intermediate matrix $P_k A^{(k)}$ is not symmetric. Therefore, we must compute $P_k A^{(k)} P_k$ directly. Dropping the subscripts k we can write

$$\begin{aligned} PAP &= \Big(I - \frac{1}{\gamma}uu^T\Big)A\Big(I - \frac{1}{\gamma}uu^T\Big) && (9.5.2)\\ &= A - up^T - pu^T + u^T p u u^T/\gamma && (9.5.3)\\ &= A - uq^T - qu^T, \end{aligned}$$

where

$$p = Au/\gamma, \quad q = p - \beta u, \quad \beta = u^T p/(2\gamma). \tag{9.5.4}$$

If the transformations are carried out in this fashion the operation count for the reduction to tridiagonal form is reduced to about $2n^3/3$ flops,and we only need to store, say, the lower halves of the matrices.

The orthogonal reduction to tridiagonal form has the same stability property as the corresponding algorithm for the unsymmetric case, i.e., the computed tridiagonal matrix is the exact result for a matrix $A+E$, where E satisfies (9.4.22). Hence, the eigenvalues of T will differ from the eigenvalues of A by at most $cn^2u\|A\|_F$.

There is a class of symmetric matrices for which small eigenvalues are determined with a very small error compared to $\|A\|_F$. This is the class of **scaled diagonally dominant** matrices, see Barlow and Demmel [18, 1990]. A symmetric scaled diagonally dominant (s.d.d) matrix is a matrix of the form DAD, where A is symmetric and diagonally dominant in the usual sense, and D is an arbitrary diagonal matrix. An example of a s.d.d. matrix is the **graded matrix**

$$A_0 = \begin{pmatrix} 1 & 10^{-4} & \\ 10^{-4} & 10^{-4} & 10^{-8} \\ & 10^{-8} & 10^{-8} \end{pmatrix}$$

whose elements decrease progressively in size as one proceeds diagonally from top to bottom. However, the matrix

$$A_1 = \begin{pmatrix} 10^{-6} & 10^{-2} & \\ 10^{-2} & 1 & 10^{-2} \\ & 10^{-2} & 10^{-6} \end{pmatrix}.$$

is neither diagonally dominant or graded in the usual sense.

The matrix A_0 has an eigenvalue λ of magnitude 10^{-8}, which is quite insensitive to small *relative* perturbations in the elements of the matrix. If the Householder reduction is performed starting from the *top* row of A as described here it is important that the matrix is presented so that the larger elements of A occur in the top left-hand corner. Then the errors in the orthogonal reduction will correspond to small relative errors in the elements of A, and the small eigenvalues of A will not be destroyed.[38]

A similar algorithm can be used to transform a Hermitian matrix into a tridiagonal Hermitian matrix using the complex Householder transformation introduced in Sec. 8.1.6. With $U = P_1P_2\cdots P_{n-2}$ we obtain $T = U^H AU$, where T is Hermitian and therefore has positive real diagonal elements. By a diagonal similarity DTD^{-1}, $D = \operatorname{diag}(e^{i\phi_1}, e^{i\phi_2}, \dots, e^{i\phi_n})$ it is possible to further transform T so that the off-diagonal elements are real and nonnegative.

If the orthogonal reduction to tridiagonal form is carried out for a symmetric banded matrix A, then the banded structure will be destroyed. By annihilating pairs of elements using Givens rotations in an ingenious order it is possible to perform the reduction *without* increasing the bandwidth. However, it will then take several rotation to eliminate a single element. This algorithm is described in Parlett [305, Sec. 10.5.1], see also Contribution II/8 in Wilkinson and Reinsch [390]. An operation count shows that the standard reduction is slower if the bandwidth is less than $n/6$. Note that the reduction of storage is often equally important!

When combined with a preliminary reduction to Hessenberg or symmetric tridiagonal form (see Sec. 9.4) the QR algorithm yields a very efficient method for finding all eigenvalues and eigenvectors of small to medium size matrices. Then the necessary modifications to make it into a practical method are described. The general nonsymmetric case is treated in Sec. 9.4.3 and the real symmetric case in Sec. 9.5.2.

9.5.2 The Hermitian QR Algorithm

When A is real symmetric and positive definite we can modify the LR algorithm and use the Cholesky factorization $A = LL^T$ instead. The algorithm then takes the form

$$A_k = L_kL_k^T, \qquad A_{k+1} = L_k^TL_k, \quad k = 1, 2, \dots. \tag{9.5.5}$$

and we have

$$A_{k+1} = L_k^{-1}A_kL_k = L_k^TA_kL_k^{-T}. \tag{9.5.6}$$

Clearly all matrices A_k are symmetric and positive definite and the algorithm is well defined. Repeated application of (9.5.6) gives

$$A_k = T_{k-1}^{-1}A_1T_{k-1} = T_{k-1}^TA_1(T_{k-1}^{-1})^T, \tag{9.5.7}$$

where $T_k = L_1L_2\cdots L_k$. Further, we have

$$A_1^k = (L_1L_2\cdots L_k)(L_k^T\cdots L_2^TL_1^T) = T_kT_k^T. \tag{9.5.8}$$

When A is real symmetric and positive definite there is a close relationship between the LR and QR algorithms. For the QR algorithm we have $A_k^T = A_k = R_k^TQ_k^T$ and hence

$$A_k^TA_k = A_k^2 = R_k^TQ_k^TQ_kR_k = R_k^TR_k, \tag{9.5.9}$$

[38]Note that in the Householder tridiagonalization described in [390], Contribution II/2 the reduction is performed instead from the bottom up.

which shows that R_k^T is the lower triangular Cholesky factor of A_k^2.

For the Cholesky LR algorithm we have from (9.4.4) and (9.4.5)

$$A_k^2 = L_k L_{k+1} (L_k L_{k+1})^T. \tag{9.5.10}$$

These two Cholesky factorizations (9.5.9) and (9.5.9) of the matrix A_k^2 must be the same and therefore $R_k^T = L_k L_{k+1}$. Thus

$$A_{k+1} = R_k Q_k = R_k A_k R_k^{-1} = L_{k+1}^T L_k^T A_k (L_{k+1}^T L_k^T)^{-1}.$$

Comparing this with (9.5.7) we deduce that one step of the QR algorithm is equivalent to two steps in the Cholesky LR algorithm. Hence, the matrix $A_{(2k+1)}$ obtained by the Cholesky LR algorithm equals the matrix $A_{(k+1)}$ obtained using the QR algorithm.

By the methods described in Sec. 9.4 any Hermitian (real symmetric) matrix can by a unitary (orthogonal) similarity transformation be reduced into real, symmetric tridiagonal form. We can also assume without restriction that T is unreduced, since otherwise it can be split up in smaller unreduced tridiagonal matrices.

If T is unreduced and λ an eigenvalue of T. then clearly rank $(T - \lambda I) = n - 1$ (the submatrix obtained by crossing out the first row and last column of $T - \lambda I$ has nonzero determinant, $\beta_2 \cdots \beta_n \neq 0$). Hence, there is only one eigenvector corresponding to λ and since T is diagonalizable λ must have multiplicity one. *It follows that all eigenvalues of an unreduced symmetric tridiagonal matrix are distinct.*

The QR algorithm also preserves symmetry. Hence, it follows that if T is symmetric tridiagonal, and

$$T - \tau I = QR, \qquad T' = RQ + \tau I, \tag{9.5.11}$$

then also $T' = Q^T T Q$ is symmetric tridiagonal.

From Theorem 9.4.4) we have the following result, which can be used to develop an implicit QR algorithm.

Theorem 9.5.1.

Let A be real symmetric, $Q = (q_1, \ldots, q_n)$ orthogonal, and $T = Q^T A Q$ an unreduced symmetric tridiagonal matrix. Then Q and T are essentially uniquely determined by the first column q_1 of Q.

Suppose we can find an orthogonal matrix Q with the same first column q_1 as in (9.5.11) such that $Q^T A Q$ is an unreduced tridiagonal matrix. Then by Theorem 9.5.1 it must be the result of one step of the QR algorithm with shift τ. Equating the first columns in $T - \tau I = QR$ it follows that $r_{11} q_1$ equals the first column t_1 in $T - \tau I$. In the implicit shift algorithm a Givens rotation G_{12} is chosen so that

$$G_{12}^T t_1 = \pm \|t_1\|_2 e_1, \qquad t_1 = (\alpha_1 - \tau, \beta_2, 0, \cdots, 0)^T.$$

We now perform the similarity transformation $G_{12}^T T G_{12}$, which results in fill-in in positions (1,3) and (3,1), pictured below for $n = 5$:

$$G_{12}^T T = \begin{pmatrix} \times & \times & + & & & \\ \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}, \quad G_{12}^T T G_{12} = \begin{pmatrix} \times & \times & + & & & \\ \times & \times & \times & & & \\ + & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

To preserve the tridiagonal form a rotation G_{23} can be used to zero out the fill-in elements.

$$G_{23}^T G_{12}^T \; T \; G_{12} G_{23} = \begin{pmatrix} \times & \times & & & & \\ \times & \times & \times & + & & \\ & \times & \times & \times & & \\ & + & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

We continue to "chase the bulge" of + elements down the diagonal, with transformations $G_{34}, \ldots, G_{n-1,n}$ after which it disappears. We have then obtained a symmetric tridiagonal matrix $Q^T T Q$, where the first column in Q is

$$G_{12} G_{23} \cdots G_{n-1,n} e_1 = G_{12} e_1.$$

By Theorem 9.4.4 it follows that the result must be the matrix T' in (9.5.11).

There are several possible ways to choose the shift. Suppose that we are working with the submatrix ending with row r, and that the current elements of the two by two trailing matrix is

$$\begin{pmatrix} \alpha_{r-1} & \beta_r \\ \beta_r & \alpha_r \end{pmatrix}, \tag{9.5.12}$$

The Rayleigh quotient shift $\tau = \alpha_r$, gives the same result as Rayleigh Quotient Iteration starting with e_r. This leads to generic cubic convergence, but not guaranteed. In practice, taking the shift to be the eigenvalue of the 2×2 trailing submatrix (9.5.12), closest to α_r, has proved to be more efficient. This is called the **Wilkinson shift**. In case of a tie ($\alpha_{r-1} = \alpha_r$) the smaller $\alpha_r - |\beta_r|$ is chosen. A suitable formula for computing this shift is

$$\tau = \alpha_r - \beta_r^2 \Big/ \left(|\delta| + \operatorname{sign}(\delta) \sqrt{\delta^2 + \beta_r^2} \right), \quad \delta = (\alpha_{r-1} - \alpha_r)/2 \tag{9.5.13}$$

(cf. Algorithm (9.5.6)). A great advantage of the Wilkinson shift is that it gives guaranteed *global* convergence.[39] It can also be shown to give almost always *local cubic convergence*, although quadratic convergence might be possible.

[39] For a proof see Wilkinson [388] or Parlett [305, Chapter 8].

Example 9.5.1. Consider an unreduced tridiagonal matrix of the form

$$T = \begin{pmatrix} \times & \times & 0 \\ \times & \times & \epsilon \\ 0 & \epsilon & t_{33} \end{pmatrix}.$$

Show, that with the shift $\tau = t_{33}$, the first step in the reduction to upper triangular form gives a matrix of the form

$$G_{12}(T - sI) = \begin{pmatrix} \times & \times & s_1\epsilon \\ 0 & a & c_1\epsilon \\ 0 & \epsilon & 0 \end{pmatrix}.$$

If we complete this step of the QR algorithm, $QR = T - \tau I$, the matrix $\hat{T} = RQ + \tau I$, has elements

$$\hat{t}_{32} = \hat{t}_{23} = -c_1\epsilon^3/(\epsilon^2 + a^2).$$

This shows that if $\epsilon \ll$ the QR method tends to converge cubically.

As for the QR algorithm for unsymmetric matrices it is important to check for negligible subdiagonal elements using the criterion

$$|\beta_i| \le \epsilon\,(|\alpha_{i-1}| + |\alpha_i|).$$

When this criterion is satisfied for some $i < n$, we set β_i equal to zero and the problem decouples. At any step we can partition the current matrix so that

$$T = \begin{pmatrix} T_{11} & & \\ & T_{22} & \\ & & D_3 \end{pmatrix},$$

where D_3 is diagonal and T_{22} is unreduced. The QR algorithm is then applied to T_{22}.

We will not give more details of the algorithm here. If full account of symmetry is taken then one QR iteration can be implemented in only $9n$ multiplications, $2n$ divisions, $n - 1$ square roots and $6n$ additions. By reorganizing the inner loop of the QR algorithm, it is possible to eliminate square roots and lower the operation count to about 4n multiplications, 3n divisions and 5n additions. This **rational QR algorithm** is the fastest way to get the eigenvalues alone, but does not directly yield the eigenvectors.

The Wilkinson shift may not give the eigenvalues in monotonic order. If some of the smallest or largest eigenvalues are wanted, then it is usually recommended to use Wilkinson shifts anyway and risk finding a few extra eigenvalues. To check if all wanted eigenvalues have been found one can use spectrum slicing, see Sec. 9.5.4. For a detailed discussion of variants of the symmetric tridiagonal QR algorithm, see Parlett [305].

If T has been obtained by reducing a Hermitian matrix to real symmetric tridiagonal form, $U^HAU = T$, then the eigenvectors are given by

$$x_i = UPe_i, \quad i = 1 : n, \tag{9.5.14}$$

where $P = Q_0Q_1Q_2\cdots$ is the product of all transformations in the QR algorithm. Note that the eigenvector matrix $X = UP$ will by definition be orthogonal.

If eigenvectors are to be computed, the cost of a QR iteration goes up to $4n^2$ flops and the overall cost to $O(n^3)$. To reduce the number of QR iterations where we accumulate transformations, we can first compute the eigenvalues *without* accumulating the product of the transformations. We then perform the QR algorithm again, now shifting with the computed eigenvalues, the **perfect shifts**, convergence occurs in one iteration. This may reduce the cost of computing eigenvectors by about 40%. As in the unsymmetric case, if fewer than a quarter of the eigenvectors are wanted, then inverse iteration should be used instead. The drawback of this approach, however, is the difficulty of getting orthogonal eigenvectors to clustered eigenvalues.

For symmetric tridiagonal matrices one often uses the QL algorithm instead of the QR algorithm. We showed in Sec. 9.4.1 that the QL algorithm is just the QR algorithm on JAJ, where J is the permutation matrix that reverses the elements in a vector. If A is tridiagonal then JAJ is tridiagonal with the diagonal elements in reverse order.

In the implicit QL algorithm one chooses the shift from the top of A and chases the bulge from bottom to top. The reason for preferring the QL algorithm is simply that in practice it is often the case that the tridiagonal matrix is graded with the large elements at the bottom. Since for reasons of stability the small eigenvalues should be determined first the QL algorithm is preferable in this case. For matrices graded in the other direction the QR algorithm should be used, or rows and columns reversed before the QL algorithm is applied.

9.5.3 The QR–SVD Algorithm

The SVD of a matrix $A \in \mathbf{R}^{m\times n}$ is closely related to a symmetric eigenvalue problem, and hence a QR algorithm for the SVD can be developed. We assume in the following that $m \geq n$. This is no restriction, since otherwise we can consider A^T.

It is usually advisable to compute as a first step the QR decomposition with column pivoting of A,

$$A\Pi = Q\begin{pmatrix} R \\ 0 \end{pmatrix}. \tag{9.5.15}$$

Then, if $R = U_R\Sigma V^T$ is the SVD of R, it follows that

$$A\Pi = U\Sigma V^T, \quad U = Q\begin{pmatrix} U_R \\ 0 \end{pmatrix}. \tag{9.5.16}$$

Hence, the singular values and the right singular vectors of $A\Pi$ and R are the same and the first n left singular vectors of A are easily obtained from those of R.

We first consider the unshifted QR algorithm. Set $R_1 = R$, and compute a sequence of upper triangular matrices R_{k+1}, $k = 1, 2, 3, \ldots$, as follows. In step k, R_{k+1} is computed from the QR factorization of the *lower* triangular matrix

$$R_k^T = Q_{k+1}R_{k+1}, k = 1, 2, 3, \ldots. \tag{9.5.17}$$

Using (9.5.17) we observe that

$$R_k^T R_k = Q_{k+1}(R_{k+1}R_k)$$

is the QR factorization of $R_k^T R_k$. Forming the product in reverse order gives

$$\begin{aligned}(R_{k+1}R_k)Q_{k+1} &= R_{k+1}R_{k+1}^T Q_{k+1}^T Q_{k+1} = R_{k+1}R_{k+1}^T \\ &= R_{k+2}^T Q_{k+2}^T Q_{k+2} R_{k+2} = R_{k+2}^T R_{k+2}.\end{aligned}$$

Hence, two successive iterations of (9.5.17) are equivalent to one iteration of the basic QR algorithm for $R^T R$. Moreover this is achieved without forming $R^T R$, which is essential to avoid loss of accuracy.

Using the orthogonality of Q_{k+1} it follows from (9.5.17) that $R_{k+1} = Q_{k+1}^T R_k^T$, and hence

$$R_{k+1}^T R_{k+1} = R_k(Q_{k+1}Q_{k+1}^T)R_k^T = R_k R_k^T.$$

Further, we have

$$R_{k+2}R_{k+2}^T = R_{k+2}R_{k+1}Q_{k+2} = Q_{k+2}^T(R_k R_k^T)Q_{k+2}. \tag{9.5.18}$$

which shows that we are simultaneously performing an iteration on $R_k R_k^T$, again without explicitly forming this matrix.

One iteration of (9.5.17) is equivalent to one iteration of the Cholesky LR algorithm applied to $B_k = R_k R_k^T$. This follows since B_k has the Cholesky factorization $B_k = R_{k+1}^T R_{k+1}$ and multiplication of these factors in reverse order gives $B_{k+1} = R_{k+1}R_{k+1}^T$. (Recall that for a symmetric, positive definite matrix two steps of the LR algorithm is equivalent to one step of the QR algorithm.)

The convergence of this algorithm is enhanced provided the QR factorization of A in the first step is performed using column pivoting. It has been shown that then already the diagonal elements of R_2 are often surprisingly good approximations to the singular values of A.

For the QR–SVD algorithm to be efficient it is necessary to first reduce A further to bidiagonal form and to introduce shifts. It was shown in Sec. 8.4.6 that any matrix $A \in \mathbf{R}^{m\times n}$ can be reduced to upper bidiagonal form by a sequence of Householder transformations alternatingly from left and right. Performing this reduction on R we have $Q_B^T R P_B = B$, where

$$B = \begin{pmatrix} q_1 & r_2 & & & \\ & q_2 & r_3 & & \\ & & \ddots & \ddots & \\ & & & q_{n-1} & r_n \\ & & & & q_n \end{pmatrix}, \tag{9.5.19}$$

where

$$Q_B = Q_1 \cdots Q_n \in \mathbf{R}^{n\times n}, \qquad P_B = P_1 \cdots P_{n-2} \in \mathbf{R}^{n\times n}.$$

This reduction can be carried out in $\frac{4}{3}n^3$ flops. If Q_B and P_B are explicitly required they can be accumulated at a cost of $2(m^2n - mn^2 + \frac{1}{3}n^3)$ and $\frac{2}{3}n^3$ flops respectively.

The singular values of B equal those of A and the left and right singular vectors can be constructed from those of B. A complex matrix can be reduced to *real* bidiagonal form using complex Householder transformations.

We notice that if in (9.5.19) $r_i = 0$, then the matrix B breaks into two upper bidiagonal matrices, for which the singular values can be computed independently. If $q_i = 0$, then B has a singular value equal to zero. Applying a sequence of Givens rotations from the left, $G_{i,i+1}, G_{i,i+2}, \dots, G_{i,n}$ the ith row be zeroed out, and again the matrix breaks up into two parts. Hence, we may without loss of generality assume that none of the elements $q_1, q_i, r_i,\ i = 2 : n$ are zero. This assumption implies that the matrix $B^T B$ has nondiagonal elements $\alpha_{i+1} = q_i r_{i+1} \neq 0$, and hence is unreduced. It follows that all eigenvalues of $B^T B$ are positive and distinct, and we have $\sigma_1 > \cdots > \sigma_n > 0$.

From Theorem 9.2.11 it follows that it should be possible to compute all singular values of a bidiagonal matrix to *full relative precision independent of their magnitudes*. For the small singular values this can be achieved by using the *unshifted* QR–SVD algorithm given by (9.5.17). This uses the iteration

$$B_k^T = Q_{k+1} B_{k+1}, \quad k = 0, 1, 2, \dots. \tag{9.5.20}$$

In each step the lower bidiagonal matrix B_k^T is transformed into an upper bidiagonal matrix B_{k+1}.

$$Q_1^T B = \begin{array}{c} \rightarrow \\ \rightarrow \\ \\ \\ \\ \end{array} \begin{pmatrix} \times & + & & & \\ \otimes & \times & & & \\ & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \end{pmatrix}, \quad Q_2 Q_1^T B = \begin{array}{c} \\ \rightarrow \\ \rightarrow \\ \\ \\ \end{array} \begin{pmatrix} \times & \times & & & \\ & \times & + & & \\ & \otimes & \times & & \\ & & \times & \times & \\ & & & \times & \times \end{pmatrix},$$

etc. Each iteration in (9.5.20) can be performed with a sequence of $n-1$ Givens rotations at a cost of only $2n$ multiplications and $n-1$ calls to givrot. Two steps of the iteration is equivalent to one step of the zero shift QR algorithm. (Recall that one step of the QR algorithm with nonzero shifts, requires $12n$ multiplications and $4n$ additions.) The zero shift algorithm is very simple and uses no subtractions, Hence, each entry of the transformed matrix is computed to high *relative* accuracy.

Algorithm 9.1. The Zero Shift QR Algorithm.
The algorithm performs p steps of the zero shift QR algorithm on the bidiagonal matrix B in (9.5.19):

```
for k = 1 : 2p
  for i = 1 : n − 1
    [c, s, r] = givrot(q_i, r_{i+1});
    q_i = r;   q_{i+1} = q_{i+1} * c;
    r_{i+1} = q_{i+1} * s;
  end
end
```

If two successive steps of the unshifted QR–SVD algorithm are interleaved we get the **zero shift QR algorithm**. The implementation of this has been studied in depth by Demmel and Kahan [96]. To give full accuracy for the smaller singular values the convergence tests used for standard shifted QR–SVD algorithm must be modified. This is a non-trivial task, for which we refer to the original paper.

For achieving rapid convergence when computing all singular values of a matrix shifts are essential. We now look at the application of the implicit shift QR algorithm to BB^T. Since forming BB^T could lead to a severe loss of accuracy in the small singular values and vectors it is essential to work directly with the matrix B. We use the Wilkinson shift for BB^T since this is known to guarantee global convergence. Thus, the shift is determined from the trailing 2×2 submatrix in BB^T,[40]

$$B_2 B_2^T = \begin{pmatrix} q_{n-1}^2 + r_n^2 & q_n r_n \\ q_n r_n & q_n^2 \end{pmatrix}.$$

We note that the sum and product of the eigenvalues are

$$\lambda_1 + \lambda_2 = \operatorname{trace}(B_2 B_2^T) = q_{n-1}^2 + q_n^2 + r_n^2, \qquad \lambda_1 \lambda_2 = \det(B_2 B_2^T) = (q_{n-1} q_n)^2.$$

The eigenvalue closes to q_n^2 is chosen as the shift. Using the formula (9.5.13) for computing the shift we obtain

$$\tau = q_n^2 - \operatorname{sign}(\delta)(q_n r_n)^2 / (|\delta| + \sqrt{\delta^2 + (q_n r_n)^2}), \tag{9.5.21}$$

where

$$\delta = \frac{1}{2}((q_n + q_{n-1})(q_n - q_{n-1}) - r_n^2).$$

These expressions should not be used directly, since they suffer from possible overflow or underflow in the squared subexpressions. A method based on these expressions, which computes the singular values and vectors with high *relative accuracy* is given by Demmel and Kahan [96, 1990].

In the implicit shift QR algorithm for $B^T B$ we first determine a Givens rotation $T_1 = G_{12}$ so that $G_{12}^T t_1 = \pm \|t_1\|_2 e_1$, where

$$t_1 = (BB^T - \tau I) r_1 = \begin{pmatrix} q_1^2 + r_2^2 - \tau \\ q_2 r_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{9.5.22}$$

where t_1 is the first column in $B^T B - \tau I$ and τ is the shift. Suppose we next apply a sequence of Givens transformations such that

$$T_{n-1}^T \cdots T_2^T T_1^T B B^T T_1 T_2 \cdots T_{n-1}$$

[40]Golub and Reinsch [181] use the trailing 2×2 submatrix of $B^T B$, which leads to a slightly different shift.

is tridiagonal, but we wish to avoid doing this explicitly. Let us start by applying the transformation T_1 to B. Then we get (take $n = 5$),

$$BT_1 = \begin{array}{c} \rightarrow \\ \rightarrow \\ \\ \\ \\ \end{array} \begin{pmatrix} \times & \times & & & \\ + & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{pmatrix}.$$

If we now pre-multiply by a Givens rotation $S_1^T = G_{12}$ to zero out the $+$ element, this creates a new nonzero element in the $(1, 3)$ position; To preserve the bidiagonal form we then choose the transformation $T_2 = R_{23}$ to zero out the element $+$:

$$S_1^T BT_1 = \begin{array}{c} \rightarrow \\ \rightarrow \\ \\ \\ \\ \end{array} \begin{pmatrix} \times & \times & + & & \\ \oplus & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{pmatrix}, \quad S_1^T BT_1T_2 = \begin{array}{c} \begin{array}{ccccc} & \downarrow & \downarrow & & \end{array} \\ \begin{pmatrix} \times & \times & \oplus & & \\ & \times & \times & & \\ & + & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{pmatrix} \end{array}.$$

We can now continue to chase the element $+$ down, with transformations alternately from the right and left until we get a new bidiagonal matrix

$$\hat{B} = (S_{n-1}^T \cdots S_1^T)B(T_1 \cdots T_{n-1}) = U^T BP.$$

But then the matrix

$$\hat{T} = \hat{B}^T \hat{B} = P^T B^T U U^T BP = P^T TP$$

is tridiagonal, where the first column of P equals the first column of T_1. Hence, if $\hat{T}$ is unreduced it must be the result of one QR iteration on $T = B^T B$ with shift equal to τ.

The subdiagonal entries of T equal $q_i e_{i+1}$, $i = 1 : n - 1$. If some element e_{i+1} is zero, then the bidiagonal matrix splits into two smaller bidiagonal matrices

$$B = \begin{pmatrix} B_1 & 0 \\ 0 & B_2 \end{pmatrix}.$$

If $q_i = 0$, then we can zero the ith row by pre-multiplication by a sequence Givens transformations $R_{i,i+1}, \dots, R_{i,n}$, and the matrix then splits as above. In practice two convergence criteria are used. After each QR step if

$$|r_{i+1}| \leq 0.5u(|q_i| + |q_{i+1}|),$$

where u is the unit roundoff, we set $r_{i+1} = 0$. We then find the smallest p and the largest q such that B splits into quadratic subblocks

$$\begin{pmatrix} B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \end{pmatrix},$$

of dimensions $p, n-p-q$ and, q where B_3 is diagonal and B_2 has a nonzero subdiagonal. Second, if diagonal elements in B_2 satisfy

$$|q_i| \le 0.5u(|r_i| + |r_{i+1}|),$$

set $q_i = 0$, zero the superdiagonal element in the same row, and repartition B. Otherwise continue the QR algorithm on B_2. A justification for these tests is that roundoff in a rotation could make the matrix indistinguishable from one with a q_i or r_{i+1} equal to zero. Also, the error introduced by the tests is not larger than some constant times $u\|B\|_2$. When all the superdiagonal elements in B have converged to zero we have $Q_S^T B T_S = \Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_n)$. Hence

$$U^T A V = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}, \quad U = Q_B \operatorname{diag}(Q_S, I_{m-n}), \quad V = T_B T_S \tag{9.5.23}$$

is the singular value decomposition of A.

Usually less than $2n$ iterations are needed in the second phase. One QR iteration requires $14n$ multiplications and $2n$ calls to givrot. Accumulating the rotations into U requires $6mn$ flops. Accumulating the rotations into V requires $6n^2$ flops. If both left and right singular vectors are desired, the cost of one QR iteration increases to $4n^2$ flops and the overall cost to $O(n^3)$. Note that if the SVD is to be used for solving a least squares problem $\min_x \|Ax - b\|_2$, then the left singular vectors U need not be saved or accumulated since they can be applied directly to the right hand side b. Asymptotic flop counts for some different variants are summarized in Table 9.5.3. As usual, only the highest order terms in m and n are shown.

Table 9.5.1. *Approximate flop counts for the QR–SVD algorithm.*

Option	Golub–Reinsch SVD
Σ, U_1, V	$14mn^2 + \frac{22}{3}n^3$
Σ, U_1	$14mn^2 - 2n^3$
Σ, V	$4n^2(m + 2n)$
Σ	$4n^2(m - n/3)$

The implicit QR–SVD algorithm can be shown to be backward stable. This essentially follows from the fact that we have only applied a sequence of orthogonal transformations to A. Hence, the computed singular values $\bar{\Sigma} = \operatorname{diag}(\bar{\sigma}_k)$ are the exact singular values of a nearby matrix $A + E$, where $\|E\|_2 \le c(m,n) \cdot u\sigma_1$. Here $c(m,n)$ is a constant depending on m and n and u the unit roundoff. From Theorem 8.1.50 it follows that

$$|\bar{\sigma}_k - \sigma_k| \le c(m,n) \cdot u\sigma_1. \tag{9.5.24}$$

Thus, if A is nearly rank deficient, this will always be revealed by the computed singular values. Note, however, that the smaller singular values may not be computed with high relative accuracy.

The backward error bound (9.5.24) does not guarantee that small singular values of A are computed with small *relative* accuracy. If A has rows and columns of widely varying norm the accuracy can be improved by first sorting the rows after decreasing norm and then performing a QR decomposition of PA using column pivoting.

An important implementation issue is that the bidiagonal matrix is often graded, i.e., the elements may be large at one end and small at the other. If an initial QR decomposition of A with column pivoting has been done the bidiagonal matrix is usually graded from large at upper left to small at lower right as illustrated below

$$\begin{pmatrix} 1 & 10^{-1} & & \\ & 10^{-2} & 10^{-3} & \\ & & 10^{-4} & 10^{-5} \\ & & & 10^{-6} \end{pmatrix}. \tag{9.5.25}$$

The QR algorithm as described tries to converge to the singular values from smallest to largest, and "chases the bulge" from top to bottom. Convergence will then be fast. However, if B is graded the opposite way then the QR algorithm may require many more steps. In this case the rows and columns of B could be reversed before the QR algorithm is applied. Many algorithms check for the direction of grading. Note that the matrix may break up into diagonal blocks which are graded in different ways.

The QR–SVD algorithm is designed for computing all singular values and possibly also the corresponding singular vectors of a matrix. In some applications, like the TLS problem, only the singular subspace associated with the smallest singular values are needed. A QR–SVD algorithm, modified to be more efficient for this case and called the PSVD algorithm is given by Van Huffel and Vandewalle [370, Chap. 4].

9.5.4 A Divide and Conquer Algorithm

The QR algorithm is one of the most elegant and efficient algorithms in linear algebra. However, it is basically a sequential algorithm and does not lend itself well to parallelization. For the symmetric tridiagonal eigenproblem and the bidiagonal SVD there are several alternative algorithms, which are faster and in some situations also more accurate.

A divide and conquer algorithm for the symmetric tridiagonal case was first suggested by Cuppen [81] and later modified by Dongarra and Sorensen [112] and Gu and Eisenstat [190].

The basic idea in the divide and conquer algorithm for the symmetric tridiagonal eigenproblem is to divide the tridiagonal matrix $T \in \mathbf{R}^{n\times n}$ into two smaller symmetric tridiagonal matrices T_1 and T_2 as follows:

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix} = \begin{pmatrix} T_1 & \beta_k e_{k-1} & 0 \\ \beta_k e_{k-1}^T & \alpha_k & \beta_{k+1} e_1^T \\ 0 & \beta_{k+1} e_1 & T_2 \end{pmatrix} \tag{9.5.26}$$

Here e_j is the jth unit vector of appropriate dimension and T_1 and T_2 are $(k-1) \times (k-1)$ and $(n-k) \times (n-k)$ symmetric tridiagonal principle submatrices of T.

Suppose now that the eigen-decompositions of $T_i = Q_i D_i Q_i^T$, $i = 1, 2$ are known. Permuting row and column k to the first position and substituting into (9.5.26) we get

$$T = \begin{pmatrix} \alpha_k & \beta_k e_{k-1}^T & \beta_{k+1} e_1^T \\ \beta_k e_{k-1} & Q_1 D_1 Q_1^T & 0 \\ \beta_{k+1} e_1 & 0 & Q_2 D_2 Q_2^T \end{pmatrix} = QHQ^T, \tag{9.5.27}$$

where

$$H = \begin{pmatrix} \alpha_k & \beta_k l_1^T & \beta_{k+1} f_2^T \\ \beta_k l_1 & D_1 & 0 \\ \beta_{k+1} f_2 & 0 & D_2 \end{pmatrix}, \qquad Q = \begin{pmatrix} 0 & Q_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & Q_2 \end{pmatrix}.$$

and $l_1 = Q_1^T e_{k-1}$ is the last row of Q_1 and $f_2 = Q_2^T e_1$ is the first row of Q_2. Hence, the matrix T is reduced to the bordered diagonal form

$$H = \begin{pmatrix} z_1 & z_2 & \cdots & z_n \\ z_2 & \alpha_2 & & \\ \vdots & & \ddots & \\ z_n & & & \alpha_n \end{pmatrix} \tag{9.5.28}$$

by the orthogonal similarity transformation Q. The matrix H is also called a **symmetric arrowhead matrix**. If $H = U\Lambda U^T$ is the the spectral decomposition of H, then the spectral decomposition of T equals

$$T = (QU)\Lambda(QU)^T. \tag{9.5.29}$$

To compute the eigensystems of T_1 and T_2, the splitting in (9.5.26) can be applied recursively until the original tridiagonal matrix T has been reduced to a desired number of small subproblems. Then the above relations may be applied from the bottom up to glue the eigensystems together.

We now give an algorithm for computing the eigenvalues and eigenvectors for the symmetric arrowhead matrix H. This problem is discussed in detail in Wilkinson [387, pp. 95–96]. It is no restriction to assume that $d_2 \le d_3 \le \cdots \le d_n$, since this can be achieved by a symmetric permutation. We make the following observations:

- If $z_i j = 0$, then one eigenvalue equals d_j, and the degree of the secular equation is decreased by one.

- If $d_j = d_{j+1}$ for some j, $2 \le j \le n-1$, then it can be shown that one eigenvalue of H equals d_i, and again the degree of the secular equation may be reduced by one.

We illustrate these observations for the 3×3 case. Suppose that $z_2 = 0$ and permute rows and columns 2 and 3. Then

$$P_{23}\begin{pmatrix} z_1 & 0 & z_3 \\ 0 & d_2 & 0 \\ z_3 & 0 & d_3 \end{pmatrix} P_{23} = \begin{pmatrix} z_1 & z_3 & 0 \\ z_3 & d_3 & 0 \\ 0 & 0 & d_2 \end{pmatrix} = \begin{pmatrix} H' & 0 \\ 0 & d_2 \end{pmatrix}.$$

Clearly d_2 is an eigenvalue and we can work with the deflated matrix H'.

To illustrate the second case, assume that $d_2 = d_3$. Then we can apply Givens transformations from left and right to zero out the element z_3. Since $G_{23}d_2IG_{23}^T = d_2G_{23}G_{23}^T = d_2I$, we obtain

$$G_{23}\begin{pmatrix} z_1 & z_2 & z_3 \\ z_2 & d_2 & 0 \\ z_3 & 0 & d_2 \end{pmatrix} G_{23}^T = \begin{pmatrix} z_1 & z_2' & 0 \\ z_2' & d_2 & 0 \\ 0 & 0 & d_2 \end{pmatrix} = \begin{pmatrix} H' & 0 \\ 0 & d_2 \end{pmatrix}.$$

Again d_2 is an eigenvalue and the the problem deflates.

Therefore, we can make the assumption that the elements d_i are distinct and the elements z_j are nonzero. In practice these assumptions above must be replaced by

$$d_{j+1} - d_j \ge \tau\|H\|_2, \qquad |z_j| \ge \tau\|H\|_2, \quad j = 2:n,$$

where τ is a small multiple of the unit roundoff.

Expanding $\det(H-\lambda I)$ along the first row (see Sec. 7.1.3) the characteristic polynomial of H is

$$\det(H-\lambda I) = (z_1-\lambda)\prod_{i=2}^{n}(d_i-\lambda) - \sum_{j=2}^{n} z_i^2 \prod_{i\ne j}^{n}(d_i-\lambda).$$

If λ is an eigenvalue the corresponding eigenvector satisfies the linear system $(H - \lambda_i I)x = 0$. Setting $x_1 = -1$, the remaining components satisfy

$$-z_i + (d_i - \lambda)x_i = 0, \quad i = 2:n.$$

Thus, we find the following characterization of the eigenvalues and eigenvectors:

Lemma 9.5.2.

The eigenvalues of the arrowhead matrix H satisfy the interlacing property

$$\lambda_1 \le d_2 \le \lambda_2 \le \cdots \le d_n \le \lambda_n,$$

and the secular equation

$$\phi(\lambda) = \lambda - z_1 + \sum_{j=2}^{n} \frac{z_j^2}{d_j - \lambda} = 0. \tag{9.5.30}$$

For each eigenvalue λ_i of H, a corresponding normalized eigenvector is $u_i = \tilde{u}_i/\|\tilde{u}_i\|_2$, where

$$u_i = \left(-1, \frac{z_2}{d_2 - \lambda_i}, \ldots, \frac{z_n}{d_n - \lambda_i}\right)^T, \quad \|\tilde{u}_i\|_2^2 = \left(1 + \sum_{j=2}^{n} \frac{z_j^2}{(d_j - \lambda_i)^2}\right). \quad (9.5.31)$$

The the roots of the secular equation are simple and isolated in an interval (d_i, d_{i+1}) where $f(\lambda)$ is monotonic and smooth. Although Newton's method could be used to find the roots it may not be suitable, since the function f has poles at $d_2, \ldots, d_n$. A zero finder based on rational osculatory interpolation with guaranteed quadratic convergence was developed by Bunch et al. [53]. Even this can sometimes require too many iterations and an improved algorithm of Ren-Cang Li [264] is to be preferred.

The main work in the updating is to form the matrix product $X = QU$ in (9.5.29). Since Q is essentially a block 2×2 matrix of order n the work in forming X is approximately n^3 flops. As in recursive Cholesky factorization (see Sec. 7.3.2) at next lower level we have two subproblems which each takes $1/2^3$ as much work, so the number of flops roughly reduced by a factor of four for each stage. Thus, the total work in the divide and conquer method equals

$$n^3\left(1 + 1/4 + 1/4^2 + \cdots\right) = n^3/(1 - 1/4) = 4n^3/3 \text{ flops}.$$

Also, these flops are spent in matrix multiplication and can use BLAS 3 subroutines. What if only eigenvalues are wanted?

While the eigenvalues are always well conditioned with respect to small perturbations, the eigenvectors can be extremely sensitive in the presence of close eigenvalues. Then computing the eigenvectors using (9.5.31) will not give eigenvectors which are accurately orthogonal unless extended precision is used.

In practice the formula for the eigenvectors in Lemma 9.5.2 cannot be used directly. The reason for this is that we can only compute an approximation $\hat{\lambda}_i$ to λ_i. Even if $\hat{\lambda}_i$ is very close to λ_i, the approximate ratio $z_j/(d_j - \hat{\lambda}_i)$ can be very different from the corresponding exact ratio. These errors may lead to computed eigenvectors of T which are numerically not orthogonal. An ingenious solution to this problem has been found, which involves modifying the vector z rather than increasing the accuracy of the $\hat{\lambda}_i$; see Gu and Eisenstat [190, 1975]. The resulting algorithm seems to outperform the QR algorithm even on single processor computers.

Because of the overhead in the divide and conquer algorithm it is less efficient than the QR algorithm for matrices of small dimension. A suitable value to switch has been found to be $n = 25$.

A Divide and Conquer Algorithm for the SVD

The computation of the bidiagonal singular value decomposition can also be speeded up by using a divide and conquer algorithm. Such an algorithm was given by Jessup and Sorensen [226] and later improved by Gu and Eisenstat [188]. Given the upper bidiagonal matrix $B \in \mathbf{R}^{n\times n}$ this is recursively divided into subproblems as follows:

$$B = \begin{pmatrix} q_1 & r_1 & & & \\ & q_2 & r_2 & & \\ & & \ddots & \ddots & \\ & & & q_{n-1} & r_{n-1} \\ & & & & q_n \end{pmatrix} = \begin{pmatrix} B_1 & 0 \\ q_k e_k^T & r_k e_1^T \\ 0 & B_2 \end{pmatrix} \tag{9.5.32}$$

where $B_1 \in \mathbf{R}^{n_1\times(n_1+1)}$ and $B_2 \in \mathbf{R}^{n_2\times n_2}$, $k = n_1+1$, and $n_1+n_2 = n-1$. Given the SVDs of B_1 and B_2,

$$B_1 = Q_1 \begin{pmatrix} D_1 & 0 \end{pmatrix} W_1^T, \qquad B_2 = Q_2 D_2 W_2^T,$$

and substituting into (9.5.32) we get

$$B = \begin{pmatrix} Q_1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & Q_2 \end{pmatrix} \begin{pmatrix} D_1 & 0 & 0 \\ q_k l_1^T & q_k\lambda_1 & r_k f_2^T \\ 0 & 0 & D_2 \end{pmatrix} \begin{pmatrix} W_1 & 0 \\ 0 & W_2 \end{pmatrix}^T, \tag{9.5.33}$$

where $\begin{pmatrix} l_1^T & \lambda_1 \end{pmatrix} = e_k^T W_1$ is the last row of W_1 and $f_2^T = e_1^T W_2$ is the first row of W_2.

After a symmetric permutation of block rows 1 and 2 in the middle matrix it has the form

$$M = \begin{pmatrix} q_k\lambda_1 & q_k l_1^T & r_k f_2^T \\ 0 & D_1 & 0 \\ 0 & 0 & D_2 \end{pmatrix},$$

If the SVD of this matrix is $M = X\Sigma Y^T$, then the SVD of B is

$$B = Q\Sigma W^T, \quad Q = \begin{pmatrix} 0 & Q_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & Q_2 \end{pmatrix} X, \quad W = \begin{pmatrix} W_1 P & 0 \\ 0 & W_2 \end{pmatrix} Y, \tag{9.5.34}$$

where P is the permutation matrix that permutes the last column into first position.

The middle matrix in (9.5.33) has the form

$$M = \begin{pmatrix} z_1 & z_2 & \cdots & z_n \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix} = D + e_1 z^T, \tag{9.5.35}$$

where $D = \operatorname{diag}(d_1, d_2, \ldots, d_n)$, $d_1 \equiv 0$, contains the elements in D_1 and D_2. Here d_1 is introduced to simplify the presentation. We further assume that

$$0 = d_1 \le d_2 \le d_3 \le \ldots \le d_n,$$

which can be achieved by a row and column permutation.

We note that:

- If $z_i = 0$, then d_i is a singular value of M and the degree of the secular equation may be reduced by one.
- If $d_i = d_{i+1}$ for some i, $2 \le i \le n-1$, then d_i is a singular value of M and the degree of the secular equation may be reduced by one.

We can therefore assume that $|z_i| \neq 0$, $i = 1 : n$, and that $d_i \neq d_{i+1}$, $i = 1 : n-1$. In practice the assumptions above must be replaced by

$$d_{j+1} - d_j \ge \tau \|M\|_2, \qquad |z_j| \ge \tau \|M\|_2,$$

where τ is a small multiple of the unit roundoff.

In order to compute the SVD $M = D + e_1 z^T = X\Sigma Y^T$ we use the fact that the square of the singular values Σ^2 are the eigenvalues and the right singular vectors Y the eigenvectors of the matrix

$$M^T M = X\Sigma^2 X^T = D^2 + z e_1^T e_1 z^T = D^2 + zz^T.$$

This matrix has the same form as in Theorem 9.2.10 with $\mu = 1$. Further, $My_i = \sigma_i x_i$, which shows that so if y_i is a right singular vector then My_i is a vector in the direction of the corresponding left singular vector. This leads to the following characterization of the singular values and vectors of M.

Lemma 9.5.3.

Let the SVD of the matrix in (9.5.35) *be* $M = X\Sigma Y^T$ *with*

$$X = (x_1, \ldots, x_n), \quad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n), \quad Y = (y_1, \ldots, y_n).$$

Then the singular value satisfy the interlacing property

$$0 = d_1 < \sigma_1 < d_2 < \sigma_2 < \cdots < d_n < \sigma_n < d_n + \|z\|_2$$

where $z = (z_1, \ldots, z_n)^T$. *The secular equation is*

$$f(\sigma) = 1 + \sum_{k=1}^{n} \frac{z_k^2}{d_k^2 - \sigma^2} = 0.$$

The singular vectors are $x_i = \tilde{x}_i / \|\tilde{x}_i\|_2$, $y_i = \tilde{y}_i / \|\tilde{y}_i\|_2$, $i = 1 : n$, *where*

$$\tilde{y}_i = \left(\frac{z_1}{d_1^2 - \sigma_i^2}, \ldots, \frac{z_n}{d_n^2 - \sigma_i^2} \right), \quad \tilde{x}_i = \left(-1, \ \frac{d_2 z_2}{d_2^2 - \sigma_i^2}, \ldots, \frac{d_n z_n}{d_n^2 - \sigma_i^2} \right). \tag{9.5.36}$$

and

$$\|\tilde{y}_i\|_2^2 = \left(\sum_{j=1}^{n} \frac{z_j^2}{(d_j^2 - \sigma_i^2)^2} \right), \quad \|\tilde{x}_i\|_2^2 = \left(1 + \sum_{j=2}^{n} \frac{(d_j z_j)^2}{(d_j^2 - \sigma_i^2)^2} \right). \tag{9.5.37}$$

In the divide and conquer algorithm for computing the SVD of B this process is recursively applied to B_2 and B_2, until the sizes of the subproblems are sufficiently small. This requires at most $\log_2 n$ steps. The process has to be modified slightly since, unlike B, B_1 is not a square matrix.

The secular equation can be solved efficiently and accurately by the algorithm of Ren-Cang Li. The singular values of M are always well conditioned with respect to small perturbations, but the singular vectors can be extremely sensitive in the presence of close singular values. To get singular vectors which are accurately orthogonal without using extended precision a similar approach as used for obtaining orthogonal eigenvectors can be used; see Gu and Eisenstat [189].

9.5.5 Spectrum Slicing

Sylvester's law of inertia (see Theorem 7.3.8) leads to a simple and important method called **spectrum slicing** for counting the eigenvalues greater than a given real number τ of a Hermitian matrix A. In the following we treat the real symmetric case, but everything goes through also for general Hermitian matrices. The following theorem is a direct consequence of Sylvester's Law of Inertia.

Theorem 9.5.4.

Assume that symmetric Gaussian elimination can be carried through for $A-\tau I$ yielding the factorization (cf. (7.3.2)*)*

$$A - \tau I = LDL^T, \quad D = \operatorname{diag}(d_1, \dots, d_n), \tag{9.5.38}$$

where L is a unit lower triangular matrix. Then $A - \tau I$ is congruent to D, and hence the number of eigenvalues of A greater than τ equals the number of positive elements $\pi(D)$ in the sequence $d_1, \dots, d_n$.

Example 9.5.2.

The LDL^T factorization

$$A - 1 \cdot I = \begin{pmatrix} 1 & 2 & \\ 2 & 2 & -4 \\ & -4 & -6 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -2 & \\ & & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 & \\ & 1 & 2 \\ & & 1 \end{pmatrix}.$$

shows that the matrix A has two eigenvalues greater than 1.

The LDL^T factorization may fail to exist if $A-\tau I$ is not positive definite. This will happen for example if we choose the shift $\tau = 2$ for the matrix in Example 9.5.2. Then $a_{11} - \tau = 0$, and the first step in the factorization cannot be carried out. A closer analysis shows that the factorization will fail if, and only if, τ equals an eigenvalue to one or more of the $n - 1$ leading principal submatrices of A. If τ is chosen in a small interval around each of these values, big growth of elements occurs and the factorization may give the wrong count. In such cases one should perturb τ by a small amount (**how small??**) and restart the factorization from the beginning.

For the special case when A is a symmetric tridiagonal matrix the procedure outlined above becomes particularly efficient and reliable. Here the factorization is

$T - \tau I = LDL^T$, where L is unit lower bidiagonal and $D = \operatorname{diag}(d_1, \dots, d_n)$. The remarkable fact is that if we only take care to avoid over/underflow then *element growth will not affect the accuracy of the slice.*

Algorithm 9.2. Tridiagonal Spectrum Slicing.

Let T be the tridiagonal matrix (9.5.1). Then the number π of eigenvalues greater than a given number τ is generated by the following algorithm:

$d_1 := \alpha_1 - \tau;$
$\pi :=$ **if** $d_1 > 0$ **then** 1 **else** $0;$
for $k = 2 : n$
    $d_k := (\alpha_k - \beta_k(\beta_k/d_{k-1})) - \tau;$
    if $|d_k| < \sqrt{\omega}$ **then** $d_k := \sqrt{\omega};$
    if $d_k > 0$ **then** $\pi := \pi + 1;$
end

Here, to prevent breakdown of the recursion, a small $|d_k|$ is replaced by $\sqrt{\omega}$ where ω is the underflow threshold. The recursion uses only $2n$ flops, and it is not necessary to store the elements d_k. The number of multiplications can be halved by computing initially β_k^2, which however may cause unnecessary over/underflow. Assuming that no over/underflow occurs Algorithm 9.5.5 is backward stable. A round-off error analysis shows that the computed values $\bar{d}_k$ satisfy exactly

$$\begin{aligned}\bar{d}_k &= fl\big((\alpha_k - \beta_k(\beta_k/\bar{d}_{k-1})) - \tau\big)\\ &= \Big(\Big(\alpha_k - \frac{\beta_k^2}{\bar{d}_{k-1}}(1+\epsilon_{1k})(1+\epsilon_{2k})\Big)(1+\epsilon_{3k}) - \tau\Big)(1+\epsilon_{4k}) \qquad (9.5.39)\\ &\equiv \alpha_k' - \tau - (\beta_k')^2/\bar{d}_{k-1}, \qquad k = 1:n,\end{aligned}$$

where $\beta_1 = 0$ and $|\epsilon_{ik}| \le u$. Hence, the computed number $\bar{\pi}$ is the exact number of eigenvalues greater than τ of a matrix A', where A' has elements satisfying

$$|\alpha_k' - \alpha_k| \le u(2|\alpha_k| + |\tau|), \qquad |\beta_k' - \beta_k| \le 2u|\beta_k|. \qquad (9.5.40)$$

This is a very satisfactory backward error bound. It has been improved even further by Kahan [229, 1966], who shows that the term $2u|\alpha_k|$ in the bound can be dropped, see also Problem 1. Hence, it follows that eigenvalues found by bisection differ by a factor at most $(1 \pm u)$ from the exact eigenvalues of a matrix where only the off-diagonal elements are subject to a relative perturbed of at most $2u$. This is obviously a very satisfactory result.

The above technique can be used to locate any individual eigenvalue λ_k of A. Assume we have two values τ_l and τ_u such that for the corresponding diagonal factors we have

$$\pi(D_l) \ge k, \qquad \pi(D_u) < k$$

so that λ_k lies in the interval $[\tau_l, \tau_u)$. We can then using p steps of the bisection (or multisection) method (see Volume I, Sec. 6.1.1) locate λ_k in an interval of length $(\tau_u - \tau_l)/2^p$. From Gerschgorin's theorem it follows that all the eigenvalues of a tridiagonal matrix are contained in the union of the intervals

$$\alpha_i \pm (|\beta_i| + |\beta_{i+1}|), \quad i = 1 : n$$

where $\beta_1 = \beta_{n+1} = 0$.

Using the bound (9.2.25) it follows that the bisection error in each computed eigenvalue is bounded by $|\bar{\lambda}_j - \lambda_j| \leq \|A' - A\|_2$, where from (9.3.11), using the improved bound by Kahan, and the inequalities $|\tau| \leq \|A\|_2$, $|\alpha_k| \leq \|A\|_2$ it follows that

$$|\bar{\lambda}_j - \lambda_j| \leq 5u\|A\|_2. \tag{9.5.41}$$

This shows that the absolute error in the computed eigenvalues is always small. If some $|\lambda_k|$ is small it may be computed with poor *relative* precision. In some special cases (for example, tridiagonal, graded matrices see Sec. 9.5.1) even very small eigenvalues are determined to high relative precision by the elements in the matrix.

If many eigenvalues of a general real symmetric matrix A are to be determined by spectrum slicing, then A should initially be reduced to tridiagonal form. However, if A is a banded matrix and only few eigenvalues are to be determined then the Band Cholesky Algorithm 6.4.6 can be used to slice the spectrum. It is then necessary to monitor the element growth in the factorization. We finally mention that the technique of spectrum slicing is also applicable to the computation of selected singular values of a matrix and to the generalized eigenvalue problem

$$Ax = \lambda Bx,$$

where A and B are symmetric and B or A positive definite, see Sec. 9.8.

Singular Values by Spectrum Slicing

We proceed by first forming the symmetric matrix

$$C = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix} \in \mathbf{R}^{2n \times 2n}. \tag{9.5.42}$$

whose eigenvalues are $\pm\sigma_i$, $i = 1 : n$. After a symmetric reordering of rows and columns in C we obtain the symmetric tridiagonal matrix with zeros on the main diagonal

$$T = P^T C P = \begin{pmatrix} 0 & q_1 & & & & \\ q_1 & 0 & r_2 & & & \\ & r_2 & 0 & q_2 & & \\ & & q_2 & 0 & \ddots & \\ & & & \ddots & \ddots & q_n \\ & & & & q_n & 0 \end{pmatrix} \tag{9.5.43}$$

Here P is the permutation matrix whose columns are those of the identity in the order $(n+1, 1, n+2, 2, \ldots, 2n, n)$. Hence, the QR algorithm, the divide and conquer algorithm, and spectrum slicing (see Sec. 9.5.5) are all applicable to this special tridiagonal matrix to compute the singular values of B. A disadvantage of this approach is that the dimension is essentially doubled.

An algorithm for computing singular values can be developed by applying Algorithm 9.5.5 for spectrum slicing to the special symmetric tridiagonal matrix T in (9.5.43). Taking advantage of the zero diagonal this algorithm simplifies and one slice requires only of the order $2n$ flops. Given the elements $q_1, \ldots, q_n$ and $r_2, \ldots, r_n$ of T in (9.5.43), the following algorithm generates the number π of singular values of T greater than a given value $\sigma > 0$.

Algorithm 9.3. *Singular Values by Spectrum Slicing.*

Let T be the tridiagonal matrix (9.5.1). Then the number π of eigenvalues greater than a given number σ is generated by the following algorithm:

$$
\begin{array}{l}
d_1 := -\sigma;\\
flip := -1;\\
\pi := \textbf{if } d_1 > 0 \textbf{ then } 1 \textbf{ else } 0;\\
\textbf{for } k = 2 : 2n\\
flip := -flip;\\
\textbf{if } flip = 1 \textbf{ then } \beta = q_{k/2}\\
\qquad \textbf{else } \beta = r_{(k+1)/2};\\
\textbf{end}\\
\qquad d_k := -\beta(\beta/d_{k-1}) - \tau;\\
\qquad \textbf{if } |d_k| < \sqrt{\omega} \textbf{ then } d_k := \sqrt{\omega};\\
\qquad \textbf{if } d_k > 0 \textbf{ then } \pi := \pi + 1;\\
\textbf{end}
\end{array}
$$

Spectrum slicing algorithm for computing singular values has been analyzed by Fernando [127]. and shown to provide high relative accuracy also for tiny singular values.

9.5.6 Jacobi Methods

One of the oldest methods for solving the eigenvalue problem for real symmetric (or Hermitian) matrices is **Jacobi's**[41]**method**. Because it is at least three times

[41]Carl Gustaf Jacob Jacobi (1805–1851), German mathematician. Jacobi joined the faculty of Berlin university in 1825. Like Euler, he was a profilic calculator, who drew a great deal of insight from immense algorithmical work. His method for computing eigenvalues was published in 1846; see [225].

slower than the QR algorithm it fell out of favor for a period. However, Jacobi's method is easily parallelized and sometimes more accurate; see [97].

There are special situations when Jacobi's method is very efficient and should be preferred. For example, when the matrix is nearly diagonal or when one has to solve eigenvalue problems for a sequence of matrices, differing only slightly from each other. Jacobi's method, with a proper stopping criterion, can be shown to compute *all eigenvalues of symmetric positive definite matrices with uniformly better relative accuracy, than any algorithms which first reduces the matrix to tridiagonal form.* Note that, although the QR algorithm is backward stable (see Section 9.4), high relative accuracy can only be guaranteed for the larger eigenvalues (those near $\|A\|$ in magnitude).

The Jacobi method solves the eigenvalue problem for $A \in \mathbf{R}^{n\times n}$ by employing a sequence of similarity transformations

$$A_0 = A, \qquad A_{k+1} = J_k^T A_k J_k \tag{9.5.44}$$

such that the sequence of matrices A_k, $k = 1, 2, ...$ tends to a diagonal form. For each k, J_k is chosen as a plane rotations $J_k = G_{pq}(\theta)$, defined by a pair of indices (p, q), $p < q$, called the pivot pair. The angle θ is chosen so that the off-diagonal elements $a_{pq} = a_{qp}$ are reduced to zero, i.e. by solving a 2×2 subproblems. We note that only the entries in rows and columns p and q of A will change, and since symmetry is preserved only the upper triangular part of each A needs to be computed.

To construct the Jacobi transformation J_k we consider the symmetric 2×2 eigenvalue problem for the principal submatrix A_{pq} formed by rows and columns p and q. For simplicity of notation we rename $A_{k+1} = A'$ and $A_k = A$. Hence, we want to determine $c = \cos\theta$, $s = \sin\theta$ so that

$$\begin{pmatrix} l_p & 0 \\ 0 & l_q \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}. \tag{9.5.45}$$

Equating the off-diagonal elements we obtain (as $a_{pq} = a_{qp}$)

$$0 = (a_{pp} - a_{qq})cs + a_{pq}(c^2 - s^2), \tag{9.5.46}$$

which shows that the angle θ satisfies

$$\tau \equiv \cot 2\theta = (a_{qq} - a_{pp})/(2a_{pq}), \qquad a_{pq} \neq 0. \tag{9.5.47}$$

The two diagonal elements a_{pp} and a_{qq} are transformed as follows,

$$\begin{aligned} a'_{pp} &= c^2 a_{pp} - 2cs a_{pq} + s^2 a_{qq} = a_{pp} - t a_{pq}, \\ a'_{qq} &= s^2 a_{pp} + 2cs a_{pq} + c^2 a_{qq} = a_{qq} + t a_{pq}. \end{aligned}$$

where $t = \tan\theta$. We call this a **Jacobi transformation**. The following stopping criterion should be used:

$$\text{if } |a_{ij}| \leq \text{tol}\,(a_{ii}a_{jj})^{1/2}, \text{set } a_{ij} = 0, \tag{9.5.48}$$

where tol is the relative accuracy desired.

A stable way to perform a Jacobi transformation is to first compute $t = \tan\theta$ as the root of smallest modulus to the quadratic equation $t^2+2\tau t-1 = 0$. This choice ensures that $|\theta| < \pi/4$, and can be shown to minimize the difference $\|A' - A\|_F$. In particular, this will prevent the exchange of the two diagonal elements a_{pp} and a_{qq}, when a_{pq} is small, which is critical for the convergence of the Jacobi method. The transformation (9.5.45) is best computed by the following algorithm.

Algorithm 9.4.
Jacobi transformation matrix ($a_{pq} \neq 0$):

$$
\begin{aligned}
&[c, s, l_p, l_q] = \text{jacobi}(a_{pp}, a_{pq}, a_{qq})\\
&\quad \tau = (a_{qq} - a_{pp})/(2a_{pq});\\
&\quad t = \text{sign}\,(\tau)/(|\tau| + \sqrt{1+\tau^2});\\
&\quad c = 1/\sqrt{1+t^2}; \quad s = t\cdot c;\\
&\quad l_p = a_{pp} - t a_{pq};\\
&\quad l_q = a_{qq} + t a_{pq};\\
&\quad \text{end}
\end{aligned}
$$

The computed transformation is applied also to the remaining elements in rows and columns p and q of the full matrix A. These are transformed for $j \neq p, q$ according to

$$
\begin{aligned}
a'_{jp} = a'_{pj} &= c a_{pj} - s a_{qj} = a_{pj} - s(a_{qj} + r a_{pj}),\\
a'_{jq} = a'_{qj} &= s a_{pj} + c a_{qj} = a_{qj} + s(a_{pj} - r a_{qj}).
\end{aligned}
$$

where $r = s/(1+c) = \tan(\theta/2)$. (The formulas are written in a form, due to Rutishauser [326, 1971], which reduces roundoff errors.)

If symmetry is exploited, then one Jacobi transformation takes about $4n$ flops. Note that an off-diagonal element made zero at one step will in general become nonzero at some later stage. The Jacobi method will also destroy the band structure if A is a banded matrix.

The convergence of the Jacobi method depends on the fact that in each step the quantity

$$S(A) = \sum_{i\neq j} a_{ij}^2 = \|A - D\|_F^2,$$

i.e., the Frobenius norm of the off-diagonal elements is reduced. To see this, we note that the Frobenius norm of a matrix is invariant under multiplication from left or right with an orthogonal matrix. Therefore, since $a'_{pq} = 0$ we have

$$(a'_{pp})^2 + (a'_{qq})^2 = a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2.$$

We also have that $\|A'\|_F^2 = \|A\|_F^2$, and it follows that

$$S(A') = \|A'\|_F^2 - \sum_{i=1}^{n} (a'_{ii})^2 = S(A) - 2a_{pq}^2.$$

There are various strategies for choosing the order in which the off-diagonal elements are annihilated. Since $S(A')$ is reduced by $2a_{pq}^2$, the optimal choice is to annihilate the off-diagonal element of largest magnitude. This is done in the **classical Jacobi** method. Then since

$$2a_{pq}^2 \geq S(A_k)/N, \qquad N = n(n-1)/2,$$

we have $S(A_{k+1}) \leq (1-1/N)S(A_k)$. This shows that for the classical Jacobi method A_{k+1} converges at least linearly with rate $(1-1/N)$ to a diagonal matrix. In fact it has been shown that ultimately the rate of convergence is quadratic, so that for k large enough, we have $S(A_{k+1}) < cS(A_k)^2$ for some constant c. The iterations are repeated until $S(A_k) < \delta\|A\|_F$, where δ is a tolerance, which can be chosen equal to the unit roundoff u. From the Bauer–Fike Theorem 9.2.4 it then follows that the diagonal elements of A_k then approximate the eigenvalues of A with an error less than $\delta\|A\|_F$.

In the Classical Jacobi method a large amount of effort must be spent on searching for the largest off-diagonal element. Even though it is possible to reduce this time by taking advantage of the fact that only two rows and columns are changed at each step, the Classical Jacobi method is almost never used. In a **cyclic Jacobi method**, the $N = \frac{1}{2}n(n-1)$ off-diagonal elements are instead annihilated in some predetermined order, each element being rotated exactly once in any sequence of N rotations called a **sweep**. Convergence of any cyclic Jacobi method can be guaranteed if any rotation (p,q) is omitted for which $|a_{pq}|$ is smaller than some **threshold**; see Forsythe and Henrici [138, 1960]. To ensure a good rate of convergence this threshold tolerance should be successively decreased after each sweep.

For sequential computers the most popular cyclic ordering is the row-wise scheme, i.e., the rotations are performed in the order

$$\begin{array}{cccc} (1,2), & (1,3), & \dots & (1,n) \\ & (2,3), & \dots & (2,n) \\ & & \dots & \dots \\ & & & (n-1,n) \end{array} \tag{9.5.49}$$

which is cyclically repeated. About $2n^3$ flops per sweep is required. In practice, with the cyclic Jacobi method not more than about 5 sweeps are needed to obtain eigenvalues of more than single precision accuracy even when n is large. The number of sweeps grows approximately as $O(\log n)$, and about $10n^3$ flops are needed to compute all the eigenvalues of A. This is about 3–5 times more than for the QR algorithm.

An orthogonal system of eigenvectors of A can easily be obtained in the Jacobi method by computing the product of all the transformations

$$X_k = J_1 J_2 \cdots J_k.$$

Then $\lim_{k\to\infty} X_k = X$. If we put $X_0 = I$, then we recursively compute

$$X_k = X_{k-1} J_k, \quad k = 1, 2, \dots \tag{9.5.50}$$

In each transformation the two columns (p,q) of $X_{k-1}$is rotated, which requires $4n$ flop. Hence, in each sweep an additional $2n$ flops is needed, which doubles the operation count for the method.

The Jacobi method is very suitable for parallel computation since several noninteracting rotations, (p_i,q_i) and (p_j,q_j), where p_i,q_i are distinct from p_j,q_j, can be performed simultaneously. If n is even the $n/2$ Jacobi transformations can be performed simultaneously. A sweep needs at least $n-1$ such parallel steps. Several parallel schemes which uses this minimum number of steps have been constructed. These can be illustrated in the $n=8$ case by

$$(p,q) = \begin{array}{llll} (1,2), & (3,4), & (5,6), & (7,8) \\ (1,4), & (2,6), & (3,8), & (5,7) \\ (1,6), & (4,8), & (2,7), & (3,5) \\ (1,8), & (6,7), & (4,5), & (2,3) \\ (1,7), & (8,5), & (6,3), & (4,2) \\ (1,5), & (7,3), & (8,2), & (6,4) \\ (1,3), & (5,2), & (7,4), & (8,6) \end{array}.$$

The rotations associated with *each row* of the above can be calculated simultaneously. First the transformations are constructed in parallel; then the transformations from the left are applied in parallel, and finally the transformations from the right.

Several Jacobi-type methods for computing the SVD $A=U\Sigma V^T$ of a matrix were developed in the 1950's. The shortcomings of some of these algorithms have been removed, and as for the real symmetric eigenproblem, there are cases for which Jacobi's method is to be preferred over the QR-algorithm for the SVD. In particular, it computes the smaller singular values more accurately than any algorithm based on a preliminary bidiagonal reduction.

There are two different ways to generalize the Jacobi method for the SVD problem. We assume that $A\in\mathbf{R}^{n\times n}$ is a square nonsymmetric matrix. This is no restriction, since we can first compute the QR factorization

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}$$

and then apply the Jacobi SVD method to R. In the **two-sided Jacobi SVD** algorithm for the SVD of A (Kogbetliantz [242]) the elementary step consists of two-sided Givens transformations

$$A' = J_{pq}(\phi)AJ_{pq}^T(\psi), \tag{9.5.51}$$

where $J_{pq}(\phi)$ and $J_{pq}(\psi)$ are determined so that $a'_{pq}=a'_{qp}=0$. Note that only rows and columns p and q in A are affected by the transformation. The rotations $J_{pq}(\phi)$ and $J_{pq}(\psi)$ are determined by computing the SVD of a 2×2 submatrix

$$A = \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix}, \qquad a_{pp}\ge 0, \quad a_{qq}\ge 0.$$

The assumption of nonnegative diagonal elements is no restriction, since we can change the sign of these by pre-multiplication with an orthogonal matrix diag $(\pm 1,\pm 1)$.

Since the Frobenius norm is invariant under orthogonal transformations it follows that

$$S(A') = S(A) - (a_{pq}^2 + a_{qp}^2), \qquad S(A) = \|A - D\|_F^2.$$

This relation is the basis for a proof that the matrices generated by two-sided Jacobi method converge to a diagonal matrix containing the singular values of A. Orthogonal systems of left and right singular vectors can be obtained by accumulating the product of all the transformations.

At first a drawback of the above algorithm seems to be that it works all the time on a full $m \times n$ unsymmetric matrix. However, if a proper cyclic rotation strategy is used, then at each step the matrix will be essentially triangular. If the column cyclic strategy

$$(1,2), (1,3), (2,3), \ldots, (1,n), \ldots, (n-1,n)$$

is used an upper triangular matrix will be successively transformed into a lower triangular matrix. The next sweep will transform it back to an upper triangular matrix. During the whole process the matrix can be stored in an upper triangular array. The initial QR factorization also cures some global convergence problems present in the twosided Jacobi SVD method.

In the **one-sided Jacobi SVD** algorithm Givens transformations are used to find an orthogonal matrix V such that the matrix AV has orthogonal columns. Then $AV = U\Sigma$ and the SVD of A is readily obtained. The columns can be explicitly interchanged so that the final columns of AV appear in order of decreasing norm. The basic step rotates two columns:

$$(\hat{a}_p, \hat{a}_q) = (a_p, a_q) \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \qquad p < q. \tag{9.5.52}$$

The parameters c, s are determined so that the rotated columns are orthogonal, or equivalently so that

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} \|a_p\|_2^2 & a_p^T a_q \\ a_q^T a_p & \|a_q\|_2^2 \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}^T$$

is diagonal. This 2×2 symmetric eigenproblem can be solved by a Jacobi transformation, where the rotation angle is determined by (cf. (9.5.47))

$$\tau \equiv \cot 2\theta = (\|a_q\|_2^2 - \|a_p\|_2^2)/(2a_q^T a), \qquad a_q^T a_p \neq 0.$$

Alternatively, we can first compute the QR factorization

$$(a_p, a_q) = (q_1,\ q_2) \begin{pmatrix} r_{pp} & r_{pq} \\ 0 & r_{qq} \end{pmatrix} \equiv QR,$$

and then the 2×2 SVD $R = U\Sigma V^T$. then since $RV = U\Sigma$

$$(a_p, a_q)V = (q_1,\ q_2)U\Sigma$$

will have orthogonal columns. It follows that V is the desired rotation in (9.5.52).

Clearly, the one-sided algorithm is mathematically equivalent to applying Jacobi's method to diagonalize $C = A^T A$, and hence its convergence properties are the same. Convergence of Jacobi's method is related to the fact that in each step the sum of squares of the off-diagonal elements

$$S(C) = \sum_{i \neq j} c_{ij}^2, \qquad C = A^T A$$

is reduced. Hence, the rate of convergence is ultimately quadratic, also for multiple singular values. Note that the one-sided Jacobi SVD will by construction have U orthogonal to working accuracy, but loss of orthogonality in V may occur. Therefore, the columns of V should be reorthogonalized using a Gram–Schmidt process at the end.

The one-sided method can be applied to a general real (or complex) matrix $A \in \mathbf{R}^{m\times n}$, $m \geq n$, but an initial QR factorization should performed to speed up convergence. If this is performed with *row and column pivoting*, then high relative accuracy can be achieved for matrices A that are *diagonal scalings of a well-conditioned matrix*, that is which can be decomposed as

$$A = D_1 B D_2,$$

where $D_1,\ D_2$ are diagonal and B well-conditioned. It has been demonstrated that if pre-sorting the rows after decreasing norm $\|a_{i,:}\|_\infty$ and then using column pivoting only gives equally good results. By a careful choice of the rotation sequence the essential triangularity of the matrix can be preserved during the Jacobi iterations.

In a cyclic Jacobi method, the off-diagonal elements are annihilated in some predetermined order, each element being rotated exactly once in any sequence of $N = n(n-1)/2$ rotations called a **sweep**. Parallel implementations can take advantage of the fact that noninteracting rotations, (p_i, q_i) and (p_j, q_j), where p_i, q_i and p_j, q_j are distinct, can be performed simultaneously. If n is even $n/2$ transformations can be performed simultaneously, and a sweep needs at least $n-1$ such parallel steps. In practice, with the cyclic Jacobi method not more than about five sweeps are needed to obtain singular values of more than single precision accuracy even when n is large. The number of sweeps grows approximately as $O(\log n)$.

The alternative algorithm for the SVD of 2×2 upper triangular matrix below always gives high *relative accuracy* in the singular values and vectors, has been developed by Demmel and Kahan, and is based on the relations in Problem 5.

9.5.7 The QD Algorithm

(To be written.) Parlett [304], Dhillon [102], Dhillon and Parlett [105, 104], Fernando and Parlett [128], von Matt [377].

Review Questions

5.1 (a) Show that the symmetry of a matrix is preserved by the QR algorithm. What about normality?

5.2 For a certain class of symmetric matrices small eigenvalues are determined with a very small error compared to $\|A\|_F$. Which class?

5.3 What condensed form is usually chosen for the singular value decomposition? What kind of transformations are used for bringing the matrix to condensed form?

(b) Does the reduction in (a) apply to a complex matrix A?

5.4 What is the asymptotic speed of convergence for the classical Jacobi method? Discuss the advantages and drawbacks of Jacobi methods compared to the QR algorithm.

5.6 There are two different Jacobi-type methods for computing the SVD were developed. What are they called? What 2×2 subproblems are they based on?

5.7 (a) Describe the method of spectrum slicing for determining selected eigenvalues of a real symmetric matrix A.

(b) How is the method of spectrum slicing applied for computing singular values?

Problems

5.1 Perform a QR step without shift on the matrix

$$A = \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & 0 \end{pmatrix}$$

and show that the nondiagonal elements are reduced to $-\sin^3\theta$.

5.2 Reduce to tridiagonal form, using an exact orthogonal similarity, the real symmetric matrix

$$A = \begin{pmatrix} 1 & \sqrt{2} & \sqrt{2} & \sqrt{2} \\ \sqrt{2} & -\sqrt{2} & -1 & \sqrt{2} \\ \sqrt{2} & -1 & \sqrt{2} & \sqrt{2} \\ 2 & \sqrt{2} & \sqrt{2} & -3 \end{pmatrix}$$

5.3 Show that if a real skew-symmetric matrix K, $K^T = -K$, is reduced to Hessenberg form H by an orthogonal similarity, then H must be skew-symmetric and tridiagonal. Perform this reduction of the skew-symmetric circulant matrix K (see (9.1.26)) with first row equal to

$$(0, 1, 1, 0, -1, -1).$$

5.4 (a) Let $Q \in \mathbf{R}^{3\times 3}$ be an orthogonal matrix. Assume that $Q \neq I$, and $\det(Q) = +1$, so that Q represents a pure rotation. Show that Q has a real eigenvalue equal to $+1$, which corresponds to the screw axis of rotation. Show that the two other eigenvalues are of the form $\lambda = e^{\pm i\phi}$.

(b) Let

$$M = \tfrac{1}{2}(Q^T + Q), \qquad K = \tfrac{1}{2}(Q^T - Q),$$

be the symmetric and skew-symmetric part of Q. Show that M and K have the same eigenvectors as Q. What are their eigenvalues of M and K?

(c) Show that the eigenvector corresponding to the zero eigenvalue of

$$K = \begin{pmatrix} 0 & k_{12} & k_{13} \\ -k_{12} & 0 & k_{23} \\ -k_{13} & -k_{23} & 0, \end{pmatrix}$$

equals $u_1 = (k_{23}, -k_{13}, k_{12})^T$. Derive the characteristic equation $\det(K - \lambda I) = 0$ and conclude that the two remaining eigenvalues are $\pm i \sin\phi$, where

$$\sin^2\phi = k_{12}^2 + k_{13}^2 + k_{23}^2.$$

5.5 To compute the eigenvalues of the following real symmetric pentadiagonal matrix

$$A = \begin{pmatrix} 4 & 2 & 1 & 0 & 0 & 0 \\ 2 & 4 & 2 & 1 & 0 & 0 \\ 1 & 2 & 4 & 2 & 1 & 0 \\ 0 & 1 & 2 & 4 & 2 & 1 \\ 0 & 0 & 1 & 2 & 4 & 2 \\ 0 & 0 & 0 & 1 & 2 & 4 \end{pmatrix},$$

the matrix is first reduced to A tridiagonal form.

(a) Determine a Givens rotation G_{23} which zeros the element in position $(3,1)$. Compute the the transformed matrix $A^{(1)} = G_{23} A G_{23}^T$.

(b) In the matrix $A^{(1)}$ a new nonzero element has been introduced. Show how this can be zeroed by a new rotation without introducing any new nonzero elements.

(c) Device a "zero chasing" algorithm to reduce a general real symmetric pentadiagonal matrix $A \in \mathbf{R}^{n\times n}$ to symmetric tridiagonal form. How many rotations are needed? How many flops?

5.6 Let T be the tridiagonal matrix in (9.5.1), and suppose a QR step using the shift $\tau = \alpha_n$ is carried out,

$$T - \alpha_n I = QR, \qquad \tilde{T} = RQ + \alpha_n I.$$

Generalize the result from Problem 2, and show that if $\gamma = \min_i |\lambda_i(T_{n-1}) - \alpha_n| > 0$, then $|\tilde{\beta}_n| \leq |\beta_n|^3/\gamma^2$.

5.7 Let C be the matrix in (9.5.42) and P the permutation matrix whose columns are those of the identity matrix in the order $(n+1, 1, n+2, 2, \dots, 2n, n)$. Show that the matrix $P^T C P$ becomes a tridiagonal matrix T of the form in (9.5.43).

5.8 Modify Algorithm 9.7.1 for the zero shift QR–SVD algorithm so that the two loops are merged into one.

5.9 (a) Let σ_i be the singular values of the matrix

$$M = \begin{pmatrix} z_1 & & & \\ z_2 & d_2 & & \\ \vdots & & \ddots & \\ z_n & & & d_n \end{pmatrix} \in \mathbf{R}^{n\times n},$$

where the elements d_i are distinct. Show the interlacing property

$$0 < \sigma_1 < d_2 < \cdots < d_n < \sigma_n < d_n + \|z\|_2.$$

(b) Show that σ_i satisfies the secular equation

$$f(\sigma) = 1 + \sum_{k=1}^{n} \frac{z_k^2}{d_k^2 - \sigma^2} = 0.$$

Give expressions for the right and left singular vectors of M.

Hint: See Lemma 9.5.2.

5.10 Implement Jacobi's algorithm, using the stopping criterion (9.5.48) with tol $= 10^{-12}$. Use it to compute the eigenvalues of

$$A = \begin{pmatrix} -0.442 & -0.607 & -1.075 \\ -0.607 & 0.806 & 0.455 \\ -1.075 & 0.455 & -1.069 \end{pmatrix},$$

How many Jacobi steps are used?

5.11 Suppose the matrix

$$\tilde{A} = \begin{pmatrix} 1 & 10^{-2} & 10^{-4} \\ 10^{-2} & 2 & 10^{-2} \\ 10^{-4} & 10^{-2} & 4 \end{pmatrix}.$$

has been obtained at a certain step of the Jacobi algorithm. Estimate the eigenvalues of $\tilde{A}$ as accurately as possible using the Gerschgorin circles with a suitable diagonal transformation, see Problem 9.3.3.

5.12 Jacobi-type methods can also be constructed for Hermitian matrices using *elementary unitary rotations* of the form

$$U = \begin{pmatrix} \cos\theta & \alpha\sin\theta \\ -\bar{\alpha}\sin\theta & \cos\theta \end{pmatrix}, \qquad |\alpha| = 1.$$

Show that if we take $\alpha = a_{pq}/|a_{pq}|$ then equation (9.5.47) for the angle θ becomes

$$\tau = \cot 2\theta = (a_{pp} - a_{qq})/(2|a_{pq}|), \qquad |a_{pq}| \neq 0.$$

(Note that the diagonal elements a_{pp} and a_{qq} of a Hermitian matrix are real.)

5.13 Let $A \in \mathbf{C}^{2\times 2}$ be a given matrix, and U a unitary matrix of the form in Problem 3. Determine U so that the matrix $B = U^{-1}AU$ becomes upper triangular, that is, the Schur Canonical Form of A. Use this result to compute the eigenvalues of

$$A = \begin{pmatrix} 9 & 10 \\ -2 & 5 \end{pmatrix}.$$

Outline a Jacobi-type method to compute the Schur Canonical form of a general matrix A.

5.14 (a) Use one Givens rotation to transform the matrix

$$A = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix},$$

to tridiagonal form.

(b) Compute the largest eigenvalue of A, using spectrum slicing on the tridiagonal form derived in (a). Then compute the corresponding eigenvector.

5.15 Show that (9.5.38) can be written

$$\hat{d}_k = \alpha_k - \frac{\beta_k^2}{\hat{d}_{k-1}} \frac{(1+\epsilon_{1k})(1+\epsilon_{2k})}{(1+\epsilon_{3,k-1})(1+\epsilon_{4,k-1})} - \frac{\tau}{(1+\epsilon_{3k})}, \quad k = 1:n,$$

where we have put $\bar{d}_k = \hat{d}_k(1+\epsilon_{3k})(1+\epsilon_{4k})$, and $|\epsilon_{ik}| \le u$. Conclude that since $\operatorname{sign}(\hat{d}_k) = \operatorname{sign}(\bar{d}_k)$ the computed number $\bar{\pi}$ is the exact number of eigenvalues a tridiagonal matrix A' whose elements satisfy

$$|\alpha_k' - \alpha_k| \le u|\tau|, \quad |\beta_k' - \beta_k| \le 2u|\beta_k|.$$

5.16 To compute the SVD of a matrix $A \in \mathbf{R}^{m\times 2}$ we can first reduce A to upper triangular form by a QR decomposition

$$A = (a_1,\ a_2) = (q_1,\ q_2)\begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad R = \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}.$$

Then, as outlined in Golub and Van Loan [184, Problem 8.5.1], a Givens rotation G can be determined such that $B = GRG^T$ is symmetric. Finally, B can be diagonalized by a Jacobi transformation. Derive the details of this algorithm!

5.17 Show that if Kogbetliantz's method is applied to a triangular matrix then after one sweep of the row cyclic algorithm (9.5.49) an upper (lower) triangular matrix becomes lower (upper) triangular.

5.18 In the original divide and conquer algorithm by Cuppen [81] for symmetric tridiagonal matrices a different splitting is used than in Sec. 9.5.4. The matrix is split into two smaller matrices T_1 and T_2 as follows:

$$T = \left(\begin{array}{cccc|cccc} \alpha_1 & \beta_2 & & & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \beta_k & \alpha_k & \beta_{k+1} & & & \\ \hline & & & \beta_{k+1} & \alpha_{k+1} & \beta_{k+2} & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & & & & \beta_n & \alpha_n \end{array}\right)$$

$$= \begin{pmatrix} T_1 & \beta_{k+1} e_k e_1^T \\ \beta_{k+1} e_1 e_k^T & T_2 \end{pmatrix}.$$

(a) Let $\tilde{T}_1$ be the matrix T_1 with the element α_k replaced by $\alpha_k - \beta_{k+1}$ and $\tilde{T}_2$ be the matrix T_2 with the element α_{k+1} replaced by $\alpha_{k+1} - \beta_k$. Show that

$$T = \begin{pmatrix} \tilde{T}_1 & 0 \\ 0 & \tilde{T}_2 \end{pmatrix} + \beta_{k+1} \begin{pmatrix} e_k \\ e_1 \end{pmatrix} \begin{pmatrix} e_k^T & e_1^T \end{pmatrix}.$$

(b) The splitting in (a) is a rank one splitting of the form $T = D + \mu z z^T$, where D is block diagonal. Use the results in Theorem 9.2.10 to develop a divide and conquer algorithm for the eigenproblem of T.

9.6 Matrix Series and Matrix Functions

9.6.1 Convergence of Matrix Power Series

We start with a definition of the limit of a sequence of matrices:

Definition 9.6.1.
An infinite sequence of matrices $A_1, A_2, \ldots$ is said to converge to a matrix A, $\lim_{n\to\infty} A_n = A$, if

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

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

An infinite sum of matrices is defined by:

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

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

Theorem 9.6.2.
If $\|\cdot\|$ is any matrix norm, and $\sum_{k=0}^{\infty} \|B_k\|$ is convergent, then $\sum_{k=0}^{\infty} B_k$ is convergent.

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

A power series $\sum_{k=0}^{\infty} B_k z^n, z \in \mathbf{C}$, has a *circle of convergence* in the z-plane which is equivalent to the smallest of the circles of convergence corresponding to the series for the matrix elements. In the interior of the convergence circle, formal operations such as term-wise differentiation and integration with respect to z are valid for the element series and therefore also for matrix series.

We now investigate the convergence of matrix power series. First we prove a theorem which is also of fundamental importance for the theory of convergence of iterative methods studied in Chapter 10. We first recall the the following result:

Lemma 9.6.3. *For any consistent matrix norm*

$$\rho(A) \le \|A\|, \tag{9.6.1}$$

where $\rho(A) = \max_i |\lambda_i(A)|$ is the **spectral radius** *of A.*

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

We now return to the question of convergence of matrix series.

Theorem 9.6.4.
If the infinite series $f(z) = \sum_{k=0}^{\infty} a_k z^k$ has radius of convergence r, then the matrix series $f(A) = \sum_{k=0}^{\infty} a_k A^k$ converges if $\rho < r$, where $\rho = \rho(A)$ is the spectral radius of A. If $\rho > r$, then the matrix series diverges; the case $\rho = r$ is a "questionable case".

Proof. By Theorem 9.6.2 the matrix series $\sum_{k=0}^{\infty} a_k A^k$ converges if the series $\sum_{k=0}^{\infty} |a_k|\|A^k\|$ converges. By Theorem 9.6.5 for any $\epsilon > 0$ there is a matrix norm such that $\|A\|_T = \rho + \epsilon$. If $\rho < r$ then we can choose r_1 such that $\rho(A) \le r_1 < r$, and we have

$$\|A^k\|_T \le \|A\|_T^k \le (\rho + \epsilon)^k = O(r_1^k).$$

Here $\sum_{k=0}^{\infty} |a_k| r_1^k$ converges, and hence $\sum_{k=0}^{\infty} |a_k|\|A^k\|$ converges. If $\rho > r$, let $Ax = \lambda x$ with $|\lambda| = \rho$. Then $A^k x = \lambda^k x$, and since $\sum_{k=0}^{\infty} a_k \lambda^k$ diverges $\sum_{k=0}^{\infty} a_k A^k$ cannot converge. $\square$

Theorem 9.6.5.

Given a matrix $A \in \mathbf{R}^{n\times n}$ with spectral radius $\rho = \rho(A)$. Denote by $\|\cdot\|$ any l_p-norm, $1 \le p \le \infty$, and set $\|A\|_T = \|T^{-1}AT\|$. Then the following holds: If A has no defective eigenvalues with absolute value ρ then there exists a nonsingular matrix T such that

$$\|A\|_T = \rho.$$

Proof. If A is diagonalizable, we can simply take T as the diagonalizing transformation. Then clearly $\|A\|_T = \|D\| = \rho$, where $D = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$. In the general case, we first bring A to Jordan canonical form, $X^{-1}AX = J$, where

$$J = \operatorname{diag}\big(J_1(\lambda_1), \cdots, J_t(\lambda_t)\big),$$

and

$$J_i(\lambda_i) = \lambda_i I + N_i \in \mathbf{C}^{m_i\times m_i}, \quad m_i \ge 1,$$

and $J_i(\lambda_i)$ is a Jordan block. We shall find a diagonal matrix $D = \operatorname{diag}(D_1, \dots, D_t)$, such that a similarity transformation with $T = XD$, $K = T^{-1}AT = D^{-1}JD$ makes K close to the diagonal of J. Note that $\|A\|_T = \|K\|$, and

$$K = \operatorname{diag}(K_1, K_2, \dots, K_t), \quad K_i = D_i^{-1}J_i(\lambda_i)D_i.$$

If $m_i = 1$, we set $D_i = 1$, hence $\|K_i\| = |\lambda_i|$. Otherwise we choose

$$D_i = \operatorname{diag}\left(1, \delta_i, \delta_i^2, \dots, \delta_i^{m_i-1}\right), \quad \delta_i > 0. \tag{9.6.2}$$

Then $K_i = \lambda_i I + \delta_i N_i$, and $\|K\| = \max_i(\|K_i\|)$. (Verify this!) We have $\|N_i\| \le 1$, because $N_i x = (x_2, x_3, ..., x_{m_i}, 0)^T$, so $\|N_i x\| \le \|x\|$ for all vectors x. Hence,

$$\|K_i\| \le |\lambda_i| + \delta_i. \tag{9.6.3}$$

If $m_i > 1$ and $|\lambda_i| < \rho$, we choose $\delta_i = \rho - |\lambda_i|$, hence $\|K_i\| \le \rho$. $\square$

Note that

$$1/\kappa(T) \le \|A\|_T/\|A\| \le \kappa(T).$$

For every natural number n, we have $\|A^n\|_T \le \|A\|_T^n = \rho(A)^n$, and hence

$$\|A^n\|_p \le \kappa(T)\|A^n\|_T \le \kappa(T)\rho^n.$$

For some classes of matrices, an efficient (or rather efficient) norm can be found more easily than by the construction used in the proof of Theorem 9.6.5 This may have other advantages as well, e.g., a better conditioned T. Consider, for example, the weighted max-norm

$$\|A\|_w = \|T^{-1}AT\|_\infty = \max_i \sum_j |a_{ij}|w_j/w_i,$$

where $T = \operatorname{diag}(w_1, \dots, w_n) > 0$, and $\kappa(T) = \max w_i/\min w_i$. We then note that if we can find a positive vector w such that $|A|w \le \alpha w$, then $\|A\|_w \le \alpha$.

9.6.2 Analytic Matrix Functions

For an analytic function f with Taylor expansion $f(z) = \sum_{k=0}^{\infty} a_k z^k$, we define the matrix function by

$$f(A) = \sum_{k=0}^{\infty} a_k A^k. \tag{9.6.4}$$

(Note that this is not the same as a matrix-valued function of a complex variable.) Alternatively

$$f(A) = \int_{\Gamma} (zI - A)^{-1} f(z)\, dz \tag{9.6.5}$$

where Γ is any contour enclosing the spectrum of A [233].

If the matrix A is diagonalizable, $A = X\Lambda X^{-1}$, we define

$$f(A) = X \operatorname{diag}\big(f(\lambda_1), \dots, f(\lambda_n)\big) X^{-1} = X f(\Lambda) X^{-1}. \tag{9.6.6}$$

This expresses the matrix function $f(A)$ in terms of the function f evaluated at the spectrum of A and is often the most convenient way to compute $f(A)$.

For the case when A is not diagonalizable we first give an explicit form for the kth power of a Jordan block $J_m(\lambda) = \lambda I + N$. Since $N^j = 0$ for $j \geq m$ we get using the binomial theorem

$$J_m^k(\lambda) = (\lambda I + N)^k = \lambda^k I + \sum_{p=1}^{\min(m-1,k)} \binom{k}{p} \lambda^{k-p} N^p, \quad k \geq 1.$$

Since an analytic function can be represented by its Taylor series we are led to the following definition:

Definition 9.6.6.

Suppose that the analytic function $f(z)$ is regular for $z \in D \subset \mathbf{C}$, where D is a simply connected region, which contains the spectrum of A in its interior. Let

$$A = XJX^{-1} = X \operatorname{diag}\big(J_{m_1}(\lambda_1), \cdots, J_{m_t}(\lambda_t)\big) X^{-1}$$

be the Jordan canonical form of A. We then define

$$f(A) = X \operatorname{diag}\Big(f\big(J_{m_1}(\lambda_1)\big), \cdots, f\big(J_{m_t}(\lambda_t)\big)\Big) X^{-1}. \tag{9.6.7}$$

where the analytic function f of a Jordan block is

$$\begin{aligned} f(J_{m_k}) &= f(\lambda_k) I + \sum_{p=1}^{m-1} \frac{1}{p!} f^{(p)}(\lambda_k) N^p \\ &= \begin{pmatrix} f(\lambda_k) & f'(\lambda_k) & \cdots & \dfrac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{pmatrix}. \end{aligned} \tag{9.6.8}$$

If A is diagonalizable, $A = X^{-1}\Lambda X$, then for the exponential function we have,

$$\|e^A\|_2 = \kappa(X)e^{\alpha(A)},$$

where $\alpha(A) = \max_i \Re\lambda_i$ is the **spectral abscissa** of A and $\kappa(X)$ denotes the condition number of the eigenvector matrix. If A is normal, then V is orthogonal and $\kappa(V) = 1$.

One can show that for every non-singular matrix T it holds

$$f(T^{-1}AT) = T^{-1}f(A)T. \tag{9.6.9}$$

With this definition, the theory of analytic functions of a matrix variable closely follows the theory of a complex variable. If $\lim_{n\to\infty} f_n(z) = f(z)$ for $z \in D$, then $\lim_{n\to\infty} f_n(J(\lambda_i)) = f(J(\lambda_i))$. Hence, *if the spectrum of A lies in the interior of D* then $\lim_{n\to\infty} f_n(A) = f(A)$. This allows us to deal with operations involving limit processes.

The following important theorem can be obtained, which shows that Definition 9.6.6 is consistent with the more restricted definition (by a power series) given in Theorem 9.6.4.

Theorem 9.6.7.

All identities which hold for analytic functions of one complex variable z for $z \in D \subset \mathbf{C}$, where D is a simply connected region, also hold for analytic functions of one matrix variable A if the spectrum of A is contained in the interior of D. The identities also hold if A has eigenvalues on the boundary of D, provided these are not defective.

Example 9.6.1.

We have, for example,

$$\cos^2 A + \sin^2 A = I, \qquad \forall A;$$

$$\ln(I - A) = -\sum_{n=1}^{\infty} \frac{1}{n}A^n, \qquad \rho(A) < 1;$$

$$\int_0^{\infty} e^{-st}e^{At}dt = (sI - A)^{-1}, \qquad Re(\lambda_i) < Re(s);$$

For two arbitrary analytic functions f and g which satisfy the condition of Definition, it holds that $f(A)g(A) = g(A)f(A)$. However, when several non-commutative matrices are involved, one can no longer use the usual formulas for analytic functions.

Example 9.6.2.

$e^{(A+B)t} = e^{At}e^{Bt}$ for all t if and only if $BA = AB$. We have

$$e^{At}e^{Bt} = \sum_{p=0}^{\infty} \frac{A^p t^p}{p!} \sum_{q=0}^{\infty} \frac{B^q t^q}{q!} = \sum_{n=0}^{\infty} \frac{t^n}{n!} \sum_{p=0}^{n} \binom{n}{p} A^p B^{n-p}.$$

This is in general not equivalent to

$$e^{(A+B)t} = \sum_{n=0}^{\infty} \frac{t^n}{n!}(A+B)^n.$$

The difference between the coefficients of $t^2/2$ in the two expressions is

$$(A+B)^2 - (A^2 + 2AB + B^2) = BA - AB \neq 0, \qquad \text{if } BA \neq AB.$$

Conversely, if $BA = AB$, then it follows by induction that the binomial theorem holds for $(A+B)^n$, and the two expressions are equal.

An alternative definition of matrix functions can be obtained by using polynomial interpolation. Denote by $\lambda_1, \ldots . \lambda_t$ the distinct eigenvalues of A and let m_k be the index of λ_k, that is, the order of the largest Jordan block containing λ_k. Then $f(A) = p(A)$, where p is the unique Hermite interpolating polynomial of degree less than $\sum_{k=1}^{t}$ that satisfies the interpolating conditions

$$p^{(i)}(\lambda_k) = f^{(j)}(\lambda_k), \quad j = 0 : m_k, \quad i = 1 : t. \tag{9.6.10}$$

The function is said to be defined on the spectrum of A if all the derivatives in (9.6.10) exist. Note that the eigenvalues are allowed to be complex. The proof is left to Problem 7.10. Formulas for complex Hermite interpolation are given in Volume I, Sec. 4.3.2.

Note also that if $f(z)$ is analytic inside the closed contour C, and if the whole spectrum of A is inside C, the Cauchy integral definition is (cf. Problem 7.11)

$$\frac{1}{2\pi i} \int_C (zI - A)^{-1} f(z) dz = f(A). \tag{9.6.11}$$

9.6.3 Matrix Exponential and Logarithm

The **matrix exponential** e^{At}, where A is a constant matrix, can be defined by the series expansion

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \cdots.$$

This series converges for all A and t since the radius of convergence of the power series $\sum_{k=0}^{\infty} \|A\|^k t^k / k!$ is infinite. The series can thus be differentiated everywhere and

$$\frac{d}{dt}(e^{At}) = A + A^2t + \frac{1}{2!}A^3t^2 + \cdots = Ae^{At}.$$

Hence, $y(t) = e^{At}c \in \mathbf{R}^n$ solves the initial value problem for the linear system of ordinary differential equations with constant coefficients

$$dy(t)/dt = Ay(t), \quad y(0) = c. \tag{9.6.12}$$

Such systems occurs in many physical, biological, and economic processes. Similarly, the functions $\sin(z)$, $\cos(z)$, $\log(z)$, can be defined for matrix arguments from their Taylor series representation.

The matrix exponential and its qualitative behavior has been studied extensively. A wide variety of methods for computing e^A have been proposed; see Moler and Van Loan [281]. Consider the 2 by 2 upper triangular matrix

$$A = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}.$$

The exponential of this matrix is

$$e^{tA} = \begin{cases} \begin{pmatrix} e^{\lambda t} & \alpha \dfrac{e^{\lambda t} - e^{\mu t}}{\lambda - \mu} \\ 0 & e^{\mu t} \end{pmatrix}, & \text{if } \lambda \neq \mu, \\ \begin{pmatrix} e^{\lambda t} & \alpha t e^{\lambda t} \\ 0 & e^{\mu t} \end{pmatrix}, & \text{if } \lambda = \mu \end{cases}. \tag{9.6.13}$$

When $|\lambda - \mu|$ is small, but not negligible neither of these two expressions are suitable, since severe cancellation will occur in computing the divided difference in the (1,2)-element in (9.6.13). When the same type of difficulty occurs in non-triangular problems of larger size the cure is by no means easy!

Another property of e^{At} that does not occur in the scalar case is illustrated next.

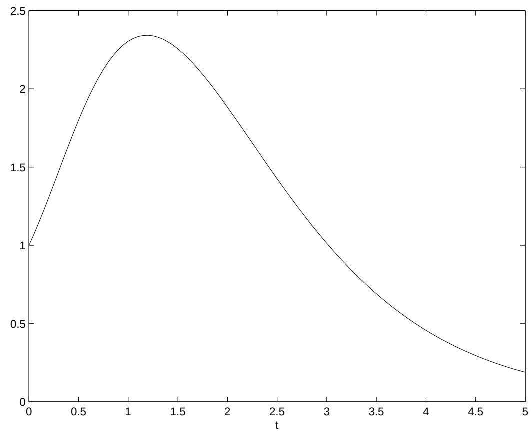

Figure 9.6.1. $\|e^{tA}\|$ *as a function of t for the matrix in Example 9.6.3.*

Example 9.6.3. Consider the matrix

$$A = \begin{pmatrix} -1 & 4 \\ 0 & -2 \end{pmatrix}.$$

Since $\max\{-1, -2\} = -1 < 0$ it follows that $\lim_{t\to\infty} e^{tA} = 0$. In Figure 9.6.1 we have plotted $\|e^{tA}\|_2$ as a function of t. The curve has a **hump** illustrating that as t increases some of the elements in e^{tA} first increase before they start to decay.

One of the best methods to compute e^A, the method of scaling and squaring, uses the fundamental relation

$$e^A = (e^{A/m})^m, \quad m = 2^s$$

of the exponential function. Here the exponent s is chosen so that $e^{A/m}$ can be reliably computed, e.g. from a Taylor or Padé approximation. Then $e^A = (e^{A/m})^{2^s}$ can be formed by squaring the result s times.

Instead of the Taylor series it is advantageous to use the diagonal Padé approximation of e^x; see Volume I, page 349.

$$r_{m,m}(z) = \frac{P_{m,m}(z)}{Q_{m,m}(z)} = \frac{\sum_{j=0}^{m} p_j z^j}{\sum_{j=0}^{n} q_j z^j}, \tag{9.6.14}$$

which are known explicitly for all m. We have

$$p_j = \frac{(2m-j)!\, m!}{(2m)!\,(m-j)!j!}, \qquad q_j = (-1)^j p_j, \quad j = 0 : m. \tag{9.6.15}$$

with the error

$$e^z - \frac{P_{m,m}(z)}{Q_{m,m}(z)} = (-1)^k \frac{(m!)^2}{(2m)!(2m+1)!} z^{2m+1} + O(z^{2m+2}). \tag{9.6.16}$$

Note that $P_{m,m}(z) = Q_{m,m}(-z)$, which reflects the property that $e^{-z} = 1/e^z$. The coefficients satisfy the recursion

$$p_0 = 1, \quad p_{j+1} = \frac{m-j}{(2m-j)(j+1)} p_j, \quad j = 0 : m-1. \tag{9.6.17}$$

To evaluate a diagonal Padé approximant of even degree m we can write

$$\begin{aligned} P_{2m,2m}(A) &= p_{2m}A^{2m} + \cdots + p_2 A^2 + p_0 I \\ &\quad + A(p_{2m-1}A^{2m-2} + \cdots + p_3 A^2 + p_1 I) = U + V. \end{aligned}$$

This can be evaluated with $m+1$ matrix multiplications by forming $A^2, A^4, \ldots, A^{2m}$. Then $Q_{2m}(A) = U - V$ needs no extra matrix multiplications. For an approximation of odd degree $2m+1$ we write

$$\begin{aligned} P_{2m+1,2m+1}(A) &= A(p_{2m+1}A^{2m} + \cdots + p_3 A^2 + p_1 I) \\ &\quad + p_{2m}A^{2m-2} + \cdots + p_2 A^2 + p_0 I = U + V. \end{aligned}$$

This can be evaluated with the same number of matrix multiplications and $Q_{2m+1}(A) = -U + V$. The final division $P_{k,m}(A)/Q_{m,m}(A)$ is performed by solving

$$Q_{m,m}(A) r_{m,m}(A) = P_{m,m}(A)$$

for $r_{m,m}(A)$ using Gaussian elimination.

The function `expm` in MATLAB uses a scaling such that $2^{-s}\|A\| < 1/2$ and a diagonal Padé approximant of degree $2m = 6$

$$P_{6,6}(z) = 1 + \frac{1}{2}z + \frac{5}{44}z^2 + \frac{1}{66}z^3 + \frac{1}{792}z^4 + \frac{1}{15840}z^5 + \frac{1}{665280}z^6.$$

```
function E = expmv(A);
% EXPMV computes the exponential of the matrix A
%
[f,e] = log2(norm(A,'inf'));
% Compute scaling parameter
s = max(0,e+1);
A = A/2^s;  X = A;
d = 2; c = 1/d;
E = eye(size(A)) + c*A;
D = eye(size(A)) - c*A;
m = 8; p = 1;
for k = 2:m
  d = d*(k*(2*m-k+1))/(m-k+1)
  c = 1/d;
  X = A*X;
  E = E + c*X;
  if p,
     D = D + c*X; else, D = D - c*X;
  end
  p = ~p;
end
E = D\E;
for k = 1:s, E = E*E; end
```

It can be shown ([281, Appendix A]) that then $r_{mm}(2^{-s}A)^{2^s} = e^{A+E}$, where

$$\frac{\|E\|}{\|A\|} < 2^3(2^{-s}\|A\|)^{2m}\frac{(m!)^2}{(2m)!(2m+1)!}.$$

For s and m chosen as in MATLAB this gives $\|E\|/\|A\| < 3.4 \cdot 10^{-16}$, which is close to the unit roundoff in IEEE double precision $2^{-53} = 1.11 \cdot 10^{-16}$. Note that this backward error result does not guarantee an accurate result. If the problem is inherently sensitive to perturbations the error can be large.

The analysis does not take roundoff errors in the squaring phase into consideration. This is the weak point of this approach. We have

$$\|A^2 - fl(A^2)\| \le \gamma_n\|A\|^2, \quad \gamma_n = \frac{nu}{1-nu}$$

but since possibly $\|A^2\| \ll \|A\|^2$ this is not satisfactory and shows the danger in matrix squaring. If a higher degree Padé approximation is chosen then the number of squarings can be reduced. Choices suggested in the literature (N. J. Higham [213]) are $m = 8$, with $2^{-s}\|A\| < 1.5$ and $m = 13$, with $2^{-s}\|A\| < 5.4$.

To compute the logarithm of a matrix the method of inverse scaling and squaring can be used; see Kenney and Laub [237]. The matrix

$$I + X = A^{1/2^k}$$

is computed by repeatedly taking square roots, until X is sufficiently small. Then a diagonal Padé approximant is used to compute $\log(I+X)$. The result is finally obtained from the identity

$$\log A = 2^k \log A^{1/2^k} = 2^k \log(I+X). \tag{9.6.18}$$

The Padé approximants can be obtained from the continued fraction expansion

$$\ln(1+x) = \frac{x}{1+}\,\frac{x}{2+}\,\frac{x}{3+}\,\frac{2^2x}{4+}\,\frac{2^2x}{5+}\cdots \tag{9.6.19}$$

The first few approximants are

$$r_{11} = X\frac{2}{2+X}, \quad r_{22} = X\frac{6+3X}{6+6X+X^2}, \quad r_{33} = X\frac{60+60X+11X^2}{60+90X+36X^2+3X^3}.$$

These Padé approximants can be evaluated , e.g., by Horner's scheme. Several other potentially more efficient methods are investigated by Higham [210].

These approximations contain both odd and even terms and unlike the exponential function there is no symmetry between the nominator and denominator that can be used to reduce the work. Using the identity

$$\ln(1+x) = \ln\left(\frac{1+z}{1-z}\right), \quad z = \frac{x/2}{1+x/2}, \tag{9.6.20}$$

the continued fraction expansion in z will contain only even terms.

$$\begin{aligned}\frac{1}{2z}\ln\left(\frac{1+z}{1-z}\right) &= \frac{1}{1-}\,\frac{z^2}{3-}\,\frac{2^2z^2}{5-}\,\frac{3^2z^2}{7-}\cdots\frac{n^2z^2}{2n+1}\cdots\\ &= 1+\frac{z^2}{3}+\frac{z^4}{5}+\cdots.\end{aligned}$$

The convergents of this expansion gives the Padé approximants; see Volume I, Example 3.5.6, the first few of which are

$$s_{11} = \frac{15+4z^2}{3(5-3z^2)}. \quad s_{22} = \frac{945-735z^2+64z^4}{15(63-70z^2+15z^4)}.$$

Here the diagonal approximants s_{mm} are most interest. For example, the approximation s_{22} matches the Taylor series up to the term z^8 and the error is approximately equal to the term $z^{10}/11$. Note that the denominators are the Legendre polynomials in $1/z$,

To evaluate $\ln(I+X)$, we first compute

$$Z = (I+\tfrac{1}{2}X)^{-1}\tfrac{1}{2}X, \qquad Z^2 = Z * Z,$$

using an LU factorization of $I+\frac{1}{2}X)$. The nominator and denominator polynomials in the Padé approximants are then evaluated, by Horner's scheme, e.g., for s_{22}

$$P(Z^2) = 63I - 49Z^2 + (64/15)Z^4, \qquad Q(Z^2) = 63I - 70Z^2 + 15Z^4.$$

Finally, the quotient $Q(Z^2)^{-1}P(Z^2)$ is computed by performing an LU factorization of $Q(Z^2)$.

9.6.4 Matrix Square Root

Any matrix X such that $X^2 = A$, where $A \in \mathbf{C}^{n\times n}$ is called a **square root** of A and denoted by $X = A^{1/2}$. Unlike a square root of a scalar, the square root of a matrix may not exist. For example, it is easy to verify that the matrix

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

can have no square root; see Problem 9.6.8.

If A is nonsingular and has s distinct eigenvalues then it has precisely 2^s square roots that are expressible as polynomials in the matrix A. If some eigenvalues appear in more than one Jordan block then there are infinitely many additional square roots, none of which can be expressed as a polynomial in A. For example, any Householder matrix is a square root of the identity matrix.

There is a **principal square root** of particular interest, namely the one whose eigenvalues lie in the right half plane. The principal square root, when it exists, is a polynomial in the original matrix.

Lemma 9.6.8.

Assume that $A \in \mathbf{C}^{n\times n}$ has no eigenvalues on the closed negative axis $\Re^-$. Then there exists a unique square root $X = A^{1/2}$, such that all the eigenvalues of X lie in the open right half plane and X is a primary matrix function of A. This matrix X is called a principal square root of A.

We assume in the following that the condition in Lemma 9.6.8 is satisfied. When A is symmetric positive definite the principal square root is the unique symmetric positive definite square root.

If X_k is an approximation to $X = A^{1/2}$ and we put $X = X_k + E_K$, then

$$A = (X_k + E_k)^2 = X_k^2 + X_kE_k + E_kX_k + E_k^2.$$

Ignoring the term E_k^2 gives

$$X_{k+1} = X_k + E_k, \quad X_kE_k + E_kX_k = A - X_k^2. \tag{9.6.21}$$

In general this iteration is expensive since at each step a Sylvester equation has to be solved for E_k. But if the initial approximation X_0 is a polynomial in A, then all following X_k will also commute with A. Then the iteration (9.6.21) simplifies to

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}A).$$

This Newton iteration is quadratically convergent. Numerically it is unstable since the computed approximations will not commute with A and this simplified iteration should not be used.

Several stable variants of the Newton iteration have been suggested, for example the Denman–Beavers [98] iteration

$$X_{k+1} = \frac{1}{2}(X_k + Y_k^{-1}), \qquad Y_{k+1} = \frac{1}{2}(Y_k + X_k^{-1}), \tag{9.6.22}$$

with initial conditions $X_0 = A$, $Y_0 = I$. For this it holds that $\lim_{k\to\infty} X_k = A^{1/2}$ and $\lim_{k\to\infty} X_k = A^{-1/2}$. This iteration needs two matrix inversions per iteration. It is mathematically equivalent to (9.6.21) and hence also quadratically convergent.

The pth root of a matrix, denoted by $A^{1/p}$, a nd its inverse $A^{-1/p}$, can be similarly defined. The principal pth root is the unique matrix which satisfies $X^p = A$ and whose eigenvalues lie in the segment $\{z \mid -\pi/p < \arg(z) < \pi/p\}$. Newton's iteration becomes

$$X_{k+1} = \frac{1}{p}((p-1)X_k + X_k^{1-p}A).$$

It has recently been proved that (in exact arithmetic¿) this iteration converges quadratically if $X_0 = I$ and all eigenvalues of A lie in the union of the open positive real axis and the set

$$\{zin\mathbf{C} \mid \Re x > 0 \text{ and } |z| \le 1\}. \tag{9.6.23}$$

The Newton iteration can be rewritten as (see Iannazzo [221, 222])

$$X_{k+1} = X_k \left(\frac{(p-1)I + M_k}{p}\right), \qquad M_{k+1} = \left(\frac{(p-1)I + M_k}{p}\right)^{-p} M_k, \tag{9.6.24}$$

with initial values equal to $X_0 = I$ and $M_0 = A$.

We now describe a method to compute the principal square root based on the Schur decomposition $A = QSQ^H$, where Q is unitary and S upper triangular. If U is an upper triangular square root of S, then $X = QUQ^H$ is a square root of A. If A is a normal matrix then $S = \operatorname{diag}(\lambda_i)$ and we can just take $U = \operatorname{diag}(\lambda_i^{1/2})$. Otherwise, from the relation $S = U^2$ we get

$$s_{ij} = \sum_{k=i}^{j} u_{ik}u_{kj}, \quad i \le j. \tag{9.6.25}$$

This gives a recurrence relation for determining the elements in U. For the diagonal elements in U we have

$$u_{ii} = s_{ii}^{1/2}, \quad i = 1:n, \tag{9.6.26}$$

and further

$$u_{ij} = \left(s_{ij} - \sum_{k=i+1}^{j-1} u_{ik}u_{kj}\right)\Big/(u_{ii} + u_{jj}). \quad i < j. \tag{9.6.27}$$

Hence, the elements in U can be determined computed from (9.6.27), for example, one diagonal at a time. Since whenever $s_{ii} = s_{jj}$ we take $u_{ii} = u_{jj}$ this recursion does not break down. (Recall we assumed that at most one diagonal element of S is zero.)

If we let $\bar{U}$ be the computed square root of S then it can be shown that

$$\bar{U}^2 = S + E, \qquad \|E\| \le c(n)u(\|S\| + \|U\|^2),$$

where u is the unit roundoff and $c(n)$ a small constant depending on n. If we define

$$\alpha = \|A^{1/2}\|^2/\|A\|,$$

then we have

$$\|E\| \leq c(n)u(1+\alpha)\|S\|.$$

We remark that for real matrices an analogue algorithm can be developed, which uses the real Schur decomposition and only employs real arithmetic.

9.6.5 Polar Decomposition and the Matrix Sign Function

Several matrix functions can be expressed in terms of the square root. In the polar decomposition of $A = PH \in \mathbf{C}^{m\times n}$ the factors can be expressed

$$H = (A^H A)^{1/2}, \qquad P = A(A^T A)^{-1/2}. \tag{9.6.28}$$

We now consider iterative methods for computing the unitary factor P. The related Hermitian factor then is $H = P^H A$. When P is a computed factor then one should take

$$H = \frac{1}{2}(P^H A + (P^H A)^H)$$

to ensure that H is Hermitian. If A is sufficiently close to a unitary matrix, then a series expansion can be used to compute P. If $z = |z|e^{i\varphi}$ is a complex number, then

$$z/|z| = z(1-(1-|z|^2))^{-1/2} = z(1-q)^{-1/2}, \quad q = 1-|z|^2.$$

and expanding the right hand side in a Taylor series in q gives

$$z(1-q)^{-1/2} = z\Big(1 + \frac{1}{2}q + \frac{3}{8}q^2 + \cdots\Big).$$

The related matrix expansion is

$$P = A\Big(I + \frac{1}{2}U + \frac{3}{8}U^2 + \cdots\Big), \quad U = I - A^H A, \tag{9.6.29}$$

This suggests the following iterative algorithm: Put $A_0 = A$, and for $k = 0, 1, 2, \ldots$, compute

$$A_{k+1} = A_k\Big(I + \frac{1}{2}U_k + \frac{3}{8}U_k^2 + \cdots\Big), \quad U_k = I - A_k^H A_k. \tag{9.6.30}$$

In particular, terminating the expansion (9.6.29) after the linear term we have

$$A_{k+1} = A_k\Big(I + \frac{1}{2}U_k\Big) = \frac{1}{2}A_k(3I - A_k^H A_k), \quad k = 0, 1, 2, \ldots.$$

The rate of convergence is quadratic. It can be shown (see [41]) that a sufficient condition for convergence is that for some consistent matrix norm $\|I - A^H A\| \leq 2$.

An alternative rational iterative algorithm for computing the polar decomposition is the Newton iteration, Assume that $A_0 = A$ is square and nonsingular, we compute

$$A_{k+1} = \frac{1}{2}\big(A_k + (A_k^H)^{-1}\big). \tag{9.6.31}$$

This iteration avoids the possible loss of information associated with the explicit formation of A^HA. It is globally convergent to P and asymptotically the convergence is quadratic. The corresponding scalar iteration is the Newton iteration for the square root of 1. The iteration (9.6.31) does not suffer from the instability noted before of the Newton square root iteration because the unit matrix commutes with any matrix.

A drawback with the iteration (9.6.31) is that it cannot directly be applied to a rectangular or singular matrix A. If it is rewritten in the form

$$A_{k+1} = \frac{1}{2}A_k\big(I + (A_k^H A_k)^{-1}\big), \tag{9.6.32}$$

we get a formula that can be applied in the rectangular and rank deficient case . Another possibility to treat a rectangular matrix is to first perform a QR factorization

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix},$$

where R is nonsingular. If the polar decomposition of R is $R = PH$ then $A = (QP)H$ is the polar decomposition of A. Hence, we can apply the iteration (9.6.31) to R.

To analyze the iteration (9.6.31) we let the singular value decomposition of $A_0 = A$ be

$$A_0 = U\Sigma_0 V^H, \quad \Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_n).$$

Then

$$A_k = UD_kV^H, \quad D_k = \operatorname{diag}(d_1^{(k)}, \dots, d_n^{(k)}),$$

where $D_0 = \Sigma$, and by (9.6.31)

$$D_{k+1} = \frac{1}{2}\big(D_k + (D_k)^{-1}\big), \quad k \geq 0.$$

Thus, (9.6.31) is equivalent to n uncoupled scalar iterations

$$d_i^{(0)} = \sigma_i, \qquad d_i^{(k+1)} = \frac{1}{2}\big(d_i^{(k+1)} + 1/d_i^{(k+1)}\big), \quad i = 1:n. \tag{9.6.33}$$

From familiar relations for the Newton square root iteration we know that

$$\frac{d_i^{(k+1)} - 1}{d_i^{(k+1)} + 1} = \Big(\frac{d_i^{(k)} - 1}{d_i^{(k)} + 1}\Big)^2 = \cdots = \Big(\frac{\sigma_i - 1}{\sigma_i + 1}\Big)^{2^{k+1}}. \quad i = 1:n. \tag{9.6.34}$$

Note that the convergence depends on the spread of the singular values of A but is independent of n.

Initially the convergence of the iteration can be slow. It follows by inspecting (9.6.33) that singular values $\sigma_i \gg 1$ will initially be reduced by a factor of two in each step. Similarly singular values $\sigma_i \ll 1$ will in the first iteration be transformed into a large singular value and then be reduced by a factor of two. Convergence can be accelerated by using the fact that the orthogonal polar factor of the scaled matrix γA, $\gamma \neq 0$, is the same as for A. The scaled version of the iteration is $A_0 = A$,

$$A_{k+1} = \frac{1}{2}\big(\gamma_k A_k + (\gamma_k A_k^H)^{-1}\big), \quad k = 0, 1, 2, \dots, \tag{9.6.35}$$

where γ_k are scale factors. The optimal scale factors are determined by the condition that $\gamma_k \sigma_1(A_k) = 1/(\gamma_k \sigma_n(A_k))$ or

$$\gamma_k = (\sigma_1(A_k)\sigma_n(A_k))^{-1/2}.$$

Since the singular values are not known we must use some cheaply computable approximation to these optimal scale factors. We have the estimate

$$\sigma_1(A) = \|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty} \leq \sqrt{n}\|A\|_2.$$

Defining

$$\alpha_k = \sqrt{\|A_k\|_1 \|A_k\|_\infty}, \qquad \beta_k = \sqrt{\|A_k^{-1}\|_1 \|A_k^{-1}\|_\infty}$$

we use the scale factors

$$\gamma_k = (\alpha_k/\beta_k)^{-1/2}.$$

(see Higham [208] and Kenney and Laub [239])

Assume that the matrix $A \in \mathbf{C}^{n\times n}$ has no eigenvalues on the imaginary axis. Let

$$A = Z^{-1} J_A Z, \quad J_A = \begin{pmatrix} J_A^{(1)} & 0 \\ 0 & J_A^{(1)} \end{pmatrix},$$

be the Jordan canonical form arranged so that the n_1 eigenvalues of $J_A^{(1)}$ lie in the open left half-plane and the n_2 eigenvalues of $J_A^{(2)}$ in the open right half-plane $(n_1 + n_2 = n)$. Then the **matrix sign function** is defined as

$$S = \operatorname{sign}(A) = Z^{-1} \begin{pmatrix} I_{n_1} & 0 \\ 0 & I_{n_2} \end{pmatrix} Z. \tag{9.6.36}$$

The matrix sign function can be used to solve many problems in control theory. For example, the solution to the Lyapunov equation

$$A^T Z + ZA = -Q,$$

can be found from the sign function identity

$$\operatorname{sign}\begin{pmatrix} -A^T & \frac{1}{2}Q \\ 0 & A \end{pmatrix} = \begin{pmatrix} I & Z \\ 0 & -I \end{pmatrix}.$$

It is related to the matrix square root by the identity

$$\operatorname{sign}(A) = A(A^2)^{-1/2}.$$

Instead of computing $\operatorname{sign}(A)$ from its definition (9.6.36) it is usually more efficient to use an iterative algorithm rich in matrix multiplications, such as the Newton iteration

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \quad X_0 = A. \tag{9.6.37}$$

The sign function has the property that $S^2 = I$, that is S is **involutory**. As for the polar decomposition the corresponding scalar iteration is the Newton iteration for computing the square root of 1. This iteration converges quadratically to sign$\,(A)$, provided A has no eigenvalues on the imaginary axis.

The iteration (9.6.37) can be analyzed using the eigen-decomposition of A. In the iteration for the polar factor the convergence was determined by the convergence of the singular values of A to 1. For the matrix sign function the same can be said for the convergence of the eigenvalues to ± 1. Convergence can be accelerated by introducing a scaling factor, taking $A_0 = A$,

$$A_{k+1} = \frac{1}{2}\left(\gamma_k A_k + (\gamma_k A_k)^{-1}\right), \quad k \geq 0. \tag{9.6.38}$$

where γ_k are a real and positive scalars. It can be shown that this iteration converges to sign$\,(A)$ if and only if $\gamma_k \to 1$.

The eigenvalues of A_k satisfy the scalar recurrences

$$\lambda_i^{(k+1)} = \frac{1}{2}\left(\mu_k \lambda_i^{(k)} + (\mu_k \lambda_i^{(k)})^{-1}\right), \quad i = 1 : n, \quad k \geq 0.$$

For the matrix sign function the optimal scaling is more complicated since it requires a complete knowledge of the spectrum of the matrix. Semi-optimal scalings have been developed that show good results; see Kenney and Laub [239].

9.6.6 Finite Markov Chains

A finite **Markov chain**[42] is a stochastic process, i.e. a sequence of random variables X_t, $t = 0, 1, 2, \dots$, in which each X_t can take on a finite number of different states $\{s_i\}_{i=1}^n$. The future development is completely determined by the present state and not at all in the way it arose. In other words, the process has no memory. Such processes have many applications in the Physical, Biological and Social sciences.

At each time step t the probability that the system moves from state s_i to state s_j is independent of t and equal to

$$p_{ij} = Pr\{X_t = s_j \mid X_{t-1} = s_i\}.$$

The matrix $P \in \mathbf{R}^{n\times n}$ of **transition probabilities** is nonnegative and must satisfy

$$\sum_{j=1}^n p_{ij} = 1, \quad i = 1 : n. \tag{9.6.39}$$

i.e., each row sum of P is equal to 1. Such a matrix is called **row stochastic**.

Let $p_i(t) \geq 0$ be the probability that a Markov chain is in state s_i at time t. Then the probability distribution vector, also called the **state vector**, is

$$p^T(t) = (p_1(t), p_2(t), \dots, p_n(t)), \quad t = 0, 1, 2, \dots$$

[42] Named after the Russian mathematician Andrej Andreevic Markov (1856–1922), who introduced them in 1908.

The initial probability distribution is given by the vector $p(0)$. Clearly we have $p(t+1) = P^T p(t)$ and

$$p(t) = (P^t)^T p(0), \quad t = 1, 2, \ldots.$$

In matrix-vector form we can write (9.6.39) as

$$Pe = e, \quad e = (1,\, 1,\, \ldots,\, 1)^T. \tag{9.6.40}$$

Thus, e is a *right* eigenvector of P corresponding to the eigenvalue $\lambda = 1$ and

$$P^k e = P^{k-1}(Pe) = P^{k-1}e = \cdots = e, \quad k > 1.$$

That is the matrix P^k, $k > 1$ is also row stochastic and is the **k-step transition matrix**.

An important problem is to find a **stationary distribution** p of a Markov chain. A state vector p of a Markov chain is said to be **stationary** if

$$p^T P = p^T, \quad p^T e = 1. \tag{9.6.41}$$

Hence, p is a *left* eigenvector of the transition matrix P corresponding to the eigenvalue $\lambda = 1 = \rho(P)$. Then p solves the singular homogeneous linear system

$$A^T p = 0, \ \text{ subject to } \ e^T p = 1, \quad A = I - P, \tag{9.6.42}$$

and p lies in the nullspace of A^T.

If the transition matrix P of a Markov chain is irreducible the chain is said to be **ergodic**. Then from the Perron–Frobenius Theorem it follows that $\lambda = 1$ is a *simple eigenvalue of* P and $\operatorname{rank}(A) = n - 1$. Further, there is a unique positive eigenvector p with $\|p\|_1 = 1$, satisfying (9.6.41). and any subset of $n-1$ columns (rows) of A are linearly independent (otherwise p would have some zero component).

If $P > 0$, there is no other eigenvalue with modulus $\rho(P)$ and we have the following result:

Theorem 9.6.9.

Assume that a Markov chain has a positive transition matrix. Then, independent of the initial state vector,

$$\lim_{t\to\infty} p(t) = p,$$

where p spans the nullspace of $A^T = (I - P^T)$.

If P is not positive then, as shown by the following example, the Markov chain may not converge to a stationary state.

Example 9.6.4.

Consider a Markov chain with two states for which state 2 is always transformed into state 1 and state 2 into state 1. The corresponding transition matrix

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

with two eigenvalues of modulus $\rho(P)$: $\lambda_1 = 1$ and $\lambda_2 = -1$. Here P is symmetric and its left eigenvalue equals $p = (0.5,\, 0.5)^T$. However, for any initial state different from p, the state will oscillate and not converge.

This example can be generalized by considering a Markov chain with m states and taking P equal to the permutation matrix corresponding to a cyclic shift. Then P has m eigenvalues on the unit circle in the complex plane.

Many results in the theory of Markov chains can be phrased in terms of the so-called **group inverse** of the matrix $A = (I - P)$.

Definition 9.6.10.

The **Drazin inverse** *of A is the unique matrix X satisfying the three equations*

$$A^k XA = A, \quad XAX = X, \quad AX = XA. \tag{9.6.43}$$

It exists only if $k \geq \operatorname{index}(A)$, where

$$\operatorname{index}(A) = \min\{k \mid \operatorname{rank}(A^{k+1}) = \operatorname{rank}(A^k)\}. \tag{9.6.44}$$

The group inverse $A^\ddagger$ of A is the unique matrix satisfying (9.6.43) for $k = 1$.

The two first identities in (9.6.43) say that Drazin inverse X is an $(1,2)$-inverse. The last identity says that X commutes with A. The group inverse of A exists if and only if the matrix A has index one, i.e. $\operatorname{rank}(A^2) = \operatorname{rank}(A)$.[43] This condition is satisfied for every transition matrix (Meyer [278, Theorem 2.1]). Further, we have

$$I - ep^T = AA^\ddagger.$$

(Note that $AA^\ddagger$ is a projection matrix since $(ep^T)^2 = p^T eep^T = ep^T$.)

Theorem 9.6.11 (*Golub and Meyer [172]***).**

Lat $A = I - P$, where P is the transition matrix of an ergodic Markov chain and consider the QR factorization of A. Then the R-factor is uniquely determined and has the form

$$R = \begin{pmatrix} U & -Ue \\ 0 & 0 \end{pmatrix}. \tag{9.6.45}$$

The stationary distribution p is given by the last column $q = Qe_n$ of Q and equals

$$p = q \Big/ \sum_{i=1}^{n} q_i. \tag{9.6.46}$$

Further, it holds that

$$A^\ddagger = (I - ep^T) \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T (I - ep^T), \tag{9.6.47}$$

[43] It is known that if this condition holds then A belongs to a set that forms a multiplicative group under ordinary matrix multiplication.

where p is the stationary distribution of for P.

Proof.

That R has the form above follows from the fact that the first $n-1$ columns of A are linearly independent and that

$$0 = Ae = \begin{pmatrix} U & u_n \\ 0 & 0 \end{pmatrix} e = \begin{pmatrix} Ue + u_n \\ 0 \end{pmatrix}$$

and thus $u_n = -Ue$. Since the last row of $R = Q^T A$ is zero, it is clear that $q^T A = 0$, where q is the last column of e_n. By the Perron–Frobenius theorem it follows that $q > 0$ or $q < 0$ and (9.6.46) follows.

If we set

$$A^- = \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T,$$

then it can be verified that $AA^-A = A$. Using the definition of a group inverse it follows that

$$\begin{aligned} A^\ddagger &= A^\ddagger A A^\ddagger = A^\ddagger (AA^-A) A^\ddagger = (A^\ddagger A) A^- (AA^\ddagger) \\ &= (I - ep^T) \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T (I - ep^T). \end{aligned}$$

□

For an ergodic chain define the matrix M of *mean first passage times*, where the element m_{ij} is the expected number of steps before entering state s_j after the initial state s_i. These matrices are useful in analyzing, e.g., safety systems and queuing models. The matrix M is the unique solution of the linear equation

$$AX = ee^T - P \operatorname{diag}(X).$$

The mean first passage times matrix M can be expressed in terms of $A^\ddagger$ as

$$M = I - A^\ddagger + ee^T \operatorname{diag}(A^\ddagger).$$

The theory of Markov chains for general reducible nonnegative transition matrices P is more complicated. It is then necessary to classify the states. We say that a state s_i has access to a state s_j if it is possible to move from state s_i to s_j in a finite number of steps. If also s_j has access to s_i s_i and s_j are said to communicate. This is an equivalence relation on the set of states and partitions the states into classes. If a class of states has access to no other class it is called **final**. If a final class contains a single state then the state is called **absorbing**.

Suppose that P has been permuted to its block triangular form

$$P = \begin{pmatrix} P_{11} & 0 & \dots & 0 \\ P_{21} & P_{22} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ P_{s1} & P_{s2} & \dots & P_{ss} \end{pmatrix} \tag{9.6.48}$$

where the diagonal blocks P_{ii} are square and irreducible. Then these blocks correspond to the classes associated with the corresponding Markov chain. The class associated with P_{ii} is final if and only if $P_{ij} = 0$, $j = 1 : i-1$. If the matrix P is irreducible then the corresponding matrix chain contains a single class of states.

Example 9.6.5. Suppose there is an epidemic in which every month 10% of those who are well become sick and of those who are sick 20% dies, and the rest become well. This can be modeled by the Markov process with three states dead, sick,well, and transition matrix

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0.1 & 0 & 0.9 \\ 0 & 0.2 & 0.8 \end{pmatrix}.$$

Then the left eigenvector is $p = e_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T$, i.e. in the stationary distribution all are dead. Clearly the class dead is absorbing!

We now describe a way to force a Markov chain to become irreducible.

Example 9.6.6 (Eldén).

Let $P \in \mathbf{R}^{n\times n}$ be a row stochastic matrix and set

$$Q = \alpha P + (1-\alpha)\frac{1}{n}ee^T, \quad \alpha > 0,$$

where e is a vector of all ones. Then $Q > 0$ and since $e^Te = n$ we have $Pe = (1-\alpha)e + \alpha e = 1$, so Q is row stochastic. From the Perron–Frobenius Theorem it follows that there is no other eigenvalue of Q with modulus 1

We now show that if the eigenvalues of P equal $1, \lambda_2, \lambda_3, \dots, \lambda_n$ then the eigenvalues of Q are $1, \alpha\lambda_2, \alpha\lambda_3, \dots, \alpha\lambda_n$. Proceeding as in the proof of the Schur normal form (Theorem 9.1.11) we define the orthogonal matrix $U = (u_1\ U_2)$, where $u_1 = e/\sqrt{n}$. Then

$$\begin{aligned} U^TPU &= U^T\begin{pmatrix} P^Tu_1 & P^TU_2 \end{pmatrix} = U^T\begin{pmatrix} u_1 & P^TU_2 \end{pmatrix} \\ &= \begin{pmatrix} u_1^Tu_1 & u_1^TP^TU_2 \\ U_2^Tu_1 & U_2^TP^TU_2 \end{pmatrix} = \begin{pmatrix} 1 & v^T \\ 0 & T \end{pmatrix}. \end{aligned}$$

This is a similarity transformation so T has eigenvalues $\lambda_2, \lambda_3, \dots, \lambda_n$. Further, $U^Te = \sqrt{n}e_1$ so that $U^Tee^TU = ne_1e_1^T$, and we obtain

$$\begin{aligned} U^TQU &= U^T\left(\alpha P + (1-\alpha)\frac{1}{n}ee^T\right)U \\ &= \alpha\begin{pmatrix} 1 & v^T \\ 0 & T \end{pmatrix} + (1-\alpha)\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & \alpha v^T \\ 0 & \alpha T \end{pmatrix}. \end{aligned}$$

The result now follows.

Review Questions

6.1 Define the matrix function e^A. Show how this can be used to express the solution to the initial value problem $y'(t) = Ay(t)$, $y(0) = c$?

6.2 What can be said about the behavior of $\|A^k\|$, $k \gg 1$, in terms of the spectral radius and the order of the Jordan blocks of A? (See Problem 8.)

6.3 (a) Given a square matrix A. Under what condition does there exist a vector norm, such that the corresponding operator norm $\|A\|$ equals the spectral radius? If A is diagonalizable, mention a norm that has this property.

(b) What can you say about norms that come close to the spectral radius, when the above condition is not satisfied? What sets the limit to their usefulness?

6.4 Show that

$$\lim_{t\to\infty} \frac{1}{t} \ln \|e^{At}\| = \max_{\lambda\in\lambda(A)} \mathrm{Re}(\lambda), \qquad \lim_{t\to 0} \frac{1}{t} \ln \|e^{At}\| = \mu(A).$$

6.5 Under what conditions can identities which hold for analytic functions of complex variable(s) be generalized to analytic functions of matrices?

6.6 (a) Show that any permutation matrix is doubly stochastic.

(b) What are the eigenvalues of matrix

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}?$$

6.7 Suppose that P and Q are row stochastic matrices.

(a) Show that $\alpha P + (1 - \alpha Q)$ is a row stochastic matrix.

(b) Show that PQ is a row stochastic matrix.

Problems and Computer Exercises

6.1 (a) Let $A \in \mathbf{R}^{n\times n}$, and consider the matrix polynomial

$$p(A) = a_0 A^n + a_1 A^{n-1} + \cdots + a_n I \in \mathbf{R}^{n\times n}.$$

Show that if $Ax = \lambda x$ then $p(\lambda)$ is an eigenvalue and x an associated eigenvector of $p(A)$.

(b) Show that the same is true in general for an analytic function $f(A)$. Verify (9.6.9). Also construct an example, where $p(A)$ has other eigenvectors in addition to those of A.

6.2 Show that the series expansion

$$(I - A)^{-1} = I + A + A^2 + A^3 + \dots$$

converges if $\rho(A) < 1$.

6.3 (a) Let $\|\cdot\|$ be a consistent matrix norm, and ρ denote the spectral radius. Show that

$$\lim_{k\to\infty} \|A^k\|^{1/k} = \rho(A).$$

(b) Show that

$$\lim_{t\to\infty} \frac{\ln \|e^{At}\|}{t} = \max_{\lambda\in\lambda(A)} \Re(\lambda).$$

Hint: Assume, without loss of generality, that A is in its Jordan canonical form.

6.4 Show that

$$e^A \otimes e^B = e^{B\oplus A},$$

where $\oplus$ denotes the Kronecker sum.

6.5 (a) Show that if $A = \begin{pmatrix} \lambda_1 & 1 \\ 0 & \lambda_2 \end{pmatrix}$ and $\lambda_1 \neq \lambda_2$ then

$$f(A) = \begin{pmatrix} f(\lambda_1) & \dfrac{f(\lambda_1) - f(\lambda_2)}{\lambda_1 - \lambda_2} \\ 0 & f(\lambda_2) \end{pmatrix}.$$

Comment on the numerical use of this expression when $\lambda_2 \to \lambda_1$.

(b) For $A = \begin{pmatrix} 0.5 & 1 \\ 0 & 0.6 \end{pmatrix}$, show that $\ln(A) = \begin{pmatrix} -0.6931 & 1.8232 \\ 0 & 0.5108 \end{pmatrix}$.

6.6 (a) Compute e^A, where

$$A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix},$$

using the method of scaling and squaring. Scale the matrix so that $\|A/2^s\|_\infty < 1/2$ and approximate the exponential of the scaled matrix by a Padé approximation of order (4,4).

(b) Compute the eigendecomposition $A = X\Lambda X^{-1}$ and obtain $e^A = Xe^\Lambda X^{-1}$. Compare the result with that obtained in (a).

6.7 (Higham [208]) (a) Let $A \in \mathbf{R}^{n\times n}$ be a symmetric and positive definite matrix. Show that if

$$A = LL^T, \qquad L^T = PH$$

are the Cholesky and polar decomposition respectively, then $A^{1/2} = H$.

(b) The observation in (a) lead to an attractive algorithm for computing the square root of A. Suppose s steps of the iteration (9.6.31) is needed to compute the polar decomposition. How many flops are required to compute the square root if the triangular form of L is taken into account?

6.8 (a) Show that the relation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

cannot hold for any a, b, c, and d.
(b) Show that $X^2 = A$, where

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad X = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Can X be represented as a polynomial in A?

6.9 Show the relation

$$\operatorname{sign}\begin{pmatrix} 0 & A \\ I & 0 \end{pmatrix} = \begin{pmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{pmatrix}. \tag{9.6.49}$$

6.10 Show that an analytic function of the matrix A can be computed by Newton's interpolation formula, i.e.,

$$f(A) = f(\lambda_1)I + \sum_{j=1}^{n^*} f(\lambda_1, \lambda_2, \dots, \lambda_j)(A - \lambda_1 I) \cdots (A - \lambda_j I)$$

where λ_j, $j = 1 : n^*$ are the distinct eigenvalues of A, each counted with the same multiplicity as in the minimal polynomial. Thus, n^* is the degree of the minimal polynomial of A.

6.11 We use the notation of Theorem 9.6.5. For a given n, show by an appropriate choice of ϵ that $\|A^n\|_p \le Cn^{m^*-1}\rho^n$, where C is independent of n. Then derive the same result from the Jordan Canonical form.

Hint: See the comment after Theorem 9.6.5.

6.12 Let C be a closed curve in the complex plane, and consider the function,

$$\phi_C(A) = \frac{1}{2\pi i}\int_C (zI - A)^{-1} dz,$$

If the whole spectrum of A is inside C then, by (9.6.11) $\phi_C(A) = I$. What is $\phi_C(A)$, when only part of the spectrum (or none of it) is inside C? Is it generally true that $\phi_C(A)^2 = \phi_C(A)$?

Hint: First consider the case, when A is a Jordan block.

6.13 Higher order iterations for the orthogonal factor in the polar decomposition based on Padé approximations are listed in [238]. A special case is the third order Halley's method, (see Vol. I, Sec. 6.3.4). which has been studied by Gander [144]. This leads to the iteration

$$A_{k+1} = A_k\big(3I + A_k^H A_k\big)\big(I + 3A_k^H A_k\big)^{-1}.$$

Implement this method and test it on ten matrices $A \in \mathbf{R}^{20\times 10}$ with random elements uniformly distributed on $(0,1)$. (In MATLAB such matrices are generated by the command $A = rand(20,10)$.) How many iteration are needed to reduce $\|I - A_k^T A_k\|_F$ to the order of machine precision?

9.7 The Rayleigh–Ritz Procedure

In many applications eigenvalue problems arise involving matrices so large that they cannot be conveniently treated by the methods described so far. For such problems, it is not reasonable to ask for a complete set of eigenvalues and eigenvectors, and usually only some extreme eigenvalues (often at one end of the spectrum) are required. In the 1980's typical values could be to compute 10 eigenpairs of a matrix of order 10,000. In the late 1990's problems are solved where 1,000 eigenpairs are computed for matrices of order 1,000,000!

We concentrate on the symmetric eigenvalue problem since fortunately many of the very large eigenvalue problems that arise are symmetric. We first consider the general problem of obtaining approximations from a subspace of $\mathbf{R}^n$. We then survey the two main classes of methods developed for large or very large eigenvalue problems.

Let $\mathcal{S}$ be the subspace of $\mathbf{R}^n$ spanned by the columns of a given matrix $S = (s_1, \dots, s_m) \in \mathbf{R}^{n\times m}$ (usually $m \ll n$). We consider here the problem of finding the best set of approximate eigenvectors in $\mathcal{S}$ to eigenvectors of a Hermitian matrix A. The following generalization of the Rayleigh quotient is the essential tool needed.

Theorem 9.7.1.

Let A be Hermitian and $Q \in \mathbf{R}^{n\times p}$ be orthonormal, $Q^H Q = I_p$. Then the residual norm $\|AQ - QC\|_2$ is minimized for $C = M$ where

$$M = \rho(Q) = Q^H A Q \tag{9.7.1}$$

is the corresponding Rayleigh quotient matrix. Further, if $\theta_1, \dots, \theta_p$ are the eigenvalues of M, there are p eigenvalues $\lambda_{i1}, \dots, \lambda_{ip}$ of A, such that

$$|\lambda_{ij} - \theta_j| \le \|AQ - QM\|_2, \quad j = 1 : p. \tag{9.7.2}$$

Proof. See Parlett [305, Sec. 11-5]. □

We can now outline the complete procedure:

Algorithm 9.5.

The Rayleigh–Ritz procedure

1. Determine an orthonormal matrix $Q = (q_1, \dots, q_m)$ such that $\mathcal{R}(Q) = \mathcal{S}$.
2. Form the matrix $B = AQ = (Aq_1, \dots, Aq_m)$ and the generalized Rayleigh quotient matrix
$$M = Q^H (AQ) \in \mathbf{R}^{m\times m}. \tag{9.7.3}$$

3. Compute the $p \le m$ eigenpairs of the Hermitian matrix M which are of interest
$$Mz_i = \theta_1 z_i, \qquad i = 1 : p. \tag{9.7.4}$$
The eigenvectors can be chosen such that $Z = (z_1, \dots, z_m)$ is a unitary matrix. The eigenvalues θ_i are the **Ritz values**, and the vectors $y_i = Qz_i$ the **Ritz vectors**.

4. Compute the residual matrix $R = (r_1, \dots, r_p)$, where
$$r_i = Ay_i - y_i\theta_i = (AQ)z_i - y_i\theta_i. \tag{9.7.5}$$
Then each interval
$$\big[\theta_i - \|r_i\|_2, \theta_i + \|r_i\|_2\big], \qquad i = 1 : p, \tag{9.7.6}$$
contains an eigenvalue λ_i of A.

The pairs (θ_i, y_i), $i = 1 : p$ are the best approximate eigenpairs of A which can be derived from the space $\mathcal{S}$. If some of the intervals in (9.7.6) overlap, we cannot be sure to have approximations to p eigenvalues of A. However, there are always p eigenvalues in the intervals defined by (9.7.2).

We can get error bounds for the approximate eigenspaces from an elegant generalization of Theorem 9.2.17. We first need to define the **gap** of the spectrum of A with respect to a given set of approximate eigenvalues.

Definition 9.7.2.

Let $\lambda(A) = \{\lambda_1, \dots, \lambda_n\}$ be eigenvalues of a Hermitian matrix A. For the set $\rho = \{\theta_1, \dots, \theta_p\}$, let $s_\rho = \{\lambda_{i_1}, \dots, \lambda_{i_p}\}$ be a subset of $\lambda(A)$ minimizing $\max_j |\theta_j - \lambda_{i_j}|$. Then we define
$$\operatorname{gap}(\rho) = \min_{\lambda \in \lambda(A)} |\lambda - \theta_i|, \quad \lambda \notin s_\rho, \quad \theta_i \in \rho. \tag{9.7.7}$$

Theorem 9.7.3.

Let $Q \in \mathbf{R}^{n\times p}$ be orthonormal and A a Hermitian matrix. Let $\{\theta_1, \dots, \theta_p\}$ be the eigenvalues of $H = \rho(Q) = Q^H AQ$, and let $s_r = \{\lambda_{i_1}, \dots, \lambda_{i_p}\}$ be a subset of eigenvalues of A such that $\max_j |\theta_j - \lambda_{i_j}|$ is minimized. If $\mathcal{Z}$ is the invariant subspace of A corresponding to s_r, then
$$\theta(\mathcal{Q}, \mathcal{Z}) \le \|AQ - QH\|_2 / \operatorname{gap}(\rho). \tag{9.7.8}$$
where $\sin\theta(\mathcal{Q}, \mathcal{Z})$ is the largest angle between the subspaces $\mathcal{Q}$ and $\mathcal{Z}$.

9.7.1 Subspace Iteration for Hermitian Matrices

In Sec. 9.3.4 subspace iteration, or orthogonal iteration, was introduced as a block version of the power method. Subspace iteration has long been one of the most important methods for solving large sparse eigenvalue problems. In particular it has been used much in structural engineering, and developed to a high standard of refinement.

In simple subspace iteration we start with an initial matrix $Q_0 \in \mathbf{R}^{n\times p}$ ($1 < p \ll n$) with orthogonal columns. From this a sequence of matrices $\{Q_k\}$ are computed from

$$Z_k = AQ_{k-1}, \qquad Q_k R_k = Z_k, \qquad k = 1, 2, \dots, \tag{9.7.9}$$

where $Q_k R_k$ is the QR decomposition of the matrix Z_k. There is no need for the matrix A to be known explicitly; only an algorithm (subroutine) for computing the matrix-vector product Aq for an arbitrary vector q is required. This iteration (9.7.9) generates a sequence of subspaces $\mathcal{S}_k = \mathcal{R}(A^k Q_0) = \mathcal{R}(Q_k)$, and we seek approximate eigenvectors of A in these subspaces. It can be shown (see Sec. 9.3.4) that if A has p dominant eigenvalues $\lambda_1, \cdots, \lambda_p$, i.e.,

$$|\lambda_1| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n|$$

then the subspaces $\mathcal{S}_k$, $k = 0, 1, 2, \dots$ converge almost always to the corresponding dominating invariant subspace.The convergence is linear with rate $|\lambda_{p+1}/\lambda_p|$.

For the individual eigenvalues $\lambda_i > \lambda_{i+1}$, $i \leq p$, it holds that

$$|r_{ii}^{(k)} - \lambda_i| = O(|\lambda_{i+1}/\lambda_i|^k), \qquad i = 1 : p.$$

where $r_{ii}^{(k)}$ are the diagonal elements in R_k. This rate of convergence is often unacceptably slow. We can improve this by including the Rayleigh–Ritz procedure in orthogonal iteration. For the real symmetric (Hermitian) case this leads to the improved algorithm below.

Algorithm 9.6.
Orthogonal Iteration, Hermitian Case.
With $Q_0 \in \mathbf{R}^{n\times p}$ compute for $k = 1, 2, \dots$ a sequence of matrices Q_k as follows:

1. Compute $Z_k = AQ_{k-1}$;
2. Compute the QR decomposition $Z_k = \bar{Q}_k R_k$;
3. Form the (matrix) Rayleigh quotient $B_k = \bar{Q}_k^T (A\bar{Q}_k)$;
4. Compute eigenvalue decomposition $B_k = U_k \Theta_k U_k^T$;
5. Compute the matrix of Ritz vectors $Q_k = \bar{Q}_k U_k$.

It can be shown that

$$|\theta_i^{(k)} - \lambda_i| = \mathrm{O}(|\lambda_{p+1}/\lambda_i|^k), \qquad \Theta_k = \mathrm{diag}\,(\theta_1^{(k)}, \dots, \theta_p^{(k)}),$$

which is a much more favorable rate of convergence than without the Rayleigh–Ritz procedure. The columns of Q_k are the Ritz vectors, and they will converge to the corresponding eigenvectors of A.

Example 9.7.1.

Let A have the eigenvalues $\lambda_1 = 100$, $\lambda_2 = 99$, $\lambda_3 = 98$ $\lambda_4 = 10$, and $\lambda_5 = 5$. With $p = 3$ the asymptotic convergence ratios for the jth eigenvalue with and without Rayleigh–Ritz acceleration are:

j	without R-R	with R-R
1	0.99	0.1
2	0.99	0.101
3	0.102	0.102

The work in step 1 of Algorithm 9.7.1 consists of p matrix times vector operations with the matrix A. If the modified Gram-Schmidt method is used step 2 requires $p(p+1)n$ flops. To form the Rayleigh quotient matrix requires a further p matrix times vector multiplications and $p(p+1)n/2$ flops, taking the symmetry of B_k into account. Finally, steps 4 and 5 take about $5p^3$ and p^2n flops, respectively.

Note that the same subspace $\mathcal{S}_k$ is generated by k consecutive steps of 1, as with the complete Algorithm 9.7.1. Therefore, the rather costly orthogonalization and Rayleigh–Ritz acceleration need not be carried out at every step. However, to be able to check convergence to the individual eigenvalues we need the Rayleigh–Ritz approximations. If we then form the residual vectors

$$r_i = Aq_i^{(k)} - q_i^{(k)}\theta_i = (AQ_k)u_i^{(k)} - q_i^{(k)}\theta_i. \tag{9.7.10}$$

and compute $\|r_i\|_2$ each interval $[\theta_i - \|r_i\|_2, \theta_i + \|r_i\|_2]$ will contain an eigenvalue of A. Sophisticated versions of subspace iteration have been developed. A highlight is the Contribution II/9 by Rutishauser in [326].

Algorithm 9.7.1 can be generalized to nonsymmetric matrices, by substituting in step 4 the Schur decomposition

$$B_k = U_k S_k U_k^T,$$

where S_k is upper triangular. The vectors q_i then converge to the Schur vector u_i of A.

If interior eigenvalues are wanted then we can consider the **spectral transformation** (see Sec. 9.3.2)

$$\hat{A} = (A - \mu I)^{-1}.$$

The eigenvalues of $\hat{A}$ and A are related through $\hat{\lambda}_i = 1/(\lambda_i - \mu)$. Hence, the eigenvalues λ in a neighborhood of μ will correspond to outer eigenvalues of $\hat{A}$, and can be determined by applying subspace iteration to $\hat{A}$. To perform the multiplication $\hat{A}q$ we need to be able to solve systems of equations of the form

$$(A - \mu I)p = q. \tag{9.7.11}$$

This can be done, e.g., by first computing an LU factorization of $A - \mu I$ or by an iterative method.

9.7.2 Krylov Subspaces

Many methods for the solving the eigenvalue problem developed by Krylov and others in the 1930's and 40's aimed at bringing the characteristic equation into polynomial form. Although this in general is a bad idea, we will consider one approach, which is of interest because of its connection with Krylov subspace methods and the **Lanczos process**.

Throughout this section we assume that $A \in \mathbf{R}^{n\times n}$ is a real symmetric matrix. Associated with A is the characteristic polynomial (9.1.4)

$$p(\lambda) = (-1)^n(\lambda^n - \xi_{n-1}\lambda^{n-1} - \cdots \xi_0) = 0.$$

The Cayley–Hamilton theorem states that $p(A) = 0$, that is

$$A^n = \xi_{n-1}A^{n-1} + \cdots \xi_1 A + \xi_0. \tag{9.7.12}$$

In particular, we have

$$\begin{aligned} A^n v &= \xi_{n-1}A^{n-1}v + \cdots \xi_1 Av + \xi_0 v \\ &= [v, Av, \dots, A^{n-1}v]x, \end{aligned}$$

where $x = (\xi_0, \xi_1, \dots, \xi_{n-1})^T$.

Consider the **Krylov sequence** of vectors, $v_0 = v$,

$$v_{j+1} = Av_j, \quad j = 0 : n-1. \tag{9.7.13}$$

We assume in the following that v is chosen so that $v_i \neq 0$, $i = 0 : n-1$, Then we may write (9.7.13) as

$$xBx = v_n, \quad B = [v_0, v_1, \dots, v_{n+1}] \tag{9.7.14}$$

which is a linear equations in n unknowns.

Multiplying (9.7.14) on the left with B^T we obtain a symmetric linear system, the normal equations

$$Mx = z, \quad M = B^T B, \quad z = B^T v_n.$$

The elements m_{ij} of the matrix M are

$$m_{i+1,j+1} = v_i^T v_j = (A^i v)^T A^j v = v^T A^{i+j} v.$$

They only depend on the sum of the indices and we write

$$m_{i+1,j+1} = \mu_{i+j}, \quad i + j = 0; 2n-1.$$

Unfortunately this system tends to be very ill-conditioned. For larger values of n the Krylov vectors soon become parallel to the eigenvector associated with the dominant eigenvalue.

The Krylov subspace $\mathcal{K}_m(v, A)$ depends on both A and v. However, it is important to note the following simply verified invariance properties:

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

These invariance can be used to deduce some important properties of methods using Krylov subspaces. Since A and $-A$ generate the same subspaces the left and right part of the spectrum of A are equally approximated. The invariance with respect to shifting shows, e.g, that it does not matter if A is positive definite or not.

We note that the Krylov subspace $\mathcal{K}(v, A)$ is spanned by the vectors generated by performing $k-1$ steps of the power method starting with v. However, in the power method we throw away previous vectors and just use the last vector $A^k v$ to get an approximate eigenvector. It turns out that this is wasteful and that much more powerful methods can be developed which work with the complete Krylov subspace.

Any vector $x \in \mathcal{K}_m(v)$ can be written in the form

$$x = \sum_{i=0}^{m-1} c_i A^i v = P_{m-1}(A) v,$$

where P_{m-1} is a polynomial of degree less than m. This provides a link between polynomial approximation and Krylov type methods, the importance of which will become clear in the following.

A fundamental question is: How well can an eigenvector of A be approximated by a vector in $\mathcal{K}(v, A)$? Let Π_k denote the orthogonal projector onto the Krylov subspace $\mathcal{K}(v, A)$. The following lemma bounds the distance $\|u_i - \Pi_k u_i\|_2$, where u_i is a particular eigenvector of A.

Theorem 9.7.4.

Assume that A is diagonalizable and let the initial vector v have the expansion

$$v = \sum_{k=1}^{n} \alpha_k u_k \tag{9.7.15}$$

in terms of the normalized eigenvectors $u_1, \ldots, u_n$. Let P_{k-1} be the set of polynomials of degree at most $k-1$ such that $p(\lambda_i) = 1$. Then, if $\alpha_i \neq 0$ the following inequality holds:

$$\|u_i - \Pi_k u_i\|_2 \leq \xi_i \epsilon_i^{(k)}, \quad \xi_i = \sum_{j \neq i} |\alpha_j| / |\alpha_i|, \tag{9.7.16}$$

where

$$\epsilon_i^{(k)} = \min_{p \in P_{k-1}} \max_{\lambda \in \lambda(A) - \lambda_i} |p(\lambda)|. \tag{9.7.17}$$

Proof. We note that any vector in $\mathcal{K}_k$ can be written $q(A)v$, where q is a polynomial $q \in P_{k-1}$. Since Π_k is the orthogonal projector onto $\mathcal{K}_k$ we have

$$\|(I - \Pi_k) u_i\|_2 \leq \|u_i - q(A) v\|_2.$$

Using the expansion (9.7.15) of v it follows that for any polynomial $p \in P_{k-1}$ with $p(\lambda_i) = 1$ we have

$$\|(I - \Pi_k)\alpha_i u_i\|_2 \le \Big\|\alpha_i u_i - \sum_{j=1}^{n} \alpha_j p(\lambda_j) u_j\Big\|_2 \le \max_{j\neq i} |p(\lambda_j)| \Big|\sum_{j\neq i} |\alpha_j|\Big|.$$

The last inequality follows noticing that the component in the eigenvector u_i is zero and using the triangle inequality. Finally, dividing by $|\alpha_i|$ establishes the result. $\square$

To obtain error bounds we use the properties of the Chebyshev polynomials. We now consider the Hermitian case and assume that the eigenvalues of A are simple and ordered so that $\lambda_1 > \lambda_2 > \cdots > \lambda_n$. Let $T_k(x)$ be the Chebyshev polynomial of the first kind of degree k. Then $|T_k(x)| \le 1$ for $|x| \le 1$, and for $|x| \ge 1$ we have

$$T_k(x) = \frac{1}{2}\left[(x + \sqrt{x^2 - 1})^k + (x - \sqrt{x^2 - 1})^k\right]. \tag{9.7.18}$$

Now if we take

$$x = l_i(\lambda) = 1 + 2\frac{\lambda - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}, \quad \gamma_i = l_i(\lambda_i) = 1 + 2\frac{\lambda_i - \lambda_{i+1}}{\lambda_i - \lambda_n}. \tag{9.7.19}$$

the interval $\lambda = [\lambda_{i+1}, \lambda_n]$ is mapped onto $x = [-1, 1]$, and $\gamma_1 > 1$. In particular, for $i = 1$, we take

$$p(\lambda) = \frac{T_{k-1}(l_1(\lambda))}{T_{k-1}(\gamma_1)}.$$

Then $p(\lambda_1) = 1$ as required by Theorem 9.7.4. When k is large we have

$$\epsilon_1^{(k)} \le \max_{\lambda \in \lambda(A) - \lambda_i} |p(\lambda)| \le \frac{1}{T_{k-1}(\gamma_1)} \approx 2\Big/ \left(\gamma_1 + \sqrt{\gamma_1^2 - 1}\right)^{k-1}. \tag{9.7.20}$$

The steep climb of the Chebyshev polynomials outside the interval $[-1, 1]$ explains the powerful approximation properties of the Krylov subspaces. The approximation error tends to zero with a rate depending on the gap $\lambda_1 - \lambda_2$ normalized by the spread of the rest of the eigenvalues $\lambda_2 - \lambda_n$. Note that this has the correct form with respect to the invariance properties of the Krylov subspaces.

By considering the matrix $-A$ we get analogous convergence results for the rightmost eigenvalue λ_n of A. In general, for $i > 1$, similar but weaker results can be proved using polynomials of the form

$$p(\lambda) = q_{i-1}(\lambda)\frac{T_{k-i}(l_i(\lambda))}{T_{k-i}(\gamma_i)}, \quad q_{i-1}(\lambda) = \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda}{\lambda_j - \lambda_i}.$$

Notice that $q_{i-1}(\lambda)$ is a polynomial of degree $i - 1$ with $q_{i-1}(\lambda_j) = 0$, $j = 1 : i - 1$, and $q_{i-1}(\lambda_i) = 1$. Further,

$$\max_{\lambda \in \lambda(A) - \lambda_i} |q_{i-1}(\lambda)| \le |q_{i-1}(\lambda_n)| = C_i. \tag{9.7.21}$$

Thus, when k is large we have

$$\epsilon_i^{(k)} \leq C_i / T_{k-i}(\gamma_i). \tag{9.7.22}$$

This indicates that we can expect interior eigenvalues and eigenvectors to be less well approximated by Krylov-type methods.

9.7.3 The Lanczos Process

We will now show that the Rayleigh–Ritz procedure can be applied to the sequence of Krylov subspaces $\mathcal{K}_m(v)$, $m = 1, 2, 3, \ldots$, in a very efficient way using the **Lanczos process**. The Lanczos process, developed by Lanczos [250, 1950], can be viewed as a way for reducing a symmetric matrix A to tridiagonal form $T = Q^T AQ$. Here $Q = (q_1, q_2, \ldots, q_n)$ is orthogonal, where q_1 can be chosen arbitrarily, and

$$T = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}. \tag{9.7.23}$$

is symmetric tridiagonal.

Equating the first $n-1$ columns in $A(q_1, q_2, \ldots, q_n) = (q_1, q_2, \ldots, q_n)T$ gives

$$Aq_j = \beta_j q_{j-1} + \alpha_j q_j + \beta_{j+1} q_{j+1}, \quad j = 1 : n-1.$$

where we have put $\beta_1 q_0 \equiv 0$. The requirement that $q_{j+1} \perp q_j$ gives

$$\alpha_j = q_j^T (Aq_j - \beta_j q_{j-1}),$$

(Note that since $q_j \perp q_{j-1}$ the last term could in theory be dropped; however, since a loss of orthogonality occurs in practice it should be kept. This corresponds to using the modified rather than the classical Gram-Schmidt orthogonalization process.)

Further, solving for q_{j+1},

$$\beta_{j+1} q_{j+1} = r_{j+1}, \qquad r_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1},$$

so if $r_{j+1} \neq 0$, then β_{j+1} and q_{j+1} is obtained by normalizing r_{j+1}. Given q_1 these equations can be used recursively to compute the elements in the tridiagonal matrix T and the orthogonal matrix Q.

Algorithm 9.7.
The Lanczos Process.
Let A be a symmetric matrix and $q_1 \neq 0$ a given vector. The following algorithm computes in exact arithmetic after k steps a symmetric tridiagonal matrix $T_k = \text{trid}\,(\beta_j, \alpha_j, \beta_{j+1})$ and a matrix $Q_k = (q_1, \ldots, q_k)$ with orthogonal columns spanning

the Krylov subspace $\mathcal{K}_k(q_1, A)$:

$$
\begin{aligned}
& q_0 = 0; \quad r_0 = q_1; \\
& \beta_1 = 1; \\
& \textbf{for } j = 1, 2, 3 \ldots \\
& \qquad q_j = r_{j-1}/\beta_j; \\
& \qquad r_j = Aq_j - \beta_j q_{j-1}; \\
& \qquad \alpha_j = q_j^T r_j; \\
& \qquad r_j = r_j - \alpha_j q_j; \\
& \qquad \beta_{j+1} = \|r_j\|_2; \\
& \qquad \textbf{if } \beta_{j+1} = 0 \textbf{ then } exit; \\
& \textbf{end}
\end{aligned}
$$

Note that A only occurs in the matrix-vector operation Aq_j. Hence, the matrix A need not be explicitly available, and can be represented by a subroutine. Only three n-vectors are needed in storage.

It is easy to see that if the Lanczos algorithm can be carried out for k steps then it holds

$$AQ_k = Q_k T_k + \beta_{k+1} q_{k+1} e_k^T. \tag{9.7.24}$$

The Lanczos process stops if $\beta_{k+1} = 0$ since then q_{k+1} is not defined. However, then by (9.7.24) it holds that $AQ_k = Q_k T_k$, and thus Q_k spans an invariant subspace of A. This means that the eigenvalues of T_k also are eigenvalues of A. (For example, if q_1 happens to be an eigenvector of A, the process stops after one step.) Further, eigenvalues of A can the be determined by restarting the Lanczos process with a vector orthogonal to $q_1, \ldots, q_k$.

By construction it follows that $\operatorname{span}(Q_k) = \mathcal{K}_k(A, b)$. Multiplying (9.7.24) by Q_k^T and using $Q_k^T q_{k+1} = 0$ it follows that $T_k = Q_k^T A Q_k$, and hence T_k is the generalized Rayleigh quotient matrix corresponding to $\mathcal{K}_k(A, b)$. The Ritz values are the eigenvalues θ_i of T_k, and the Ritz vectors are $y_i = Q_k z_i$, where z_i are the eigenvectors of T_k corresponding to θ_i.

In principle we could at each step compute the Ritz values θ_i and Ritz vectors y_i, $i = 1 : k$. Then the accuracy of the eigenvalue approximations could be assessed from the residual norms $\|Ay_i - \theta_i y_i\|_2$, and used to decide if the process should be stopped. However, this is not necessary since using (9.7.24) we have

$$Ay_i - y_i\theta_i = AQ_k z_i - Q_k z_i \theta_i = (AQ_k - Q_k T_k) z_i = \beta_{k+1} q_{k+1} e_k^T z_i.$$

Taking norms we get

$$\|Ay_i - y_i\theta_i\|_2 = \beta_{k+1} |e_k^T z_i|. \tag{9.7.25}$$

i.e., we can compute the residual norm just from the bottom element of the normalized eigenvectors of T_k. This is fortunate since then we need to access the Q matrix only after the process has converged. The vectors can be stored on secondary storage, or often better, regenerated at the end. The result (9.7.25) also explains why some Ritz values can be very accurate approximations even when β_{k+1} is not small.

So far we have discussed the Lanczos process in exact arithmetic. In practice, roundoff will cause the generated vectors to lose orthogonality. A possible remedy is to reorthogonalize each generated vector q_{k+1} to all previous vectors $q_k, \dots, q_1$. This is however very costly both in terms of storage and operations.

A satisfactory analysis of the numerical properties of the Lanczos process was first given by C. C. Paige [298, 1971]. He showed that it could be very effective in computing accurate approximations to a few of the extreme eigenvalues of A even in the face of total loss of orthogonality! The key to the behaviour is, that at the same time as orthogonality is lost, a Ritz pair converges to an eigenpair of A. As the algorithm proceeds it will soon start to converge to a second copy of the already converged eigenvalue, and so on. The effect of finite precision is to slow down convergence, but does not prevent accurate approximations to be found!

The Lanczos process is also the basis for several methods for solving large scale symmetric linear systems, and least squares problems, see Section 10.4.

9.7.4 Golub–Kahan–Lanczos Bidiagonalization

A Lanczos process can also be developed for computing singular values and singular vectors to a rectangular matrix A. For this purpose we consider here the Golub–Kahan bidiagonalization (GKBD) of a matrix $A \in \mathbf{R}^{m\times n}$, $m \geq n$. This has important applications for computing approximations to the large singular values and corresponding singular vectors, as well as for solving large scale least squares problems.

In Section 8.4.8 we gave an algorithm for computing the decomposition

$$A = U \begin{pmatrix} B \\ 0 \end{pmatrix} V^T, \quad U^T U = I_m, \quad V^T V = I_n, \tag{9.7.26}$$

where $U = (u_1, \dots, u_m)$ and $V = (v_1, \dots, v_n)$ are chosen as products of Householder transformations and B is upper bidiagonal. If we set $U_1 = (u_1, \dots, u_n)$ then from (9.7.26) we have

$$AV = U_1 B, \qquad A^T U_1 = V B^T. \tag{9.7.27}$$

In an alternative approach, given by Golub and Kahan [173, 1965], the columns of U and V are generated sequentially, as in the Lanczos process.

A more useful variant of this bidiagonalization algorithm is obtained by instead taking transforming A into **lower** bidiagonal form

$$B_n = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix} \in \mathbf{R}^{(n+1)\times n}. \tag{9.7.28}$$

(Note that B_n is not square.) Equating columns in (9.7.27) we obtain, setting $\beta_1 v_0 \equiv 0$, $\alpha_{n+1}v_{n+1} \equiv 0$, the recurrence relations

$$\begin{aligned} A^T u_j &= \beta_j v_{j-1} + \alpha_j v_j, \\ A v_j &= \alpha_j u_j + \beta_{j+1} u_{j+1}, \quad j = 1:n. \end{aligned} \tag{9.7.29}$$

Starting with a given vector $u_1 \in \mathbf{R}^m$, $\|u_1\|_2 = 1$, we can now recursively generate the vectors $v_1, u_2, v_2, \dots, u_{m+1}$ and corresponding elements in B_n using, for $j = 1, 2, \dots$, the formulas

$$r_j = A^T u_j - \beta_j v_{j-1}, \quad \alpha_j = \|r_j\|_2, \quad v_j = r_j/\alpha_j, \tag{9.7.30}$$

$$p_j = A v_j - \alpha_j u_j, \qquad \beta_{j+1} = \|p_j\|_2, \quad u_{j+1} = p_j/\beta_{j+1}. \tag{9.7.31}$$

For this bidiagonalization scheme we have

$$u_j \in \mathcal{K}_j(AA^T, u_1), \qquad v_j \in \mathcal{K}_j(A^T A, A^T u_1).$$

There is a close relationship between the above bidiagonalization process and the Lanczos process applied to the two matrices AA^T and $A^T A$. Note that these matrices have the same nonzero eigenvalues σ_i^2, $i = 1:n$, and that the corresponding eigenvectors equal the left and right singular vectors of A, respectively.

The GKBD process (9.7.30)–(9.7.31) generates in exact arithmetic the same sequences of vectors $u_1, u_2, \dots$ and $v_1, v_2, \dots$ as are obtained by simultaneously applying the Lanczos process to AA^T with starting vector $u_1 = b/\|b\|_2$, and to $A^T A$ with starting vector $v_1 = A^T b/\|A^T b\|_2$.

In floating point arithmetic the computed Lanczos vectors will lose orthogonality. In spite of this the extreme (largest and smallest) singular values of the truncated bidiagonal matrix $B_k \in \mathbf{R}^{(k+1)\times k}$ tend to be quite good approximations to the corresponding singular values of A, even for $k \ll n$. Let the singular value decomposition of B_k be $B_k = P_{k+1}\Omega_k Q_k^T$. Then approximations to the singular vectors of A are

$$\hat{U}_k = U_k P_{k+1}, \qquad \hat{V}_k = V_k Q_k.$$

This is a simple way of realizing the Ritz–Galerkin projection process on the subspaces $\mathcal{K}_j(A^T A, v_1)$ and $\mathcal{K}_j(AA^T, Av_1)$. The corresponding approximations are called Ritz values and Ritz vectors.

Lanczos algorithms for computing selected singular values and vectors have been developed, which have been used, e.g., in information retrieval problems and in seismic tomography. In these applications typically, the 100–200 largest singular values and vectors for matrices having up to 30,000 rows and 20,000 columns are required.

9.7.5 Arnoldi's Method

Of great importance for iterative methods are the subspaces of the form

$$\mathcal{K}_m(A, v) = \operatorname{span}\bigl(v, Av, \dots, A^{m-1}v\bigr), \tag{9.7.32}$$

generated by a matrix A and a single vector v. These are called **Krylov subspaces**[44] and the corresponding matrix

$$K_m(A,v) = \big(v, Av, \dots, A^{m-1}v\big)$$

is called a Krylov matrix. If $m \le n$ the dimension of $\mathcal{K}_m$ usually equals m unless v is specially related to A.

and a related square Hessenberg matrix $H_k = (h_{ij}) \in \mathbf{R}^{k\times k}$. Further, we have

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1}\bar{H}_k, \tag{9.7.33}$$

where

$$\bar{H}_k = \begin{pmatrix} H_k \\ h_{k+1,k}e_k^T \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ & \ddots & \ddots & \vdots \\ & & h_{k,k-1} & h_{kk} \\ & & & h_{k+1,k} \end{pmatrix} \in \mathbf{R}^{(k+1)\times k}. \tag{9.7.34}$$

Arnoldi's method is an orthogonal projection method onto Krylov subspace $\mathcal{K}_m$ for general non Hermitian matrices. The procedure starts by building an orthogonal basis for $\mathcal{K}_m$

Algorithm 9.8.
The Arnoldi process.
Let A be a matrix and v_1, $\|v_1\|_2 = 1$, a given vector. The following algorithm computes in exact arithmetic after k steps a Hessenberg matrix $H_k = (h_{ij})$ and a matrix $V_k = (v_1, \dots, v_k)$ with orthogonal columns spanning the Krylov subspace $\mathcal{K}_k(v_1, A)$:

for $j = 1 : k$
  for $i = 1 : j$
    $h_{ij} = v_i^H (Av_j);$
  end
  $r_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i;$
  $h_{j+1,j} = \|r_j\|_2;$
  if $h_{j+1,j} = 0$ **then** *exit*;
  $v_{j+1} = r_j / h_{j+1,j};$
end

[44]Named after Aleksei Nikolaevich Krylov (1877–1945) Russian mathematician. Krylov worked at the Naval Academy in Saint-Petersburg and in 1931 published a paper [246] on what is now called "Krylov subspaces".

The Hessenberg matrix $H_k \in \mathbf{C}^{k\times k}$ and the unitary matrix V_k computed in the Arnoldi process satisfy the relations

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^H, \tag{9.7.35}$$
$$V_k^H AV_k = H_k. \tag{9.7.36}$$

The process will break down at step j if and only if the vector r_j vanishes. When this happens we have $AV_k = V_k H_k$, and so $\mathcal{R}(V_k)$ is an invariant subspace of A. By (9.7.35) $H_k = V_k^H AV_k$ and thus the Ritz values and Ritz vectors are obtained from the eigenvalues and eigenvectors of H_k. The residual norms can be inexpensively obtained as follows (cf. (9.7.25))

$$\|(A - \theta_i I)y_i\|_2 = h_{m+1,m}|e_k^T z_i|. \tag{9.7.37}$$

The proof of this relation is left as an exercise.

Review Questions

7.1 Tell the names of two algorithms for (sparse) symmetric eigenvalue problems, where the matrix A need not to be explicitly available but only as a subroutine for the calculation of Aq for an arbitrary vector q. Describe one of the algorithms.

7.2 Name two algorithms for (sparse) symmetric eigenvalue problems, where the matrix A need not to be explicitly available but only as a subroutine for the calculation of Aq for an arbitrary vector q. Describe one of the algorithms in more detail.

Problems

7.1 (To be added.)

9.8 Generalized Eigenvalue Problems

The **generalized eigenvalue problem** is that of computing nontrivial solutions (λ, x) of

$$Ax = \lambda Bx, \tag{9.8.1}$$

where A and B are square matrices of order n.

The family of matrices $A - \lambda B$ is called a **matrix pencil**.[45] It is called a **regular** pencil if $\det(A - \lambda B)$ is not identically zero, else it is a **singular** pencil.

[45] The word "pencil" comes from optics and geometry, and is used for any one parameter family of curves, matrices, etc.

A simple example of a singular pencil is

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix},$$

where A and B have a null vector e_2 in common. If $A - \lambda B$ is a regular pencil, then the eigenvalues λ are the zeros of the characteristic equation

$$\det(A - \lambda B) = 0. \tag{9.8.2}$$

If the degree of the characteristic polynomial is $n - p$, then we say that $A - \lambda B$ has p eigenvalues at ∞.

Example 9.8.1.

The characteristic equation of the pencil

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

is $\det(A - \lambda B) = 1 - \lambda$ and has degree one. There is one eigenvalue $\lambda = \infty$ corresponding to the eigenvector e_1. Note that infinite eigenvalues of $A - \lambda B$ simply correspond to the zero eigenvalues of the pencil $B - \lambda A$.

For the ordinary eigenvalue problem eigenvalues are preserved under similarity transformations. The corresponding transformations for the generalized eigenvalue problem are called **equivalence transformations**.

Definition 9.8.1.

Let (A, B) be a matrix pencil and let S and T be nonsingular matrices. Then the two pencils $A - \lambda B$ and $SAT - \lambda SBT$ are said to be equivalent.

That equivalent pencils have the same eigenvalues follows from (9.8.2) since

$$\det S(A - \lambda B)T = \det(SAT - \lambda SBT) = 0.$$

Further, the eigenvectors are simply related.

If A and B are real symmetric, then symmetry is preserved under congruence transformations in which $T = S^T$. The two pencils are then said to be **congruent**. Of particular interest are orthogonal congruence transformations, $S = Q^T$ and $T = Q$, where U is a unitary. Such transformations are stable since they preserve the 2-norm,

$$\|Q^T AQ\|_2 = \|A\|_2, \qquad \|Q^T BQ|_2 = \|B\|_2.$$

9.8.1 Canonical Forms

The algebraic and analytic theory of the generalized eigenvalue problem is more complicated than for the standard problem. There is a canonical form for regular matrix pencils corresponding to the Schur form and the Jordan canonical form, Theorem 9.1.9, which we state without proof.

Theorem 9.8.2 (*Kronecker's Canonical Form*).

Let $A - \lambda B \in \mathbf{C}^{n\times n}$ be a regular matrix pencil. Then there are nonsingular matrices $X, Z \in \mathbf{C}^{n\times n}$, such that $X^{-1}(A - \lambda B)Z = \hat{A} - \lambda\hat{B}$, where

$$\hat{A} = \mathrm{diag}\,(J_{m_1}(\lambda_1), \ldots, J_{m_s}(\lambda_s), I_{m_{s+1}}, \ldots, I_{m_t}), \tag{9.8.3}$$
$$\hat{B} = \mathrm{diag}\,(I_{m_1}, \ldots, I_{m_s}, J_{m_{s+1}}(0), \ldots, J_{m_t}(0)),$$

and where $J_{m_i}(\lambda_i)$ are Jordan blocks and the blocks $s+1, \ldots, t$ correspond to infinite eigenvalues. The numbers $m_1, \ldots, m_t$ are unique and $\sum_{i=1}^{t} m_i = n$.

The disadvantage with the Kronecker Canonical Form is that it depends discontinuously on A and B and is unstable (**needs clarifying??**). There is also a generalization of the Schur Canonical Form (Theorem 9.1.11), which can be computed stably and more efficiently.

Theorem 9.8.3. Generalized Schur Canonical Form.

Let $A - \lambda B \in \mathbf{C}^{n\times n}$ be a regular matrix pencil. Then there exist unitary matrices U and V so that

$$UAV = T_A, \qquad UBV = T_B,$$

where both T_A and T_B are upper triangular. The eigenvalues of the pencil are the ratios of the diagonal elements of T_A and T_B.

Proof. See Stewart [340, Chapter 7.6]. □

As for the standard case, when A and B are real, then U and V can be chosen real and orthogonal if T_A and T_B are allowed to have 2×2 diagonal blocks corresponding to complex conjugate eigenvalues.

9.8.2 Reduction to Standard Form

When B is nonsingular the eigenvalue problem (9.8.1) is formally equivalent to the standard eigenvalue problem $B^{-1}Ax = \lambda x$. However, when B is singular such a reduction is not possible. Also, if B is close to a singular matrix, then we can expect to lose accuracy in forming $B^{-1}A$.

Of particular interest is the case when the problem can be reduced to a symmetric eigenvalue problem of standard form. A surprising fact is that any real square matrix F can be written as $F = AB^{-1}$ or $F = B^{-1}A$ where A and B are suitable symmetric matrices. For a proof see Parlett [305, Sec. 15-2] (cf. also Problem 1). Hence, even if A and B are symmetric the generalized eigenvalue problems embodies all the difficulties of the unsymmetric standard eigenvalue problem. However, if B is also positive definite, then the problem (9.8.1) can be reduced to a standard symmetric eigenvalue problem. This reduction is equivalent to the simultaneous transformation of the two quadratic forms x^TAx and x^TBx to diagonal form.

Theorem 9.8.4.
Let A and B be real symmetric square matrices and B also positive definite. Then there exists a nonsingular matrix X such that

$$X^T A X = D_A, \qquad X^T B X = D_B \tag{9.8.4}$$

are real and diagonal. The eigenvalues of $A - \lambda B$ are given by

$$\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n) = D_A D_B^{-1}.$$

Proof. Let $B = LL^T$ be the Cholesky factorization of B. Then

$$L^{-1}(A - \lambda B)L^{-T} = \tilde{A} - \lambda I, \quad \tilde{A} = \tilde{A} = L^{-1} A L^{-T}, \tag{9.8.5}$$

where $\tilde{A}$ is real and symmetric. Let $\tilde{A} = Q^T D_A Q$ be the eigen-decomposition of $\tilde{A}$. Then we have

$$X^T (A - \lambda B) X = D_A - \lambda D_B, \quad X = (QL^{-1})^T,$$

and the theorem follows. $\square$

Given the pencil $A - \lambda B$ the pencil $\hat{A} - \lambda \hat{B} = \gamma A + \sigma B - \lambda(-\sigma A + \gamma B)$, where $\gamma^2 + \sigma^2 = 1$ has the same eigenvectors and the eigenvalues are related through

$$\lambda = (\gamma \hat{\lambda} + \sigma)/(-\sigma \hat{\lambda} + \gamma). \tag{9.8.6}$$

Hence, for the above reduction to be applicable, it suffices that some linear combination $-\sigma A + \gamma B$ is positive definite. It can be shown that if

$$\inf_{x \neq 0} \Big((x^T A x)^2 + (x^T B x)^2 \Big)^{1/2} > 0$$

then there exist such γ and σ. **Need to check definition of definite pair. Need complex x even for real A and B??**

Under the assumptions in Theorem 9.8.4 the symmetric pencil $A - \lambda B$ has n real roots. Moreover, the eigenvectors can be chosen to be B-orthogonal, i.e.,

$$x_i^T B x_j = 0, \quad i \neq j.$$

This generalizes the standard symmetric case for which $B = I$.

Numerical methods can be based on the *explicit reduction to standard form* in (9.8.5). $Ax = \lambda B x$ is then equivalent to $Cy = \lambda y$, where

$$C = L^{-1} A L^{-T}, \quad y = L^T x. \tag{9.8.7}$$

Computing the Cholesky decomposition $B = LL^T$ and forming $C = (L^{-1}A)L^{-T}$ takes about $5n^3/12$ flops if symmetry is used, see Wilkinson and Reinsch, Contribution II/10, [390]. If eigenvectors are not wanted, then the transform matrix L need not be saved.

If A and B are symmetric band matrices and $B = LL^T$ positive definite, then although L inherits the bandwidth of A the matrix $C = (L^{-1}A)L^{-T}$ will in general be a full matrix. Hence, in this case it may not be practical to form C. Crawford [79] has devised an algorithm for reduction to standard form which interleaves orthogonal transformations in such way that the matrix C retains the bandwidth of A, see Problem 2.

The round-off errors made in the reduction to standard form are in general such that they could be produced by small perturbations in A and B. **Not true??** see Davies et al [87, 355].)

When B is ill-conditioned then the eigenvalues λ may vary widely in magnitude, and a small perturbation in B can correspond to large perturbations in the eigenvalues. Surprisingly, well-conditioned eigenvalues are often given accurately in spite of the ill-conditioning of B. Typically L will have elements in its lower part. This will produce a matrix $(L^{-1}A)L^{-T}$ which is graded so that the large elements appear in the lower right corner. Hence, a reduction to tridiagonal form should work from bottom to top and the QL-algorithm should be used.

Example 9.8.2. *Wilkinson and Reinsch [390, p. 310].*

The matrix pencil $A - \lambda B$, where

$$A = \begin{pmatrix} 2 & 2 \\ 2 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 2 \\ 2 & 4.0001 \end{pmatrix},$$

has one eigenvalue ≈ -2 and one $\mathrm{O}(10^4)$. The true matrix

$$(L^{-1}A)L^{-T} = \begin{pmatrix} 2 & -200 \\ -200 & 10000 \end{pmatrix}$$

is graded, and the small eigenvalue is insensitive to relative perturbation in its elements.

9.8.3 Methods for Generalized Eigenvalue Problems

The special case when A and B are symmetric and B is positive definite can been treated by making use of the Cholesky factorization $B = LL^T$. Then $Ax = \lambda Bx$ implies

$$(L^{-1}AL^{-T})y = \lambda y, \quad y = L^T x, \tag{9.8.8}$$

which is a standard eigenvalue problem for $L^{-1}AL^{-T}$. There are a number of related problems involving A and B, which can also be reduced to standard form; see Martin and Wilkinson [273]. For example, $ABx = \lambda x$ is equivalent to

$$(L^T AL)y = \lambda y, \quad y = L^T x. \tag{9.8.9}$$

The power method and inverse iteration can both be extended to the generalized eigenvalue problems. Starting with some q_0 with $\|q_0\|_2 = 1$, these iterations now become

$$B\hat{q}_k = Aq_{k-1}, \quad q_k = \hat{q}_k/\|\hat{q}_k\|,$$
$$(A-\sigma B)\hat{q}_k = Bq_{k-1}, \qquad q_k = \hat{q}_k/\|\hat{q}_k\|, \qquad k = 1,2,\dots$$

respectively. Note that $B = I$ gives the iterations in equations (9.5.47) and (9.5.50).

The Rayleigh Quotient Iteration also extends to the generalized eigenvalue problem: For $k = 0,1,2,\dots$ compute

$$(A-\rho(q_{k-1})B)\hat{q}_k = Bq_{k-1}, \qquad q_k = \hat{q}_k/\|q_k\|_2, \tag{9.8.10}$$

where the (generalized) Rayleigh quotient of x is

$$\rho(x) = \frac{x^H Ax}{x^H Bx}.$$

In the symmetric definite case the Rayleigh Quotient Iteration has asymptotically cubic convergence and the residuals $\|(A-\mu_k B)x_k\|_{B^{-1}}$ decrease monotonically.

The Rayleigh Quotient method is advantageous to use when A and B have band structure, since it does not require an explicit reduction to standard form. The method of spectrum slicing can be used to count eigenvalues of $A-\lambda B$ in an interval.

Theorem 9.8.5.

Let $A-\sigma B$ have the Cholesky factorization

$$A-\mu B = LDL^T, \qquad D = \operatorname{diag}(d_1,\dots,d_n),$$

where L is unit lower triangular. If B is positive definite then the number of eigenvalues of A greater than μ equals the number of positive elements $\pi(D)$ in the sequence $d_1,\dots,d_n$.

Proof. The proof follows from Sylvester's Law of Inertia (Theorem 7.3.8) and the fact that by Theorem 9.8.2 A and B are congruent to D_A and D_B with $\Lambda = D_A D_B^{-1}$. □

9.8.4 The QZ Algorithm

The **QZ algorithm** by Moler and Stewart [282] is a generalization of the implicit QR algorithm. It does not preserve symmetry and is therefore more expensive than the special algorithms for the symmetric case.

In the first step of the QZ algorithm the matrix a sequence of equivalence transformations is used to reduce A to upper Hessenberg form and simultaneously B to upper triangular form. Note that this corresponds to a reduction of AB^{-1} to upper Hessenberg form. This can be performed using standard Householder transformations and Givens rotations as follows. This step begins by finding an orthogonal matrix Q such that $Q^T B$ is upper triangular. The same transformation is applied also to A . Next plane rotations are used to reduce $Q^T A$ to upper Hessenberg form, while preserving the upper triangular form of $Q^T B$. This step produces

$$Q^T(A, B)Z = (Q^T AZ, Q^T BZ) = (H_A, R_B).$$

The elements in $Q^T A$ are zeroed starting in the first column working from bottom up. This process is then repeated on the columns $2 : n$. Infinite eigenvalues, which correspond to zero diagonal elements of R_B can be eliminated at this step.

After the initial transformation the implicit shift QR algorithm is applied to $H_A R_B^{-1}$, but without forming this product explicitly. This is achieved by computing unitary matrices $\tilde{Q}$ and $\tilde{Z}$ such that $\tilde{Q} H_A \tilde{Z}$ is upper Hessenberg and $\tilde{Q} H_A \tilde{Z}$ upper triangular and choosing the first column of $\tilde{Q}$ proportional to the first column of $H_A R_B^{-1} - \sigma I$. We show below how the $(5, 1)$ element in A is eliminated by pre-multiplication by a plane rotation:

$$\begin{array}{c}\\ \\ \\ \rightarrow \\ \rightarrow\end{array}\begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix}, \quad \begin{array}{c}\\ \\ \\ \rightarrow \\ \rightarrow\end{array}\begin{pmatrix} b & b & b & b & b \\ & b & b & b & b \\ & & b & b & b \\ & & & b & b \\ & & & \hat{b} & b \end{pmatrix}.$$

This introduces a new nonzero element $\hat{b}$ in the $(5, 4)$ position in B, which is zeroed by a post-multiplication by a plane rotation

$$\begin{array}{c} \begin{array}{ccccc} & & & \downarrow & \downarrow \end{array} \\ \begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix} \end{array}, \quad \begin{array}{c} \begin{array}{ccccc} & & & \downarrow & \downarrow \end{array} \\ \begin{pmatrix} b & b & b & b & b \\ & b & b & b & b \\ & & b & b & b \\ & & & b & b \\ & & & \hat{b} & b \end{pmatrix} \end{array}.$$

All remaining steps in the reduction of A to upper Hessenberg form are similar. The complete reduction requires about $34n/3$ flops. If eigenvectors are to be computed, the product of the post-multiplications must be accumulated, which requires another $3n$ flops.

If A and B are real the Francis double shift technique can be used, where the shifts are chosen as the two eigenvalues of the trailing 2×2 pencil

$$\begin{pmatrix} a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{n,n} \end{pmatrix} - \lambda \begin{pmatrix} b_{n-1,n-1} & b_{n-1,n} \\ 0 & b_{n,n} \end{pmatrix}.$$

The details of the QZ step are similar to that in the implicit QR algorithm and will not be described here. The first Householder transformation is determined by the shift. When (H_A, R_B) is pre-multiplied by this new nonzero elements are introduced. This "bulge" is chased down the matrix by pre- and post-multiplication by Householder and plane rotations until the upper Hessenberg and triangular forms are restored.

The matrix H_A will converge to upper triangular form and the eigenvalues of $A - \lambda B$ will be obtained as ratios of diagonal elements of the transformed H_A and R_B. For a more detailed description of the algorithm see Stewart [340, Chapter 7.6].

The total work in the QZ algorithm is about $15n^3$ flops for eigenvalues alone, $8n^3$ more for Q and $10n^3$ for Z (assuming 2 QZ iterations per eigenvalue). It avoids the loss of accuracy related to explicitly inverting B. Although the algorithm is applicable to the case when A is symmetric and B positive definite, the transformations do not preserve symmetry and the method is just as expensive as for the general problem.

9.8.5 Generalized Singular Value Decomposition

We now introduce the **generalized singular value decomposition** (GSVD) of two matrices $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{p\times n}$ with the same number of columns. The GSVD has applications to, e.g., constrained least squares problems. The GSVD is related to the generalized eigenvalue problem $A^TAx = \lambda B^TBx$, but as in the case of the SVD the formation of A^TA and B^TB should be avoided. In the theorems below we assume for notational convenience that $m \geq n$. The GSVD was first proposed by Van Loan [371]. A form more suitable for numerical computation was suggested by Paige and Saunders [301]. The implementation used in LAPACK is described in [16] and [14].

Theorem 9.8.6. The Generalized Singular Value Decomposition (GSVD). *Let $A \in \mathbf{R}^{m\times n}$, $m \geq n$, and $B \in \mathbf{R}^{p\times n}$ be given matrices. Assume that*

$$\operatorname{rank}(M) = k \leq n, \qquad M = \begin{pmatrix} A \\ B \end{pmatrix}.$$

Then there exist orthogonal matrices $U_A \in \mathbf{R}^{m\times m}$ and $U_B \in \mathbf{R}^{p\times p}$ and a matrix $Z \in \mathbf{R}^{k\times n}$ of rank k such that

$$U_A^TA = \begin{pmatrix} D_A \\ 0 \end{pmatrix} Z, \qquad U_B^TB = \begin{pmatrix} D_B & 0 \\ 0 & 0 \end{pmatrix} Z, \tag{9.8.11}$$

where

$$D_A = \operatorname{diag}(\alpha_1, \ldots, \alpha_k), \qquad D_B = \operatorname{diag}(\beta_1, \ldots, \beta_q), \qquad q = \min(p, k).$$

Further, we have

$$\begin{gathered} 0 \leq \alpha_1 \leq \cdots \leq \alpha_k \leq 1, \qquad 1 \geq \beta_1 \geq \cdots \geq \beta_q \geq 0, \\ \alpha_i^2 + \beta_i^2 = 1, \quad i = 1, \ldots, q, \quad \alpha_i = 1, \quad i = q+1, \ldots, k, \end{gathered}$$

and the singular values of Z equal the nonzero singular values of M.

Proof. We now give a constructive proof of Theorem 9.8.6 using the CS decomposition, Let the SVD of M be

$$M = \begin{pmatrix} A \\ B \end{pmatrix} = Q \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} P^T,$$

where Q and P are orthogonal matrices of order $(m+p)$ and n, respectively, and

$$\Sigma_1 = \operatorname{diag}(\sigma_1, \dots, \sigma_k), \qquad \sigma_1 \geq \cdots \geq \sigma_k > 0.$$

Set $t = m + p - k$ and partition Q and P as follows:

$$Q = \begin{pmatrix} Q_{11} & Q_{12} \\ \underbrace{Q_{21}}_{k} & \underbrace{Q_{22}}_{t} \end{pmatrix} \begin{matrix} \}m \\ \}p \end{matrix}, \qquad P = (\underbrace{P_1}_{k}, \underbrace{P_2}_{n-k}).$$

Then the SVD of M can be written

$$\begin{pmatrix} A \\ B \end{pmatrix} P = \begin{pmatrix} AP_1 & 0 \\ BP_1 & 0 \end{pmatrix} = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} \begin{pmatrix} \Sigma_1 & 0 \end{pmatrix}. \tag{9.8.12}$$

Now let

$$Q_{11} = U_A \begin{pmatrix} C \\ 0 \end{pmatrix} V^T, \qquad Q_{21} = U_B \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} V^T$$

be the CS decomposition of Q_{11} and Q_{21}. Substituting this into (9.8.12) we obtain

$$AP = U_A \begin{pmatrix} C \\ 0 \end{pmatrix} V^T \begin{pmatrix} \Sigma_1 & 0 \end{pmatrix},$$
$$BP = U_B \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} V^T \begin{pmatrix} \Sigma_1 & 0 \end{pmatrix},$$

and (9.8.11) follows with

$$D_A = C, \quad D_B = S, \quad Z = V^T \begin{pmatrix} \Sigma_1 & 0 \end{pmatrix} P^T.$$

Here $\sigma_1 \geq \cdots \geq \sigma_k > 0$ are the singular values of Z. □

When $B \in \mathbf{R}^{n\times n}$ is square and nonsingular the GSVD of A and B corresponds to the SVD of AB^{-1}. However, when A or B is ill-conditioned, then computing AB^{-1} would usually lead to unnecessarily large errors, so this approach is to be avoided. It is important to note that when B is not square, or is singular, then the SVD of $AB^{\dagger}$ does not in general correspond to the GSVD.

9.8.6 Structured Eigenvalue Problems

Many eigenvalue problems in Applied Mathematics have some form of symmetry, which implies that its spectrum has certain properties. We have already seen that a Hermitian matrix has real eigenvalues. Unless such a structure is preserved by the numerical method used to solve the eigenproblem useless results may be produced.

In the paper [56] several important structured eigenvalue problems are discussed. Let $m = 2n$, and set

$$J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}. \tag{9.8.13}$$

A matrix is called J-symmetric if $(JA)^T = JA$, and J-Hermitian or **Hamiltonian** if $(JA)^H = JA$. Such eigenvalue problems occur, e.g.,when solving continuous time linear quadratic optimal control problems. A Hamiltonian matrix has pairs of eigenvalues $\lambda, -\bar{\lambda}$ and eigenvectors are related by

$$Ax = \lambda x \;\Rightarrow\; (Jx)^H A = -\bar{\lambda}(Jx)^H. \tag{9.8.14}$$

A matrix is called J-orthogonal or **symplectic** if $A^T JA = J$ and J-unitary if $A^H JA = J$. For a symplectic matrix the eigenvalues occur in pairs $\lambda, 1/\lambda$ and the eigenvectors are related by

$$Ax = \lambda x \;\Rightarrow\; (Jx)^T A = 1/\lambda(Jx)^T. \tag{9.8.15}$$

Review Questions

8.1 What is meant by a regular matrix pencil? Give examples of a singular pencil, and a regular pencil that has an infinite eigenvalue.

8.2 Formulate a generalized Schur Canonical Form. Show that the eigenvalues of the pencil are easily obtained from the canonical form.

8.3 Let A and B be real symmetric matrices, and B also positive definite. Show that there is a congruence transformation that diagonalizes the two matrices simultaneously. How is the Rayleigh Quotient iteration generalized to this type of eigenvalue problems, and what is its order of convergence?

Problems

8.1 Show that the matrix pencil $A - \lambda B$ where

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

has complex eigenvalues, even though A and B are both real and symmetric.

8.2 Let A and B be symmetric tridiagonal matrices. Assume that B is positive definite and let $B = LL^T$, where the Cholesky factor L is lower bidiagonal.

(a) Show that L can be factored as $L = L_1 L_2 \cdots L_n$, where L_k differs from the unit matrix only in the kth column.

(b) Consider the recursion

$$A_1 = A, \quad A_{k+1} = Q_k L_k^{-1} A_k L_k^{-T} Q_k^T, \quad k = 1 : n.$$

Show that if Q_k are orthogonal, then the eigenvalues of A_{n+1} are the same as those for the generalized eigenvalue problem $Ax = \lambda Bx$.

(c) Show how to construct Q_k as a sequence of Givens rotations so that the matrices A_k are all tridiagonal. (The general case, when A and B have symmetric bandwidth $m > 1$, can be treated by considering A and B as block-tridiagonal.)

Notes and Further References

A still unsurpassed text on computational methods for the eigenvalue problem is Wilkinson [387, 1965]. Also the Algol subroutines and discussions in Wilkinson and Reinsch [390, 1971] are very instructive. An excellent discussion of the symmetric eigenvalue problem is given in Parlett [305, 1980]. Methods for solving large scale eigenvalue problems are treated by van der Vorst [368, 2002].

Many important practical details on implementation of eigenvalue algorithms can be found in the documentation of the EISPACK and LAPACK software; see Smith et al. [336, 1976], B. S. Garbow et al. [149, 1977], and E. Anderson et al. [7, 1999].

Section 9.2

An excellent source of results on matrix perturbation is given by Stewart and Sun [348]. Perturbation theory of eigenvalues and eigenspaces, with an emphasize on Hermitian and normal matrices, is treated by Bhatia [32, 33].

Classical perturbation theory for the Hermitian eigenvalue and singular value problems bounds the absolute perturbations. These bounds may grossly overestimate the perturbations in eigenvalues and singular values of small magnitude. Ren-Cang Li [265, 266] studies relative perturbations in eigenvalues and singular values.

Section 9.3

An analysis and a survey of inverse iteration for a single eigenvector is given by Ipsen [224]. The relation between simultaneous iteration and the QR algorithm and is explained in Watkins [382].

Section 9.4

Braman, Byers and Mathias [48, 49] have developed a version of the QR algorithm that uses a large number of shifts in each QR step. For a matrix of order $n = 2000$ (say) they use of the order of $m = 100$ shifts computed from the current lower right-hand 100×100 principal submatrix. These eigenvalues are computed using a standard QR algorithm with $m = 2$. The 100 shifts are not applied all at once by chasing a large bulge. Instead they are used to perform 50 implicit QR iterations each of degree 2. These can be be applied in tight formation allowing level 3 performance.

A stable algorithm for computing the SVD based on an initial reduction to bidiagonal form was first sketched by Golub and Kahan in [173]. The adaption of the QR algorithm, using a simplified process due to Wilkinson, for computing the SVD of the bidiagonal matrix was described by Golub [170]. The "final" form of the QR algorithm for computing the SVD was given by Golub and Reinsch [181]. The GSVD was first studied by Van Loan [184, 1996]. Paige and Saunders [301, 1981] extended the GSVD to handle all possible cases, and gave a computationally more amenable form.

For a survey of cases when it is possible to compute singular values and singular vectors with high relative accuracy; see [92].

A survey of product eigenvalue problems is given by Watkins [381].

Section 9.6

For a more complete treatment of matrix functions see Chapter V in Gantmacher [147, 148, 1959] and Lancaster [249, 1985]. Stewart and Sun [348] is a lucid treatise of matrix perturbation theory, with many historical comments and a very useful bibliography. Methods for computing A^α, $\log(A)$ and related matrix functions by contour integrals are analyzed in [196].

Chapter 10

Iterative Methods for Linear Systems

While using iterative methods still requires know-how, skill, and insight, it can be said that enormous progress has been made for their integration in real-life applications.
—Yousef Saad and Henk A. van der Vorst, Iterative solution of linear systems in the 20th century, 2000.

10.1 Classical Iterative Methods

10.1.1 Introduction

The methods discussed so far in this book for solving systems of linear equations $Ax = b$ have been direct methods based on matrix factorization. Disregarding rounding errors, direct methods yield the exact solution in a fixed finite number of operations. Iterative methods, on the other hand, start from an initial approximation, which is successively improved until a sufficiently accurate solution is obtained. The idea of solving systems of linear equations by iterative methods dates at least back to Gauss (1823).

One important feature of basic iterative methods is that they work directly with the original matrix A and only need extra storage for a few vectors. Since the matrix A is involved only in terms of matrix-vector products and there is usually no need even to store A. Iterative methods are particularly useful for solving *sparse linear systems*, which typically arise in the solution of boundary value problems of partial differential equations by finite difference or finite element methods. The matrices involved can be huge, often involving several million unknowns. Since the LU factors of matrices in such applications would typically contain order of magnitudes more nonzero elements than A itself, direct methods may become far too costly (or even impossible) to use. This is true in particular for problems arising from three-dimensional simulations in e.g., reservoir modeling, mechanical engineering, electric circuit simulation. However, in some areas, e.g., structural engineering, which typically yield very ill-conditioned matrices, direct methods are still preferred.

Before the advent of computers iterative methods used were usually noncyclic relaxation methods guided at each step by the sizes of the residuals of the current solution. When in the 1950s computers replaced desk calculators an intense development of iterative methods started.

The distinction between direct and iterative methods is not sharp. Often an iterative method is applied to a, so called, preconditioned version of the system. This involves the solution of a simpler auxiliary systems by a direct method at each iteration.

10.1.2 A Model Problem

Let $Ax = b$ be a linear system of equations, where A is square and nonsingular. Let $x^{(0)}$ be a given initial approximation (e.g., $x^{(0)} = 0$). A sequence of approximations $x^{(1)}$, $x^{(2)}, \ldots$, is then computed, which converges to the solution. Iterative methods used before the age of high speed computers were usually rather unsophisticated **relaxation methods**. In **Richardson's method**,[46] the next approximation is computed from

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 0, 1, 2, \ldots, \tag{10.1.1}$$

where $\omega_k > 0$ are parameters to be chosen. It follows easily from (10.1.1) that the residual $r^{(k)} = b - Ax^{(k)}$ and error satisfy the recursions

$$r^{(k+1)} = (I - \omega_k A)r^{(k)}, \quad x^{(k+1)} - x = (I - \omega_k A)(x^{(k)} - x).$$

In the special case that $\omega = \omega_k$ for all k, Richardson's method is a **stationary** iterative method and

$$x^{(k)} - x = (I - \omega A)^k (x^{(0)} - x).$$

If A has a nonzero diagonal it can be scaled to have all diagonal elements equal to 1. Then Richardson's method with $\omega = 1$ is equivalent to **Jacobi's method**. The convergence of stationary iterative methods will be considered in Sec. 10.1.3.

Early iterative methods were predominantly applied for solving discretized elliptic self-adjoint partial differential equations. Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad (x, y) \in \Omega = (0, 1) \times (0, 1),$$

with $u(x, y)$ prescribed on the boundary Ω, is frequently used as model problem for these methods.

To approximate the solution we impose a uniform square mesh of size $h = 1/n$ on Ω. Let $u_{i,j}$ denote approximations to $u(x_i, y_j)$ at the mesh points $x_i = ih$,

[46]Lewis Fry Richardson (1881–1953) English mathematician, who was the first to use mathematical methods for weather prediction.

$y_j = jh$. Approximating the second derivatives by symmetric difference quotients at interior mesh points gives an $(n-1)^2$ equations

$$\frac{1}{h^2}\big(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}\big) = 0, \quad 0 < i, j < n,$$

for the unknown values $u_{i,j}$. If the mesh points are enumerated line by line (the so called "natural ordering") and a vector u is formed from the unknown function values, the difference equation can then be written in matrix form as

$$Au = b, \quad u = (u_1, u_2, \dots, u_N), \quad N = (n-1)^2,$$

where u_i is the vector of unknowns in the ith line and the right hand side is formed by the known values of $u(x, y)$ at the boundary. Note that the matrix A is symmetric by construction.

The linear system arising from Poisson's equation has several typical features common to other boundary value problems for second order linear partial differential equations. One of these is that there are at most 5 nonzero elements in each row of A. This means that *only a tiny fraction of the elements are nonzero.* Such matrices are called **sparse**. Therefore a matrix-vector multiplication Ax requires only about $5 \cdot N^2$ multiplications or equivalently five multiplications per unknown. Using iterative methods which take advantage of the sparsity and other features does allow the efficient solution of such systems. This becomes even more essential for three-dimensional problems!

It can be verified that the matrix A has the block-tridiagonal form

$$A = \operatorname{trid}(-1, 2, -1) = \begin{pmatrix} 2I + T & -I & & \\ -I & 2I + T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & 2I + T \end{pmatrix} \in \mathbf{R}^{N\times N}, \qquad (10.1.2)$$

where $N = (n-1)^2$ and T is symmetric tridiagonal,

$$T = \operatorname{trid}(-1, 2, -1) = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbf{R}^{(n-1)\times(n-1)}. \qquad (10.1.3)$$

In principle Gaussian elimination can be used to solve such systems. However, even taking symmetry and the banded structure into account this would require $\frac{1}{2} \cdot N^4$ multiplications, since in the LU factors the zero elements inside the outer diagonals will fill-in during the elimination. Hence L contains about n^3 nonzero elements compared to only about $5n^2$ in A as shown in Figure 10.1.1 (right). To compute the Cholesky factorization of a symmetric band matrix of order n and (half) bandwidth w requires approximately $\frac{1}{2}nw^2$ flops (see Algorithm 6.4.6). For the matrix A in (10.1.3) the dimension is n^2 and the bandwidth equals n. Hence

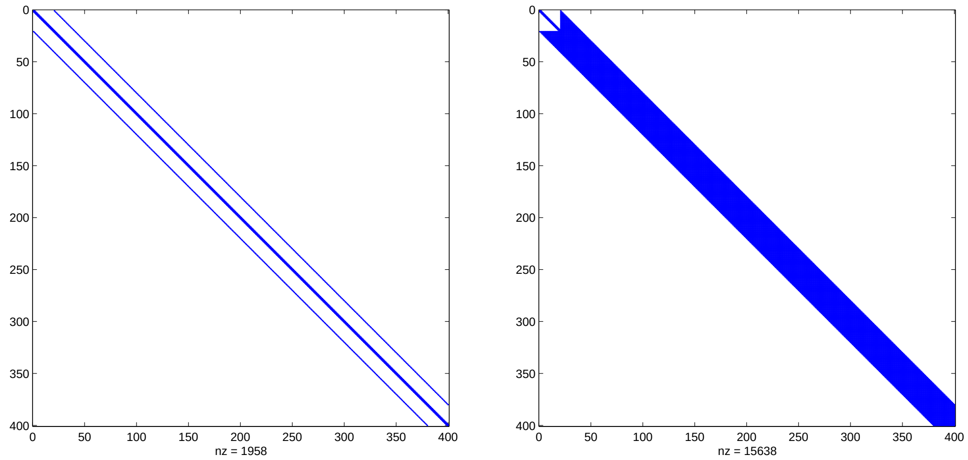

Figure 10.1.1. *Structure of A (left) and $L+U$ (right) for the Poisson problem, $n = 20$. Row-wise ordering of the unknowns)*

about $\frac{1}{2}n^4$ flops are needed for the factorization. This can be compared to the $5n^2$ flops needed per iteration, e.g., with Jacobi's method.

The above shows that for the model problem direct methods use $O(n^2)$ flops and about $O(n)$ storage per grid point. This disadvantage becomes even more accentuated if a three dimensional problem is considered. For Laplace equation in the unit cube a similar study shows that for solving n^3 unknown we need $\frac{1}{2}n^7$ flops and about n^5 storage. When n grows this quickly becomes infeasible. However, basic iterative methods still require only about $7n^3$ flops per iteration.

We still have not discussed the number of iterations needed to get acceptable accuracy. It turns out that this will depend on the condition number of the matrix. We now show that for the Laplace equation considered above this condition number will be about πh^{-2}, independent of the dimension of the problem.

Lemma 10.1.1.

Let $T = \operatorname{trid}(c, a, b) \in \mathbf{R}^{(n-1)\times(n-1)}$ be a tridiagonal matrix with constant diagonals, and assume that a, b, c are real and $bc > 0$. Then the n eigenvalues of T are given by

$$\lambda_i = a + 2\sqrt{bc}\cos\frac{i\pi}{n}, \quad i = 1 : n-1.$$

Further, the jth component of the eigenvector v_i corresponding to λ_i is

$$v_{ij} = \left(\frac{b}{c}\right)^{j/2} \sin\frac{ij\pi}{n}, \quad j = 1 : n-1.$$

From Lemma 10.1.1 it follows that the eigenvalues of $T = \operatorname{trid}(-1, 2, -1)$ are

$$\lambda_i = 2(1 + \cos(i\pi/n)), \quad i = 1 : n-1,$$

and in particular

$$\lambda_{\max} = 2(1+\cos(\pi/n)) \approx 4, \qquad \lambda_{\min} = 2(1-\cos(\pi/n)) \approx (\pi/n)^2.$$

We conclude that the spectral condition number of $T = \mathrm{trid}\,(-1, 2, -1)$ is approximately equal to $\kappa(T) = 4n^2/\pi^2$.

The matrix $A = 4(I - L - U)$ in (10.1.2) can be written in terms of the Kronecker product (see Sec. 7.5.5)

$$A = (I \otimes T) + (T \otimes I),$$

i.e., A is the Kronecker sum of T and T. It follows that the $(n-1)^2$ eigenvalues of A are $(\lambda_i + \lambda_j)$, $i, j = 1 : n-1$, and hence the condition number of A is the same as for T. The same conclusion can be shown to hold for a three dimensional problem.

The eigenvalues and eigenvectors of $C = A \otimes B$ can be expressed in terms of the eigenvalues and eigenvectors of A and B. Assume that $Ax_i = \lambda_i x_i$, $i = 1, \dots, n$, and $By_j = \mu_j y_j$, $j = 1, \dots, m$. Then, using equation (7.5.26), we obtain

$$(A \otimes B)(x_i \otimes y_j) = (Ax_i) \otimes (By_j) = \lambda_i \mu_j (x_i \otimes y_j). \tag{10.1.4}$$

This shows that the nm eigenvalues of $A \otimes B$ are $\lambda_i \mu_j$, $i = 1, \dots, n$, $j = 1, \dots, m$, and $x_i \otimes y_j$ are the corresponding eigenvectors. If A and B are diagonalizable, $A = X^{-1}\Lambda_1 X$, $B = Y^{-1}\Lambda_2 Y$, then

$$(A \otimes B) = (X^{-1} \otimes Y^{-1})(\Lambda_1 \otimes \Lambda_2)(X \otimes Y),$$

and thus $A \otimes B$ is also diagonalizable.

The matrix

$$(I_m \otimes A) + (B \otimes I_n) \in \mathbf{R}^{nm \times nm} \tag{10.1.5}$$

is the **Kronecker sum** of A and B. Since

$$\begin{aligned}\big[(I_m \otimes A) + (B \otimes I_n)\big](y_j \otimes x_i) &= y_j \otimes (Ax_i) + (By_j) \otimes x_i \\ &= (\lambda_i + \mu_j)(y_j \otimes x_i).\end{aligned} \tag{10.1.6}$$

the nm eigenvalues of the Kronecker sum equal the sum of all pairs of eigenvalues of A and B

10.1.3 Stationary Iterative Methods

We start by describing two classical iterative methods. Assume that A has nonzero diagonal entries, i.e, $a_{ii} \neq 0$, $i = 1 : n$. If A is symmetric, positive definite this is necessarily the case. Otherwise, since A is nonsingular, the equations can always be reordered so that this is true. In component form the system can then be written

$$x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j\Big), \qquad i = 1 : n. \tag{10.1.7}$$

In a (minor) step of **Jacobi's method** we pick one equation, say the ith, and adjust the ith component of $x^{(k)}$ so that this equation becomes exactly satisfied. Hence, given $x^{(k)}$ we compute

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)}. \tag{10.1.8}$$

In the days of "hand" computation one picked an equation with a large residual $|r_i$ and went through the equations in an irregular manner. This is less efficient when using a computer, and here one usually perform these adjustments for $i = 1 : n$, in a cyclic fashion. Jacobi's method is therefore also called the method of *simultaneous displacements*. Note that all components of x can be updated *simultaneously* and the result does not depend on the sequencing of the equations.

The method of successive displacements or **Gauss–Seidel's method**[47] differs from the Jacobi method only by using new values $x_j^{(k+1)}$ as soon as they are available

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)}, \quad i = 1, 2 \dots, n. \tag{10.1.9}$$

Here the components are *successively* updated and the sequencing of the equations will influence the result.

Since each new value $x_i^{(k+1)}$ can immediately replace $x_i^{(k)}$ in storage the Gauss–Seidel method storage for unknowns is halved compared to Jacobi's method. For both methods the amount of work required in each iteration step is comparable in complexity to the multiplication of A with a vector, i.e., proportional to the number of nonzero elements in A. By construction it follows that if $\lim_{k\to\infty} x^{(k)} = x$, then x satisfies (10.1.7) and therefore the system $Ax = b$.

The Jacobi, Gauss–Seidel, and Richardson methods are all special cases of a class of iterative methods, the general form of which is

$$Mx^{(k+1)} = Nx^{(k)} + b, \qquad k = 0, 1, \dots. \tag{10.1.10}$$

Here

$$A = M - N \tag{10.1.11}$$

is a **splitting** of the matrix coefficient matrix A with M nonsingular.

If the iteration (10.1.10) converges, i.e., $\lim_{k\to\infty} x^{(k)} = x$, then $Mx = Nx + b$ and it follows from (10.1.11) that the limit vector x solves the linear system $Ax = b$. For the iteration to be practical, it must be easy to solve linear systems with matrix M. This is the case, for example, if M is chosen to be triangular.

Definition 10.1.2.

An iterative method of the form (10.1.10)*, or equivalently*

$$x^{(k+1)} = Bx^{(k)} + c, \quad k = 0, 1, \dots, \tag{10.1.12}$$

where

[47] It was noted by G. Forsythe that Gauss nowhere mentioned this method and Seidel never advocated using it!

$$B = M^{-1}N = I - M^{-1}A, \qquad c = M^{-1}b. \tag{10.1.13}$$

is called a (one-step) **stationary iterative method***, and the matrix B In (10.1.13) is called the* **iteration matrix**.

Subtracting the equation $x = Bx + c$ from (10.1.10), we obtain the recurrence formula

$$x^{(k+1)} - x = B(x^{(k)} - x) = \cdots = B^{k+1}(x^{(0)} - x), \tag{10.1.14}$$

for the errors in successive approximations.

Richardson's method (10.1.1) can, for fixed $\omega_k = \omega$, be written in the form (10.1.10) with the splitting $A = M - N$, with

$$M = (1/\omega)I, \qquad N = (1/\omega)I - A.$$

To write the Jacobi and Gauss–Seidel methods in the form of one-step stationary iterative methods we introduce the **standard splitting**

$$A = D_A - L_A - U_A, \tag{10.1.15}$$

where $D_A = \operatorname{diag}(a_{11}, \ldots, a_{nn})$,

$$L_A = -\begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \quad U_A = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & 0 & a_{n-1,n} \\ & & & 0 \end{pmatrix}, \tag{10.1.16}$$

and L_A and U_A are strictly lower and upper triangular, respectively. Assuming that $D_A > 0$, we can also write

$$D_A^{-1}A = I - L - U, \qquad L = D_A^{-1}L_A, \qquad U = D_A^{-1}U_A. \tag{10.1.17}$$

With these notations the Jacobi method, (10.1.8), can be written $D_A x^{(k+1)} = (L_A + U_A)x^{(k)} + b$ or

$$x^{(k+1)} = (L + U)x^{(k)} + c, \qquad c = D_A^{-1}b. \tag{10.1.18}$$

The Gauss–Seidel method, (10.1.9), becomes $(D_A - L_A)x^{(k+1)} = U_A x^{(k)} + b$, or equivalently

$$x^{(k+1)} = (I - L)^{-1}Ux^{(k)} + c, \quad c = (I - L)^{-1}D_A^{-1}b$$

Hence these methods are special cases of one-step stationary iterative methods, and correspond to the matrix splittings

$$\begin{array}{lll} \textit{Jacobi:} & M = D_A, & N = L_A + U_A, \\ \textit{Gauss–Seidel:} & M = D_A - L_A, & N = U_A, \end{array}$$

The iteration matrices for the Jacobi and Gauss–Seidel methods are

$$\begin{aligned} B_J &= D_A^{-1}(L_A + U_A) = L + U, \\ B_{GS} &= (D_A - L_A)^{-1}U_A = (I - L)^{-1}U. \end{aligned}$$

Many matrices arising from the discretization of partial differential equations have the following property:

Definition 10.1.3.

A matrix $A = (a_{ij})$ is an M-matrix if $a_{ij} \le 0$ for $i \ne j$, A is nonsingular and $A^{-1} \ge 0$.

In particular the matrix arising from the model problem is a symmetric M-matrix. Such a matrix is also called a **Stieltjes** matrix.

Often the matrices M and N in the splitting (10.1.11) of the matrix A has special properties that can be used in the analysis. Of particular interest is the following property.

Definition 10.1.4.

For a matrix $A \in \mathbf{R}^{n\times n}$, the splitting $A = M - N$ is a **regular splitting** *if M is nonsingular and $M^{-1} \ge 0$ and $N \ge 0$.*

It can be shown that if A is an M-matrix, then any splitting where M is obtained by setting certain off-diagonal elements of A to zero, gives a regular splitting and $\rho(M^{-1}N < 1$.

For the model problem the matrix A has a positive diagonal and the off diagonal elements were non-negative. Clearly this ensures that the Jacobi and Gauss–Seidel methods both correspond to a regular splitting.

For regular splittings several results comparing asymptotic rates of convergence can be obtained.

Theorem 10.1.5.

If $A = M - N$ is a regular splitting of the matrix A and $A^{-1} \ge 0$, then

$$\rho(M^{-1}N) = \frac{\rho(A^{-1}N)}{1 + \rho(A^{-1}N}) < 1. \tag{10.1.19}$$

Thus the iterative method (10.1.12) *converges for any initial vector $x^{(0)}$.*

Proof. See Varga [375, Theorem 3.13]. □

10.1.4 Convergence Analysis

The stationary iterative method (10.1.12) is called **convergent** if the sequence $\{x^{(k)}\}_{k=1,2,\ldots}$ converges for *all initial vectors* $x^{(0)}$. It can be seen from (10.1.14) that of fundamental importance in the study of convergence of stationary iterative methods is conditions for a sequence of powers of a matrix B to converge to the null matrix. For this we need some results from the theory of eigenvalues of matrices.

In Sec. 9.1.3 we introduced the spectral radius of a matrix A as the nonnegative number

$$\rho(A) = \max_{1\le i\le n} |\lambda_i(A)|.$$

The following important results holds:

Theorem 10.1.6. *A matrix $B \in \mathbf{R}^{n\times n}$ is said to be* **convergent** *if $\rho(B) < 1$. It holds that*

$$\lim_{k\to\infty} B^k = 0 \quad \Longleftrightarrow \quad \rho(B) < 1. \tag{10.1.20}$$

Proof. We will show that the following four conditions are equivalent:

(i) $\displaystyle\lim_{k\to\infty} B^k = 0,$

(ii) $\displaystyle\lim_{k\to\infty} B^k x = 0, \quad \forall x \in \mathbf{C}^n,$

(iii) $\rho(B) < 1,$

(iv) $\|B\| < 1$ for at least one matrix norm.

For any vector x we have the inequality $\|B^k x\| \le \|B^k\|\,\|x\|$, which shows that (i) implies (ii).
If $\rho(B) \ge 1$, then there is an eigenvector $x \in \mathbf{C}^n$ such that $Bx = \lambda x$, with $|\lambda| \ge 1$. Then the sequence $B^k x = \lambda^k x$, $k = 1, 2, \ldots$, is not convergent when $k \to \infty$ and hence (ii) implies (iii).
By Theorem 10.2.9, (see Section 10.2.4) given a number $\epsilon > 0$ there exists a consistent matrix norm $\|\cdot\|$, depending on B and ϵ, such that

$$\|B\| < \rho(B) + \epsilon.$$

Therefore (iv) follows from (iii).
Finally, by applying the inequality $\|B^k\| \le \|B\|^k$, we see that (iv) implies (i). □

From this theorem we deduce the following necessary and sufficient criterion for the convergence of a stationary iterative method.

Theorem 10.1.7. *The stationary iterative method $x^{(k+1)} = Bx^{(k)} + c$ is convergent for all initial vectors $x^{(0)}$ if and only if $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of B.*

Proof. From the recurrence (10.1.14) it follows that

$$x^{(k)} - x = B^k (x^{(0)} - x). \tag{10.1.21}$$

Hence $x^{(k)}$ converges for *all* initial vectors $x^{(0)}$ if and only if $\lim_{k\to\infty} B^k = 0$. The theorem now follows from Theorem 10.1.6. □

Obtaining the spectral radius of B is usually no less difficult than solving the linear system. Hence the following upper bound is useful.

Lemma 10.1.8.
For any matrix $A \in \mathbf{R}^{n\times n}$ and for any consistent matrix norm we have

$$\rho(A) \le \|A\|. \tag{10.1.22}$$

Proof. Let λ be an eigenvalue of A such that $|\lambda| = \rho(A)$. Then $Ax = \lambda x$, $x \neq 0$, and taking norms

$$\|\lambda x\| = \rho(A)\|x\| = \|Ax\| \le \|A\|\,\|x\|.$$

Since $\|x\| > 0$, we can divide the inequality by $\|x\|$ and the theorem follows. □

From Lemma 10.1.8 it follows that a *sufficient condition* for convergence of the iterative method is that $\|B\| < 1$, for *some* matrix norm.

Usually, we are not only interested in convergence, but also in the **rate of convergence**. By (10.1.21) the error at step k, $e^{(k)} = x^{(k)} - x$, satisfies $e^{(k)} = B^k e^{(0)}$. Thus for any consistent pair of norms it holds that $\|e^{(k)}\| \le \|B^k\|\,\|e^{(0)}\|$. On the average, we gain at least a factor $(\|B^k\|)^{1/k}$ per iteration. To reduce the norm of the error by at least a factor $\delta < 1$, it suffices to perform k iterations, where k is the smallest integer that satisfies $\|B^k\| \le \delta$. Taking logarithms and multiplying with $-1/k$ we obtain the equivalent condition

$$-\frac{1}{k}\log(\|B^k\|) \ge -\frac{1}{k}\log\delta.$$

Thus it suffices to perform k iterations, where

$$k \ge -\log\delta/R_k(B), \qquad R_k(B) = -\frac{1}{k}\log\|B^k\|.$$

This motivates the following definition:

Definition 10.1.9. *Assume that the iterative method $x^{(k+1)} = Bx^{(k)} + c$ is convergent is convergent. For any consistent matrix norm $\|\cdot\|$ we define the* **average rate** *of convergence by*

$$R_k(B) = -\frac{1}{k}\log\|B^k\|, \quad (\|B^k\| < 1). \tag{10.1.23}$$

The expression

$$R_\infty(B) = \lim_{k\to\infty} R_k(B) = -\log\rho(B).$$

for the **asymptotic rate** of convergence follows from the (non-trivial) result

$$\rho(B) = \lim_{k\to\infty}(\|B^k\|)^{1/k},$$

which holds for any consistent matrix norm. This can be proved by using the Jordan normal form; see Problem 10.2.4.

We now consider the convergence of some classical methods.

Theorem 10.1.10.
Assume that all the eigenvalues λ_i of A are real and satisfy

$$0 < a \le \lambda_i \le b, \quad i = 1 : n.$$

Then the stationary Richardson's method is convergent for $0 < \omega < 2/b$.

Proof. The iteration matrix of the stationary Richardson's method is $B = I - \omega A \in \mathbf{R}^{n\times n}$, with eigenvalues $\mu_i = 1 - \omega\lambda_i$. From the assumption $1 - \omega b \le \mu_i \le 1 - \omega a$, for all i. It follows that if $1 - \omega a < 1$ and $1 - \omega b > -1$, then $|\mu_i| < 1$ for all i and the method is convergent. Since $a > 0$ the first condition is satisfied for all $\omega > 0$, while the second is true if $\omega < 2/b$. $\square$

Assuming that $a = \lambda_{min}$ and $b = \lambda_{max}$. What value of ω will minimize the spectral radius

$$\rho(B) = \max\{|1 - \omega a|, |1 - \omega b|\}$$

and thus maximize the asymptotic rate of convergence? It is left as an exercise to show that this optimal ω is that which satisfies $1 - \omega a = \omega b - 1$, i.e $\omega_{\text{opt}} = 2/(b+a)$. (*Hint*: Plot the graphs of $|1 - \omega a|$ and $|1 - \omega b|$ for $\omega \in (0, 2/b)$.) It follows that

$$\rho(B) = \frac{b-a}{b+a} = \frac{\kappa - 1}{\kappa + 1} = 1 - \frac{2}{\kappa + 1},$$

where $\kappa = b/a$ is the condition number of A. Hence the optimal asymptotic rate of convergence is

$$R_\infty(B) = -\log\left(1 - \frac{2}{\kappa + 1}\right) \approx 2/\kappa, \quad \kappa \gg 1. \tag{10.1.24}$$

is inversely proportional to κ. This illustrates a typical fact for iterative methods: *in general ill-conditioned systems require more work to achieve a certain accuracy!*

Theorem 10.1.11. *The Jacobi method is convergent if A is strictly row-wise diagonally dominant, i.e.,*

$$|a_{ii}| > \sum_{\substack{j=1\\ j\ne i}}^{n} |a_{ij}|, \quad i = 1 : n.$$

Proof. For the Jacobi method the iteration matrix $B_J = L + U$ has elements $b_{ij} = -a_{ij}/a_{ii}$, $i \ne j$, $b_{ij} = 0$, $i = j$. From the assumption it then follows that

$$\|B_J\|_\infty = \max_{1\le i\le n} \sum_{\substack{j=1\\ j\ne i}}^{n} |a_{ij}|/|a_{ii}| < 1,$$

which proves the theorem. $\square$

A similar result for strictly column-wise diagonally dominant matrices can be proved using $\|B_J\|_1$. A slightly stronger convergence result than in Theorem 10.1.11

is of importance in applications. (Note that, e.g., the matrix A in (10.1.7) is not strictly diagonal dominant!) For irreducible matrices (see Def. 9.1.5) the row sum criterion in Theorem 10.1.11 can be sharpened.

Theorem 10.1.12. *The Jacobi method is convergent if A is irreducible, and in*

$$|a_{ii}| \geq \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|, \quad i = 1 : n,$$

inequality holds for at least one row.

The column sum criterion can be similarly improved. The conditions in Theorem 10.1.11–10.1.12 are also sufficient for convergence of the Gauss–Seidel method for which $(I-L)B_{GS} = U$. Consider the strictly row-wise diagonally dominant and choose k so that

$$\|B_{GS}\|_\infty = \|B_{GS}^T\|_1 = \|B_{GS}^T e_k\|_1.$$

Then from $B_{GS}^T e_k = B_{GS}^T L^T e_k + U^T e_k$, we get

$$\|B_{GS}\|_\infty \leq \|B_{GS}\|_\infty \|L^T e_k\|_1 + \|U^T e_k\|_1.$$

Since A is strictly row-wise diagonally dominant we have $\|L^T e_k\|_1 + \|U^T e_k\|_1 \leq \|B_J\|_\infty < 1$, and it follows that

$$\|B_{GS}\|_\infty \leq \|U^T e_k\|_1 / \big(1 - \|L^T e_k\|_1\big) < 1.$$

Hence the Gauss–Seidel method is convergent. The proof for the strictly column-wise diagonally dominant case is similar but estimates $\|B_{GS}\|_1$.

Example 10.1.1.

In Section 10.1.3 it was shown that the $(n-1)^2$ eigenvalues of the matrix

$$A = (I \otimes T) + (T \otimes I)$$

arising from the model problem are equal to

$$(\lambda_i + \lambda_j), \quad i, j = 1 : n-1, \quad \lambda_i = 2(1 + \cos(i\pi/n)).$$

It follows that the eigenvalues of the corresponding Jacobi iteration matrix $B_J = L + U = (1/4)(A - 4I)$ are

$$\mu_{ij} = \tfrac{1}{2}(\cos i\pi h + \cos j\pi h), \quad i, j = 1 : n-1,$$

where $h = 1/n$ is the grid size. The spectral radius is obtained for $i = j = 1$,

$$\rho(B_J) = \cos(\pi h) \approx 1 - \tfrac{1}{2}(\pi h)^2.$$

This means that the low frequency modes of the error are damped most slowly, whereas the high frequency modes are damped much more quickly.[48] The same is true for the Gauss–Seidel method, for which

$$\rho(B_{GS}) = \cos^2(\pi h) \approx 1 - (\pi h)^2,$$

The corresponding asymptotic rates of convergence are $R_\infty(B_J) \approx \pi^2 h^2/2$, and $R_\infty(B_{GS}) \approx \pi^2 h^2$. This shows that for the model problem Gauss–Seidel's method will converge asymptotically twice as fast as Jacobi's method. However, for both methods the number of iterations required is proportional to $\kappa(A)$ for the model problem.

The rate of convergence of the basic Jacobi and Gauss–Seidel methods, as exhibited in the above example, is in general much too slow to make them of any practical use. In Section 10.2.1 we show how, with a simple modification, the the rate of convergence of the Gauss–Seidel method for the model problem can be improved by a factor of n.

10.1.5 Effects of Nonnormality and Finite Precision

While the spectral radius determines the *asymptotic* rate of growth of matrix powers, the norm will influence the *initial* behavior of the powers B^k. However, the norm of a convergent matrix can for a nonnormal matrix be arbitrarily large. By the Schur normal form any matrix A is unitarily equivalent to an upper triangular matrix. Therefore, in exact arithmetic, it suffices to consider the case of an upper triangular matrix.

Consider the 2×2 convergent matrix

$$B = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}, \quad 0 < \mu \le \lambda < 1, \quad \alpha \gg 1, \tag{10.1.25}$$

for which we have $\|B\|_2 \gg \rho(B)$. Therefore, even though $\|B^k\| \to 0$ as $k \to \infty$, the spectral norms $\|B^k\|_2$ will initially sharply increase! It is easily verified that

$$B^k = \begin{pmatrix} \lambda^k & \beta_k \\ 0 & \mu^k \end{pmatrix}, \quad \beta_k = \begin{cases} \alpha \dfrac{\lambda^k - \mu^k}{\lambda - \mu} & \text{if } \mu \ne \lambda; \\ \alpha k \lambda^{k-1} & \text{if } \mu = \lambda. \end{cases} \tag{10.1.26}$$

Clearly the element β_k will grow initially. In the case that $\lambda = \mu$ the maximum of $|\beta_k|$ will occur when $k \approx \lambda/(1-\lambda)$. (See also Computer Exercise 1.)

For matrices of larger dimension the initial increase of $\|B^k\|$ can be huge as shown by the following example:

[48] This is one of the basic observations used in the multigrid method, which uses a sequence of different meshes to efficiently damp all frequencies.

Example 10.1.2.

Consider the iteration $x^{(k+1)} = Bx^{(k)}$, where $B \in \mathbf{R}^{20\times 20}$ is the bidiagonal matrix

$$B = \begin{pmatrix} 0.5 & 1 & & & \\ & 0.5 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0.5 & 1 \\ & & & & 0.5 \end{pmatrix}, \qquad x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Here $\rho(B) = 0.5$, and hence the iteration should converge to the exact solution of the equation $(I-B)x = 0$, which is $x = 0$. From Figure 10.1.2 it is seen that $\|x^{(n)}\|_2$ increases by more than a factor 10^5 until it starts to decrease after 35 iterations! Although in the long run the norm is reduced by about a factor of 0.5 at each iteration, large intermediate values of $x^{(n)}$ give rise to persistent rounding errors.

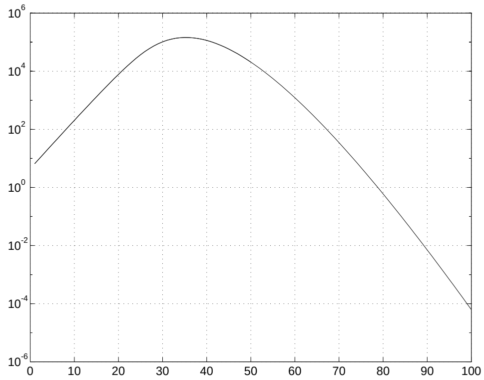

Figure 10.1.2. $\|x^{(n)}\|_2$, *where* $x^{(k+1)} = Bx^{(k)}$, *and* $x^{(0)} = (1, 1, \dots, 1)^T$.

The curve in Figure 10.1.2 shows a large hump. This is a typical phenomenon in several other matrix problems and occurs also, e.g., when computing the matrix exponential e^{Bt}, when $t \to \infty$.

For the case when the iteration process is carried out in exact arithmetic we found a complete and simple mathematical theory of convergence for iterates $x^{(k)}$ of stationary iterative methods. According to Theorem 10.1.6 there is convergence for any $x^{(0)}$ if and only if $\rho(B) < 1$, where $\rho(B)$ is the spectral radius of B. The same condition is necessary and sufficient for $\lim_{k\to\infty} B^k = 0$ to hold. In finite precision arithmetic the convergence behavior turns out to be more complex and less easy to analyze.

It may be thought that iterative methods are less affected by rounding errors than direct solution methods, because in iterative methods one continues to work with the original matrix instead of modifying it. In Section 6.6.6 we showed that the total effect of rounding errors in Gaussian elimination with partial pivoting usually is equivalent to a perturbations in the elements of the original matrix of the order of machine roundoff. It is easy to verify that, in general, iterative methods cannot be expected to do much better than that!

Consider an iteration step with the Gauss–Seidel method performed in floating point arithmetic. Typically, in the first step an improved x_1 will be computed from previous $x_2, \dots, x_n$ by

$$x_1 = fl\Big(\Big(b_1 - \sum_{j=1}^{n} a_{1j}x_j\Big)/a_{11}\Big) = \Big(b_1(1+\delta_1) - \sum_{j=1}^{n} a_{1j}x_j(1+\delta_j)\Big)/a_{11},$$

with the usual bounds for δ_i, cf. Section 2.4.1. This can be interpreted that we have performed an exact Gauss–Seidel step for a *perturbed problem* with elements $b_1(1+\delta_1)$ and $a_{1i}(1+\delta_i)$, $i = 2 : n$. The bounds for these perturbations are of the same order of magnitude that for the perturbations in Gaussian elimination. *The idea that we have worked with the original matrix is not correct*! Indeed, a round-off error analysis of iterative methods is in many ways *more difficult* to perform than for direct methods.

Example 10.1.3.

(J. H. Wilkinson) Consider the (ill-conditioned) system $Ax = b$, where

$$A = \begin{pmatrix} 0.96326 & 0.81321 \\ 0.81321 & 0.68654 \end{pmatrix}, \qquad b = \begin{pmatrix} 0.88824 \\ 0.74988 \end{pmatrix}.$$

The smallest singular value of A is $0.36 \cdot 10^{-5}$. This system is symmetric, positive definite and therefore the Gauss–Seidel method should converge, though slowly. Starting with $x_1 = 0.33116$, $x_2 = 0.70000$, the next approximation for x_1 is computed from the relation

$$x_1 = fl\big((0.88824 - 0.81321 \cdot 0.7)/0.96326\big) = 0.33116,$$

(working with five decimals). This would be an exact result if the element a_{11} was perturbed to be $0.963259\dots$, but no progress is made towards the true solution $x_1 = 0.39473\dots$, $x_2 = 0.62470\dots$. The ill-conditioning has affected the computations adversely. Convergence is so slow that the modifications to be made in each step are less than $0.5 \cdot 10^{-5}$.

Iterative methods can be badly affected by rounding errors for nonnormal matrices. In Example 10.1.2 we saw that the "hump" phenomenon can cause $\|x^{(k)}\|_2$ to increase substantially, even when the iteration eventually converges. In such a case cancellation will occur in the computation of the final solution, and a rounding error of size $u \max_k \|x^{(k)}\|_2$ remains, where u is the machine unit.

Moreover, for a nonnormal matrix B, asymptotic convergence in finite precision arithmetic is no longer guaranteed even when $\rho(B) < 1$ holds in exact arithmetic. This phenomenon is related to the fact that for a matrix of a high degree of nonnormality the spectrum can be extremely sensitive to perturbations. As shown above the computed iterate $\bar{x}^{(k)}$ will at best be the exact iterate corresponding to a perturbed matrix $B+\Delta B$. Hence even though $\rho(B) < 1$ it may be that $\rho(B+\Delta B)$ is larger than one. To have convergence in finite precision arithmetic we need a stronger condition to hold, e.g.,

$$\max \rho(B+E) < 1, \quad \|E\|_2 < u\|B\|_2,$$

where u is the machine precision. (Compare the discussion of pseudospectra in Section 9.3.3.) The following rule of thumb has been suggested:

> *The iterative method with iteration matrix B can be expected to converge in finite precision arithmetic if the spectral radius computed via a backward stable eigensolver is less than 1.*

This is an instance when an inexact result is more useful than the exact result!

10.1.6 Termination Criteria

An iterative method solving a linear system $Ax = b$ is not completely specified unless clearly defined criteria are given for when to stop the iterations. Ideally such criteria should identify when the error $x - x^{(k)}$ is small enough and also detect if the error is no longer decreasing or decreasing too slowly.

Normally a user would like to specify an absolute (or a relative) tolerance ϵ for the error, and stop as soon as

$$\|x - x^{(k)}\| \leq \epsilon \tag{10.1.27}$$

is satisfied for some suitable vector norm $\|\cdot\|$. However, such a criterion cannot in general be implemented since x is unknown. Moreover, if the system to be solved is illconditioned, then because of roundoff the criterion (10.1.27) may never be satisfied.

Instead of (10.1.27) one can use a test on the residual vector $r^{(k)} = b - Ax^{(k)}$, which is computable, and stop when

$$\|r^{(k)}\| \leq \epsilon(\|A\|\,\|x^{(k)}\| + \|b\|).$$

This is often replaced by the stricter criterion

$$\|r^{(k)}\| \leq \epsilon\|b\|, \tag{10.1.28}$$

but this may be difficult to satisfy in case $\|b\| \ll \|A\|\|x\|$. Although such residual based criteria are frequently used, it should be remembered that if A is ill-conditioned a small residual does not guarantee a small relative error in the approximate solution. Since $x - x^{(k)} = A^{-1}r^{(k)}$, (10.1.28) only guarantees that $\|x - x^{(k)}\| \leq \epsilon\|A^{-1}\|\,\|b\|$, and this bound is attainable.

Another possibility is to base the stopping criterion on the Oettli–Prager backward error, see Theorem. 6.6.4. The idea is then to compute the quantity

$$\omega = \max_i \frac{|r_i^{(k)}|}{(E|x^{(k)}| + f)_i}, \tag{10.1.29}$$

where $E > 0$ and $f > 0$, and stop when $\omega \le \epsilon$. It then follows from Theorem 6.6.4 that $x^{(k)}$ is the exact solution to a perturbed linear system

$$(A + \delta A)x = b + \delta b, \quad |\delta A| \le \omega E, \quad |\delta b| \le \omega f.$$

We could in (10.1.29) take $E = |A|$ and $f = |b|$, which corresponds to componentwise backward errors. However, it can be argued that for iterative methods a more suitable choice is to use a normwise backward error by setting

$$E = \|A\|_\infty ee^T, \quad f = \|b\|_\infty e, \quad e = (1, 1, \ldots, 1)^T.$$

This choice gives

$$\omega = \frac{\|r^{(k)}\|_\infty}{\|A\|_\infty \|x^{(k)}\|_1 + \|b\|_\infty}.$$

Review Questions

1. The standard discretization of Laplace equation on a square with Dirichlet boundary conditions leads to a certain matrix A. Give this matrix in its block triangular form.
2. What iterative method can be derived from the splitting $A = M - N$? How is a symmetrizable splitting defined?
3. Define the average and asymptotic rate of convergence for an iterative method $x^{(k+1)} = Bx^{(k)} + c$. Does the condition $\rho(B) < 1$ imply that the error norm $\|x - x^{(k)}\|_2$ is monotonically decreasing? If not, give a counterexample.
4. Give at least two different criteria which are suitable for terminating an iterative method.

Problems and Computer Exercises

1. Let $A \in \mathbf{R}^{n\times n}$ be a given nonsingular matrix, and $X^{(0)} \in \mathbf{R}^{n\times n}$ an arbitrary matrix. Define a sequence of matrices by

$$X^{(k+1)} = X^{(k)} + X^{(k)}(I - AX^{(k)}), \quad k = 0, 1, 2, \ldots.$$

(a) Prove that $\lim_{k\to\infty} X^{(k)} = A^{-1}$ if and only if $\rho(I - AX^{(0)}) < 1$.
Hint: First show that $I - AX^{(k+1)} = (I - AX^{(k)})^2$.
(b) Use the iterations to compute the inverse A^{-1}, where

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \qquad X^{(0)} = \begin{pmatrix} 1.9 & -0.9 \\ -0.9 & 0.9 \end{pmatrix}.$$

Verify that the rate of convergence is quadratic!

2. Let $A \in \mathbf{R}^{m\times n}$ be a given nonsingular matrix, Consider the stationary iterative method
$$x^{(k+1)} = x^{(k)} + \omega A^T(b - Ax^{(k)}),$$
where $A \in R^{m\times n}$ is a possibly rank deficient matrix.

(a) Show that if $\operatorname{rank}(A) = n$ and $0 < \omega < 2/\sigma^2_{\max}(A)$ then then the iteration converges to the unique solution to the normal equations $A^T Ax = A^T b$.

(b) If $\operatorname{rank}(A) < n$, then split the vector $x^{(k)}$ into orthogonal components,
$$x^{(k)} = x_1^{(k)} + x_2^{(k)}, \quad x_1^{(k)} \in \mathcal{R}(A^T), \quad x_2^{(k)} \in \mathcal{N}(A).$$
Show that the orthogonal projection of $x^{(k)} - x^{(0)}$ onto $\mathcal{N}(A)$ is zero. Conclude that in this case the iteration converges to the unique solution of the normal equations which minimizes $\|x - x^{(0)}\|_2$.

3. Show that if for a stationary iterative method $x^{(k+1)} = Bx^{(k)} + c$ it holds that $\|B\| \le \beta < 1$, and
$$\|x^{(k)} - x^{(k-1)}\| \le \epsilon(1-\beta)/\beta,$$
then the error estimate $\|x - x^{(k)}\| \le \epsilon$ holds.

4. Let B be the 2×2 matrix in (10.1.26), and take $\lambda = \mu = 0.99$, $\alpha = 4$. Verify that $\|B^k\|_2 \ge 1$ for all $k < 805$!

5. Let $B \in \mathbf{R}^{20\times 20}$ be an upper bidiagonal matrix with diagonal elements equal to $0.025, 0.05, 0.075, \dots, 0.5$ and elements in the superdiagonal all equal to 5.

(a) Compute and plot $\eta_k = \|x^{(k)}\|_2/\|x^{(0)}\|_2$, $k = 0 : 100$, where
$$x^{(k+1)} = Bx^{(k)}, \quad x^{(0)}0 = (1, 1, \dots, 1)^T.$$
Show that $\eta_k > 10^{14}$ before it starts to decrease after 25 iterations. What is the smallest k for which $\|x^{(0)}k\|_2 < \|x^{(0)}\|_2$?

(b) Compute the eigendecomposition $B = X\Lambda X^{-1}$ and determine the condition number $\kappa = \|X\|_2\|X^{-1}\|_2$ of the transformation.

10.2 Successive Overrelaxation Methods

10.2.1 The SOR Method

It was noted early that great improvement in the rate of convergence could be obtained by the simple means of introducing a **relaxation parameter** ω in the Gauss–Seidel method
$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}} r_i^{(k)}, \tag{10.2.1}$$
with $\omega > 1$ (over-relaxation) or $\omega < 1$ (under-relaxation). This lead to the famous **Successive Over Relaxation (SOR) method**, of Young [395], which remained for a long time as a "workhorse" in scientific computing.

Using the standard splitting
$$A = D_A - L_A - U_A = D_A(I - L - U),$$
introduced in Sec. 10.1.3 the SOR method can be written in matrix form as
$$x^{(k+1)} = x^{(k)} + \omega\Big(c + Lx^{(k+1)} - (I-U)x^{(k)}\Big), \tag{10.2.2}$$

where $c = D_A^{-1}b$, or after rearranging

$$(I - \omega L)x^{(k+1)} = [(1-\omega)I + \omega U]x^{(k)} + \omega c.$$

The iteration matrix for SOR therefore is

$$B_\omega = (I - \omega L)^{-1}[(1-\omega)I + \omega U]. \tag{10.2.3}$$

We now consider the convergence of the SOR method and first show that only values of ω, $0 < \omega < 2$ are of interest.

Lemma 10.2.1.
Let $B = L + U$ be any matrix with zero diagonal and let B_ω be given by (10.2.3). Then

$$\rho(B_\omega) \geq |\omega - 1|, \tag{10.2.4}$$

with equality only if all the eigenvalues of B_ω are of modulus $|\omega - 1|$. Hence the SOR method can only converge for $0 < \omega < 2$.

Proof. Since the determinant of a triangular matrix equals the product of its diagonal elements we have

$$\det(B_\omega) = \det(I - \omega L)^{-1} \det[(1-\omega)I + \omega U] = (1-\omega)^n.$$

Also $\det(B_\omega) = \lambda_1\lambda_2\cdots\lambda_n$, where λ_i are the eigenvalues of B_ω. It follows that

$$\rho(B_\omega) = \max_{1\leq i\leq n} |\lambda_i| \geq |1 - \omega|$$

with equality only if all the eigenvalues have modulus $|\omega - 1|$. $\square$

The following theorem asserts that if A is a positive definite matrix, then the SOR method converges for $0 < \omega < 2$.

Theorem 10.2.2. *For a symmetric positive definite matrix A we have*

$$\rho(B_\omega) < 1, \quad \forall\omega, \quad 0 < \omega < 2.$$

Proof. We defer the proof to Theorem 10.3.4. $\square$

For an important class of matrices an explicit expression for the optimal value of ω can be given. We first introduce the class of matrices with **property A**.

Definition 10.2.3. *The matrix A is said to have property A if there exists a permutation matrix P such that PAP^T has the form*

$$\begin{pmatrix} D_1 & U_1 \\ L_1 & D_2 \end{pmatrix}, \tag{10.2.5}$$

where D_1, D_2 are diagonal matrices.

Equivalently, the matrix $A \in \mathbf{R}^{n\times n}$ has property A if the set $\{1 : n\}$ can be divided into two non-void complementary subsets S and T such that $a_{ij} = 0$ unless $i = j$ or $i \in S$, $j \in T$, or $i \in T$, $j \in S$. For example, for the tridiagonal matrix A

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}, \qquad P^T A P = \begin{pmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{pmatrix}$$

has property A, and we can choose $S = \{1, 3\}$, $T = \{2, 4\}$. Permutation of column 1 and 4 followed by a similar row permutation will give a matrix of the form above.

Definition 10.2.4.
A matrix A with the decomposition $A = D_A(I - L - U)$, D_A nonsingular, is said to be **consistently ordered** *if the eigenvalues of*

$$J(\alpha) = \alpha L + \alpha^{-1} U, \qquad \alpha \neq 0,$$

are independent of α.

A matrix of the form of (10.2.5) is consistently ordered. To show this we note that since

$$J(\alpha) = \begin{pmatrix} 0 & -\alpha^{-1} D_1^{-1} U_1 \\ -\alpha D_2^{-1} L_1 & 0 \end{pmatrix} = -\begin{pmatrix} I & 0 \\ 0 & \alpha I \end{pmatrix} J(1) \begin{pmatrix} I & 0 \\ 0 & \alpha^{-1} I \end{pmatrix},$$

the matrices $J(\alpha)$ and $J(1)$ are similar and therefore have the same eigenvalues. More generally any block-tridiagonal matrix

$$A = \begin{pmatrix} D_1 & U_1 & & & \\ L_2 & D_2 & U_2 & & \\ & L_3 & \ddots & \ddots & \\ & & \ddots & \ddots & U_{n-1} \\ & & & L_n & D_n \end{pmatrix},$$

where D_i are nonsingular *diagonal* matrices is matrix has property A and is consistently ordered. To show this, permute the block rows and columns in the order $1, 3, 5, \ldots, 2, 4, 6, \ldots$.

Theorem 10.2.5.
Let $A = D_A(I - L - U)$ be a consistently ordered matrix. Then if μ is an eigenvalue of the Jacobi matrix so is $-\mu$. Further, to any eigenvalue $\lambda \neq 0$ of the SOR matrix B_ω, $\omega \neq 0$, there corresponds an eigenvalue μ of the Jacobi matrix, where

$$\mu = \frac{\lambda + \omega - 1}{\omega \lambda^{1/2}} \tag{10.2.6}$$

Proof. Since A is consistently ordered the matrix $J(-1) = -L - U = -J(1)$ has the same eigenvalues as $J(1)$. Hence if μ is an eigenvalue so is $-\mu$. If λ is an eigenvalue of B_ω, then $\det(\lambda I - B_\omega) = 0$, or since $\det(I - \omega L) = 1$ for all ω, using (10.2.3)

$$\det[(I - \omega L)(\lambda I - B_\omega)] = \det[\lambda(I - \omega L) - (1 - \omega)I - \omega U] = 0.$$

If $\omega \neq 0$ and $\lambda \neq 0$ we can rewrite this in the form

$$\det\left(\frac{\lambda + \omega - 1}{\omega\lambda^{1/2}} I - (\lambda^{1/2} L + \lambda^{-1/2} U)\right) = 0$$

and since A is consistently ordered it follows that $\det\big(\mu I - (L + U)\big) = 0$, where μ given by (10.2.6). Hence μ is an eigenvalue of $L + U$. □

If we put $\omega = 1$ in (10.2.6) we get $\lambda = \mu^2$. Since $\omega = 1$ corresponds to the Gauss–Seidel method it follows that $\rho(B_{GS}) = \rho(B_J)^2$, which means that Gauss–Seidel's method converges twice as fast as Jacobi's method. for all consistently ordered matrices A

We now state an important result due to Young [395].

Theorem 10.2.6.
Let A be a consistently ordered matrix, and assume that the eigenvalues μ of $B_J = L + U$ are real and $\rho_J = \rho(B_J) < 1$. Then the optimal relaxation parameter ω in SOR is given by

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho_J^2}}. \tag{10.2.7}$$

For this optimal value we have

$$\rho(B_{\omega_{opt}}) = \omega_{opt} - 1. \tag{10.2.8}$$

Proof. (See also Young [396, Section 6.2].) We consider, for a given value of μ in the range $0 < \mu \leq \rho(L + U) < 1$, the two functions of λ,

$$f_\omega(\lambda) = \frac{\lambda + \omega - 1}{\omega}, \qquad g(\lambda, \mu) = \mu\lambda^{1/2}.$$

Here $f_\omega(\lambda)$ is a straight line passing through the points $(1, 1)$ and $(1-\omega, 0)$, and $g(\lambda, \mu)$ a parabola. The relation (10.2.6) can now be interpreted as the intersection of these two curves. For given μ and ω we get for λ the quadratic equation

$$\lambda^2 + 2\Big((\omega - 1) - \frac{1}{2}\mu^2\omega^2\Big)\lambda + (\omega - 1)^2 = 0. \tag{10.2.9}$$

which has two roots

$$\lambda_{1,2} = \frac{1}{2}\mu^2\omega^2 - (\omega - 1) \pm \mu\omega\Big(\frac{1}{4}\mu^2\omega^2 - (\omega - 1)\Big)^{1/2}.$$

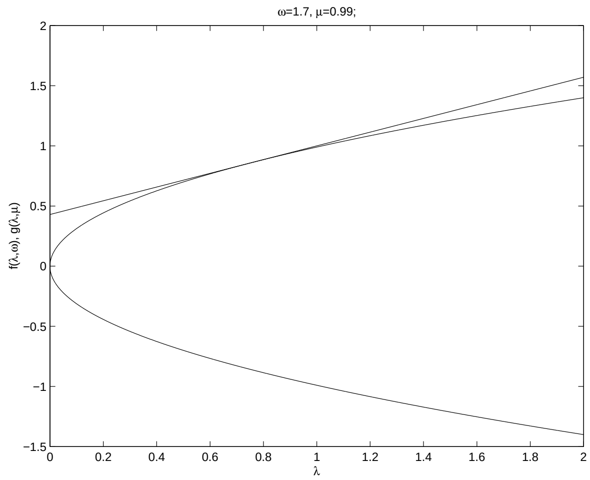

Figure 10.2.1. *$f_\omega(\lambda)$ and $g(\lambda,\mu)$ as functions of λ ($\mu = 0.99$, $\omega = \omega_b = 1.7527$).*

Table 10.2.1. *Number of iterations needed to reduce the initial error by a factor of 10^{-3} for the model problem, as a function of $n = 1/h$.*

n	10	20	50	100	200
Gauss–Seidel	69	279	1,749	6,998	27,995
SOR	17	35	92	195	413

The larger of these roots decreases with increasing ω until eventually $f_\omega(\lambda)$ becomes a tangent to $g(\lambda,\mu)$, when $\mu^2\omega^2/4 - (\omega - 1) = 0$ (see Figure 10.2.1) Solving for the root $\omega \le 2$ gives

$$\tilde{\omega} = \frac{1 - (1-\mu^2)^{1/2}}{1/2\mu^2} = \frac{2}{1+\sqrt{1-\mu^2}}.$$

If $\omega > \tilde{\omega}$, we get two complex roots λ, which by the relation between roots and coefficients in (10.2.9) satisfy

$$\lambda_1\lambda_2 = (\omega - 1)^2.$$

From this it follows that $|\lambda_1| = |\lambda_2| = \omega - 1$, $1 < \tilde{\omega} < \omega < 2$, and hence the minimum value of $\max_{i=1,2}|\lambda_i|$ occurs for $\tilde{\omega}$. Since the parabola $g(\lambda, \rho(L+U))$ is the envelope of all the curves $g(\lambda,\mu)$ for $0 < \mu \le \rho(L+U) < 1$ the theorem follows. $\square$

Example 10.2.1.

By (10.2.7) for SOR $\omega_{opt} = 2/(1+\sin\pi h)$, giving

$$\rho(B_{\omega_{opt}}) = \omega_{opt} - 1 = \frac{1-\sin\pi h}{1+\sin\pi h} \approx 1 - 2\pi h. \tag{10.2.10}$$

Note that when $\lim_{n\to\infty} \omega_{opt} = 2$.

$$R_\infty(B_{\omega_{opt}}) \approx 2\pi h,$$

which shows that for the model problem the number of iterations is proportional to n for the SOR method

In Table 10.1.1 we give the number of iterations required to reduce the norm of the initial error by a factor of 10^{-3}.

In practice, the number ρ_J is seldom known a priori, and its accurate determination would be prohibitively expensive. However, for some model problems the spectrum of the Jacobi iteration matrix is known. In the following we need the result:

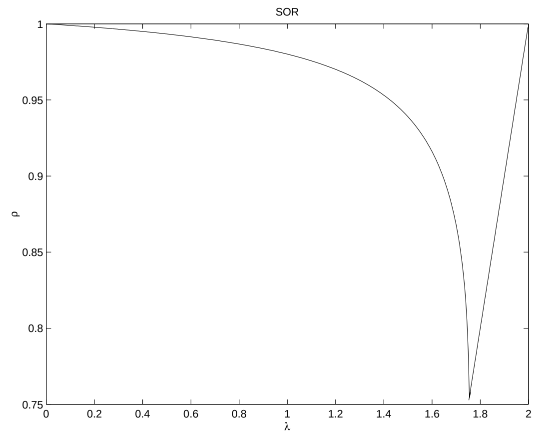

Figure 10.2.2. *The spectral radius* $\rho(B_\omega)$ *as a function of* ω *(*$\rho = 0.99$*,* $\omega_b = 1.7527$*).*

A simple scheme for estimating ω_{opt} is to initially perform a fixed number of iterations using $\omega = 1$, i.e., with the Gauss–Seidel method, and attempt to measure the rate of convergence. The successive corrections satisfy

$$\delta^{(n+1)} = B_{GS}\delta^{(n)}, \qquad \delta^{(n)} = x^{(n+1)} - x^{(n)}.$$

Hence after a sufficient number of iterations we have

$$\rho(B_J)^2 = \rho(B_{GS}) \approx \theta_n, \qquad \theta_n = \|\delta^{(n+1)}\|_\infty / \|\delta^{(n)}\|_\infty,$$

An estimate of ω_{opt} is then obtained by substituting this value into (10.2.7). A closer analysis shows, however, that the number of iterations to obtain a good estimate of

ω_{opt} is comparable to the number of iterations needed to solve the original problem by SOR. The scheme can still be practical if one wishes to solve a number of systems involving the same matrix A. Several variations of this scheme have been developed, see Young [396, p. 210].

In more complicated cases when ρ_J is not known, we have to estimate ω_{opt} in the SOR method. In Figure 10.2.2 we have plotted the spectral radius $\rho(B_\omega)$ as a function of ω in a typical case, where the optimal value is $\omega_b = 1.7527$. We note that the left derivative of $\rho(B_\omega)$ at $\omega = \omega_b$ is infinite. For $\omega \geq \omega_b$, $\rho(B_\omega)$ is a linear function with slope $(1-\omega_b)/(2-\omega_b)$. We conclude that it is better to *overestimate* ω_{opt} than to underestimate it.

10.2.2 The SSOR Method

As remarked above the iteration matrix B_ω of the SOR-method is **not** symmetric and its eigenvalues are not real. In fact, in case ω is chosen slightly larger than optimal (as recommended when ρ_J is not known) the extreme eigenvalues of B_ω lie on a circle in the complex plane. However, a symmetric version of SOR, the (**SSOR**) method of Sheldon (1955), can be constructed as follows. One iteration consists of two half iterations The first half is the same as the SOR iteration. The second half iteration is the SOR method with the equations taken in reverse order. The SSOR method can be written in matrix form as

$$\begin{aligned} x^{(k+1/2)} &= x^{(k)} + \omega\Big(c + Lx^{(k+1/2)} - (I-U)x^{(k)}\Big),\\ x^{(k+1)} &= x^{(k+1/2)} + \omega\Big(c + Ux^{(k+1)} - (I-L)x^{(k+1/2)}\Big). \end{aligned}$$

This method is due to Sheldon [1955]. The iteration matrix for SSOR is

$$S_\omega = (I-\omega U)^{-1}[(1-\omega)I + \omega L](I-\omega L)^{-1}[(1-\omega)I + \omega U].$$

It can be shown that SSOR corresponds to a splitting with the matrix

$$M_\omega = \frac{\omega}{2-\omega}\Big(\frac{1}{\omega}D_A - L_A\Big)D_A^{-1}\Big(\frac{1}{\omega}D_A - U_A\Big). \tag{10.2.11}$$

If A is symmetric, positive definite then so is M_ω. In this case the SSOR method is convergent for all $\omega \in (0,2)$. A proof of this is obtained by a simple modification of the proof of Theorem 10.3.4.

In contrast to the SOR method, the rate of convergence of SSOR is not very sensitive to the choice of ω nor does it assume that A is consistently ordered. It can be shown (see Axelsson [12]) that provided $\rho(LU) < 1/4$ a suitable value for ω is ω_b, where

$$\omega_b = \frac{2}{1+\sqrt{2(1-\rho_J)}}, \qquad \rho(S_{\omega_b}) \leq \frac{1-\sqrt{(1-\rho_J)/2}}{1+\sqrt{(1-\rho_J)/2}}.$$

In particular, for the model problem in Section 10.1.3 it follows that

$$\rho(B_{\omega_b}) \leq \frac{1-\sin \pi h/2}{1+\sin \pi h/2} \approx 1 - \pi h.$$

This is half the rate of convergence for SOR with ω_{opt}.

10.2.3 Block Iterative Methods

The basic iterative methods described so far can be generalized for block matrices A. Assume that A and b are partitioned conformally,

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \dots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

where the diagonal blocks A_{ii} are square and nonsingular. Using this partitioning we consider the splitting

$$A = D_A - L_A - U_A, \qquad D_A = \operatorname{diag}(A_{11}, A_{22}, \dots, A_{nn}),$$

where L_A and U_A are strictly lower and upper triangular. The block Jacobi method can then be written

$$D_A x^{(k+1)} = (L_A + U_A)x^{(k)} + b,$$

or, with x is partitioned conformally,

$$A_{ii}\Big(x_i^{(k+1)} - x_i^{(k)}\Big) = b_i - \sum_{j=1}^{n} A_{ij} x_j^{(k)}, \quad i = 1 : n.$$

For this iteration to be efficient it is important that linear systems in the diagonal blocks A_{ii} can be solved efficiently.

Example 10.2.2.

For the model problem in Section 10.1.3 the matrix A can naturally be written in the block form where the diagonal blocks $A_{ii} = 2I + T$ are tridiagonal and nonsingular, see (10.1.7). The resulting systems can be solved with little overhead. Note that here the partitioning is such that x_i corresponds to the unknowns at the mesh points on the ith line. Hence block methods are in this context also known as "line" methods and the other methods as "point" methods.

Block versions of the Gauss–Seidel, SOR, and SSOR methods are developed similarly. For SOR we have

$$A_{ii}\Big(x_i^{(k+1)} - x_i^{(k)}\Big) = \omega\Big(b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} A_{ij} x_j^{(k)}\Big), \quad i = 1 : n.$$

(Taking $\omega = 1$ gives the Gauss–Seidel method.) Typically the rate of convergence is improved by a factor $\sqrt{2}$ compared to the point methods.

It can easily be verified that the SOR theory as developed in Theorems 10.2.2 and 10.2.5 are still valid in the block case. We have

$$B_\omega = (I - \omega L)^{-1}\big[(1-\omega)I + \omega U\big],$$

where $L = D_A^{-1}L_A$ and $U = D_A^{-1}U_A$. Let A be a consistently ordered matrix with nonsingular diagonal blocks A_{ii}, $1 \le i \le n$. Assume that the block Jacobi matrix B has spectral radius $\rho(B_J) < 1$. Then the optimal value of ω in the SOR method is given by (10.2.7). Note that with the block splitting any block-tridiagonal matrix

$$A = \begin{pmatrix} D_1 & U_1 & & & \\ L_2 & D_2 & U_2 & & \\ & L_3 & \ddots & \ddots & \\ & & \ddots & \ddots & U_{n-1} \\ & & & L_n & D_n \end{pmatrix},$$

is consistently ordered; for the point methods this was true only in case the block diagonal matrices D_i, $i = 1 : n$ were diagonal. In particular we conclude that with the block splitting the matrix A in (10.1.2) for the model problem is consistently ordered so the SOR theory applies.

10.2.4 Chebyshev Acceleration

In the non-stationary Richardson iteration

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 1, 2, \dots,$$

where $\omega_k > 0$ are parameters to be chosen (cf. (10.1.1)). The residual vector $r^{(k)} = b - Ax^{(k)}$ satisfies

$$r^{(k)} = (I - \omega_{k-1}A)(I - \omega_{k-2}A)\cdots(I - \omega_0 A)r^{(0)} = q_k(A)r^{(0)}. \tag{10.2.12}$$

Here $q_k(z)$ is a polynomial of degree k with the property that $q_k(0) = 1$. The polynomial $q_k(x)$ is known as a **residual polynomial**. We can obtain any desired residual polynomial by choosing its roots as the parameters $\{\omega_i\}_{i=0}^{k-1}$. This process is called **polynomial acceleration**.

The most important case is Chebyshev acceleration, which we now develop. Assume that the eigenvalues $\{\lambda_i\}_{i=1}^n$ of A are real and satisfy

$$0 < a \le \lambda_i < b. \tag{10.2.13}$$

From (10.2.12) we get the estimate

$$\|r^{(k)}\| = \|q_k(A)\| \, \|r^{(0)}\|,$$

If A is a Hermitian matrix, then so is $q_k(A)$ and $\|q_k(A)\|_2 = \rho(q_k(A))$. Then, after k steps of the accelerated method the 2-norm of the residual is reduced by at least a factor of

$$\rho(q_k(A)) = \max_i |q_k(\lambda_i)| \le \max_{\lambda\in[a,b]} |q_k(\lambda)|.$$

Therefore a suitable polynomial q_k can be obtained by solving the minimization problem

$$\min_{q\in\Pi_k^1} \max_{\lambda\in[a,b]} |q(\lambda)|, \tag{10.2.14}$$

where Π_k^1 denotes the set of residual polynomials q_k of degree $\leq k$ such that $q_k(0) = 1$. The Chebyshev polynomials are defined by $T_k(z) = \cos k\phi$, $z = \cos\phi$, are known to have the minimax property that *of all nth degree polynomials with leading coefficient* 1, *the polynomial* $2^{1-n}T_n(x)$ *has the smallest magnitude* 2^{1-n} *in* $[-1,1]$. (A proof is given in Vol. I, Sec. 3.2.3.) It follows that the solution to the above minimization problem (10.2.14) is given by the shifted and normalized Chebyshev polynomials

$$q_k(\lambda) = T_k(z(\lambda))/T_k(z(0)), \tag{10.2.15}$$

where $T_k(z)$ is the Chebyshev polynomial of degree k and

$$z(\lambda) = \frac{b+a-2\lambda}{b-a} = \mu - \frac{2}{b-a}\lambda, \quad \mu = z(0) = \frac{b+a}{b-a} > 1, \tag{10.2.16}$$

which maps the interval $0 < a \leq \lambda \leq b$ onto $z \in [-1,1]$. Since $|T_k(z(\lambda))| \leq 1$, $\lambda \in [a,b]$ and $T_k(\mu) > 1$ we have

$$\rho(q_k(A)) \leq 1/T_k(\mu) < 1,$$

Setting $w = e^{i\phi} = \cos\phi + i\sin\phi$, we have $z = \frac{1}{2}(w + w^{-1})$, and

$$T_k(z) = \tfrac{1}{2}(w^k + w^{-k}), \quad w = z \pm \sqrt{z^2-1}, \quad -\infty < z < \infty. \tag{10.2.17}$$

From (10.2.16) it follows that $\mu = (\kappa+1)/(\kappa-1)$, where $\kappa = b/a$ is an upper bound for the spectral condition number of A. Then

$$w = \mu + \sqrt{\mu^2-1} = \frac{\kappa+1}{\kappa-1} + \frac{2\sqrt{\kappa}}{\kappa-1} = \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} > 1,$$

$\rho(q_k(A)) \leq 1/T_k(\mu) > 2e^{-k\gamma}$, where after some simplification

$$\gamma = \log\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right) > \frac{2}{\sqrt{\kappa}}.$$

(Verify the last inequality! See Problem 2.) Hence to reduce the error norm at least by a factor of $\delta < 1$ it suffices to perform k iterations, where

$$k > \frac{1}{2}\sqrt{\kappa}\log\frac{2}{\delta}, \tag{10.2.18}$$

Thus the number of iterations required for a certain accuracy for the Chebyshev accelerated method is proportional to $\sqrt{\kappa}$ rather than κ—a great improvement!

Since the zeros of the Chebyshev polynomials $T_k(z)$ are known it is possible to implement Chebyshev acceleration as follows. To perform N steps we compute

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 0 : N-1, \tag{10.2.19}$$

where

$$\omega_k = 2\big[(b+a) - (b-a)\cos\big((k+\tfrac{1}{2})/N\big)\big]^{-1}, \quad k = 0 : N-1. \tag{10.2.20}$$

After N steps the iterations can be repeated in a cyclic fashion. (Note that for $N = 1$ we retrieve the optimal ω for the stationary Richardson's method derived in Section 10.1.4.) Unfortunately, this scheme is known to be unstable unless N is small. The instability can be cured by reordering the roots; see Problem 10. However, one disadvantage remains, namely, the number N has to be fixed in advance.

A stable way to implement Chebyshev acceleration is based on the three term recurrence relation (see Sec. 3.2.3) $T_0(z) = 1$, $T_1(z) = zT_0$,

$$T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z), \quad k \geq 1, \tag{10.2.21}$$

valid for the Chebyshev polynomials. By (10.2.15) $T_k(z(\lambda)) = T_k(\mu)q_k(\lambda)$, and using (10.2.16) and substituting A for λ, we obtain from (10.2.21)

$$T_{k+1}(\mu)q_{k+1}(A) = 2\Big(\mu I - \frac{2}{b-a}A\Big)T_k(\mu)q_k(A) - T_{k-1}(\mu)q_{k-1}(A).$$

Multiplying by $(x^{(0)} - x)$, and using that $q_k(A)(x^{(0)} - x) = x^{(k)} - x$, we obtain

$$T_{k+1}(\mu)(x^{(k+1)} - x) = 2\Big(\mu I - \frac{2}{b-a}A\Big)T_k(\mu)(x^{(k)} - x) - T_{k-1}(\mu)(x^{(k-1)} - x).$$

Subtracting (10.2.21) with $z = \mu$ it then follows that

$$T_{k+1}(\mu)x^{(k+1)} = 2\mu T_k(\mu)x^{(k)} + \frac{4T_k(\mu)}{b-a}A(x - x^{(k)}) - T_{k-1}(\mu)x^{(k-1)}.$$

Substituting $-T_{k-1}(\mu) = -2\mu T_k(\mu) + T_{k+1}(\mu)$ and dividing with $T_{k+1}(\mu)$ we get

$$x^{(k+1)} = x^{(k-1)} + \delta_k(b - Ax^{(k)}) + \omega_k(x^{(k)} - x^{(k-1)}), \quad k \geq 1,$$

where

$$\alpha = 2/(b+a), \qquad \omega_k = 2\mu\frac{T_k(\mu)}{T_{k+1}(\mu)}, \qquad \delta_k = \alpha\omega_k.$$

This is a three term recurrence for computing the accelerated approximate solution. A similar calculation for $k = 0$ gives $x^{(1)} = x^{(0)} + \alpha(b - Ax^{(0)})$. and we have derived the following algorithm:

The Chebyshev Semi-Iterative Method;

Assume that the eigenvalues $\{\lambda_i\}_{i=1}^n$ of A are real and satisfy $0 < a \leq \lambda_i < b$. Set

$$\mu = (b+a)/(b-a), \quad \alpha = 2/(b+a).$$

and compute $x^{(1)} = x^{(0)} + \alpha(b - Ax^{(0)})$,

$$x^{(k+1)} = x^{(k-1)} + \omega_k\big(\alpha(b - Ax^{(k)}) + x^{(k)} - x^{(k-1)}\big), \quad k = 1, 2, \ldots, \tag{10.2.22}$$

where

$$\omega_0 = 2, \qquad \omega_k = \left(1 - \frac{\omega_{k-1}}{4\mu^2}\right)^{-1}, \quad k \geq 1.$$

Chebyshev acceleration can more generally be applied to any stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots, \tag{10.2.23}$$

provided it is **symmetrizable**. The iteration (10.2.23) corresponds to a matrix splitting $A = M - N$, and iteration matrix $B = M^{-1}N = I - M^{-1}A$.

Definition 10.2.7. *The stationary iterative method* (10.2.23) *is said to be symmetrizable if there is a nonsingular matrix W such that the matrix $W(I - B)W^{-1}$ is symmetric and positive definite.*

For a symmetrizable method the matrix $M^{-1}A$ has real positive eigenvalues λ_i. To apply Chebyshev acceleration we assume that $0 < a \le \lambda_i \le b$ and substitute $M^{-1}(b - Ax^{(k)})$ for the residual in the algorithm given above. Here the matrix M should be chosen as a **preconditioner**, i.e. so that the spectral condition number of $M^{-1}A$ is reduced.

A sufficient condition for a method to be symmetrizable is that both A and the splitting matrix M are symmetric, positive definite, since then there is a matrix W such that $M = W^TW$, and

$$W(I - B)W^{-1} = WM^{-1}AW^{-1} = WW^{-1}W^{-T}AW^{-1} = W^{-T}AW^{-1},$$

which again is positive definite.

Example 10.2.3.

If A is positive definite then in the standard splitting (10.1.15) $D_A > 0$, and hence the Jacobi method is symmetrizable with $W = D_A^{1/2}$. From (10.2.11) it follows that also the SSOR method is symmetrizable.

The eigenvalues of the iteration matrix of the SOR-method $B_{\omega_{opt}}$ are all complex and have modulus $|\omega_{opt}|$. Therefore in this case convergence acceleration is of no use. (A precise formulation is given in Young [396, p. 375].) However, Chebyshev acceleration can be applied to the Jacobi and SSOR methods, with

$$M_J = D_A, \qquad M_\omega = \frac{\omega}{2-\omega}\Big(\frac{1}{\omega}D_A - L_A\Big)D_A^{-1}\Big(\frac{1}{\omega}D_A - U_A\Big),$$

respectively, as well as block versions of these methods, often with a substantial gain in convergence rate.

that $M = I$ we obtain for the residual $\tilde{r}^{(k)} = b - A\tilde{x}^{(k)}$

$$\tilde{r}^{(k)} = A(x - \tilde{x}^{(k)}) = q_k(A)r^{(0)}, \qquad q_k(\lambda) = p_k(1 - \lambda). \tag{10.2.24}$$

where we have used that $A = I - B$.

The main drawback of Chebyshev acceleration is that it requires a fairly accurate knowledge of an interval $[a, b]$ enclosing the (real) spectrum of A. If this enclosure is too crude the process loses efficiency. In Section 10.2.2 we describe a method which converges at least as fast as Chebyshev semi-iteration and does not need an estimate of the spectrum.

Review Questions

1. When is the matrix A reducible? Illustrate this property using the directed graph of A.

2. Let $A = D_A(I - L - U)$, where $D_A > 0$. When is A said to have "property A". When is A consistently ordered? How are these properties related to the SOR method?

3. For the model problem the asymptotic rate of convergence for the classical iterative methods is proportional to h^p, where h is the mesh size. Give the value of p for Jacobi, Gauss–Seidel, SOR and SSOR. (For the last two methods it is assumed that the optimal ω is used.

4. Consider an iterative method based on the splitting $A = M - N$. Give conditions on the eigenvalues of $M^{-1}A$ which are sufficient for Chebyshev acceleration to be used. Express the asymptotic rate of convergence for the accelerated method in terms of eigenvalue bounds.

Problems and Computer Exercises

5. (a) Show that if A is reducible so is A^T. Which of the following matrices are irreducible?

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

(b) Is it true that a matrix A, in which the elements take the values 0 and 1 only, is irreducible if and only if the non-decreasing matrix sequence $(I + A)^k$, $k = 1, 2, 3, \dots$ becomes a full matrix for some value of k?

6. The matrix A in (10.1.7) is block-tridiagonal, but its diagonal blocks are *not* diagonal matrices. Show that in spite of this the matrix is consistently ordered.

Hint: Perform a similarity transformation with the diagonal matrix

$$D(\alpha) = \mathrm{diag}\,(D_1(\alpha), D_2(\alpha), \dots, D_n(\alpha)),$$

where $D_1(\alpha) = \mathrm{diag}\,(1, \alpha, \dots, \alpha^{n-1})$, $D_{i+1}(\alpha) = \alpha D_i(\alpha)$, $i = 1 : n - 1$.

7. Verify the recursion for ω_k for the Chebyshev semi-iteration method.

8. Show that

$$\log\bigl((1 + s)/(1 - s)\bigr) = 2(s + s^3/3 + s^5/5 + \dots), \quad 0 \le s < 1,$$

and use this result to prove (10.2.18).

9. Assume that A is symmetric indefinite with its eigenvalues contained in the union of two intervals of equal length,

$$\mathcal{S} = [a, b] \cup [c, d], \quad a < b < 0, \quad 0 < c < d,$$

where $d - c = b - a$. Then the Chebyshev semi-iterative method cannot be applied directly to the system $Ax = b$. Consider instead the equivalent system

$$Bx = c, \qquad B = A(A - \alpha I), \qquad c = Ab - \alpha b.$$

(a) Show that if $\alpha = d + a = b + c$, then the eigenvalues of B are positive and real and contained in the interval $[-bc, -ad]$.
(b) Show that the matrix B has condition number

$$\kappa(B) = \frac{d}{c} \cdot \frac{|a|}{|b|} = \frac{d}{c}\frac{d - c + |b|}{|b|}.$$

Use this to give estimates for the two special cases (i) Symmetric intervals with respect to the origin. (ii) The case when $|b| \gg c$.

10. Let A be a matrix with real eigenvalues $\{\lambda_i\}_{i=1}^n$, $0 < a \leq \lambda_i < b$. Then the Chebyshev semi-iterative method for solving $Ax = b$ can be implemented by the recursion (10.2.19)–(10.2.20). The instability of this scheme can be eliminated by using an ordering of the iteration parameters ω_k given by Lebedev and Finogenov. For $N = 2^p$ this permutation ordering κ is constructed by the following MATLABprogram:

```
% A function file to compute the Lebedev-Finogenov ordering
      N = 2^p; int = 1;
      kappa = ones(1,N);
      for i = 1:p
         int = 2*int; ins = int+1;
         for j = int/2:-1:1
            kappa(2*j)=ins - kappa(j);
            kappa(2*j-1) = kappa(j);
         end;
       end;
```

Implement and test this method using the system $Ax = b$ from the Laplace equation on the unit square, with A block tridiagonal

$$A = \text{tridiag}(-I, T + 2I, -I) \in \mathbf{R}^{n^2 \times n^2}, \quad T = \text{tridiag}(-1, 2 - 1) \in \mathbf{R}^{n \times n}.$$

Construct the right hand so that the exact solution becomes $x = (1, 1, \ldots, 1, 1)^T$. Let $x^{(0)} = 0$ as initial approximation and solve this problem using

- The implementation based on the three term recursion of Chebyshev polynomials
- Richardson implementation with natural ordering of the parameters
- Richardson implementation with the Lebedev–Finogenov ordering of the parameters

Take $n = 50$ and $N = 128$. Use the same number of iterations in all three implementations. List in each case the maximum norm of the error and the residual. Compare the results and draw conclusions!

10.3 Projection Methods

10.3.1 General Principles

Consider a linear system $Ax = b$, where $A \in \mathbf{R}^{n\times n}$. Suppose we want to find an approximate solution $\hat{x}$ in a subspace $\mathcal{K}$ of dimension $m < n$. Then m independent conditions are needed to determine $\hat{x}$. One way to obtain these is by requiring that the residual $b - A\hat{x}$ is orthogonal to a subspace $\mathcal{L}$ of dimension m, i.e.,

$$\hat{x} \in \mathcal{K}, \qquad b - A\hat{x} \perp \mathcal{L}. \tag{10.3.1}$$

Many important classes of iterative methods can be interpreted as being projection methods in this general sense. The conditions (10.3.1) are often known as **Petrov–Galerkin conditions**.

We can obtain a matrix form of (10.3.1) by introducing basis vectors in the two subspaces. If we let

$$\mathcal{K} = \mathcal{R}(U), \qquad \mathcal{L} = \mathcal{R}(V), \tag{10.3.2}$$

where $U = (u_1, \dots, u_m)$, $V = (v_1, \dots, v_m)$, then we can write (10.3.1) as

$$V^T(b - AUz) = 0, \quad z \in \mathbf{R}^m. \tag{10.3.3}$$

where $\hat{x} = Uz$. Hence $\hat{x}$ is obtained by solving the reduced system

$$\hat{A}z = V^T b, \quad \hat{A} = V^T AU \in \mathbf{R}^{m\times m}. \tag{10.3.4}$$

We usually have $m \ll n$, and when m is small this system can be solved by a direct method.

Example 10.3.1.

Even though A is nonsingular the matrix $\hat{A}$ may be singular. For example, take, $m = 1$, $U = V = e_1$, and

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

Then $\hat{A} = 0$. Note that the matrix A here is symmetric, but not positive definite.

There are two important special cases in which the matrix $\hat{A}$ can be guaranteed to be nonsingular.

1. Let A by symmetric, positive definite (s.p.d.) and $\mathcal{L} = \mathcal{K}$. Then we can take $V = U$, and have $\hat{A} = U^T AU$. Clearly $\hat{A}$ is s.p.d., and hence nonsingular. By (10.3.3)
$$\hat{x} = Uz, \quad z = (U^T AU)^{-1} U^T b.$$

2. Let A be nonsingular and $\mathcal{L} = A\mathcal{K} \equiv \mathcal{R}(AU)$. Then we get $\hat{A} = (AU)^T(AU)$ which is s.p.d. In this case
$$\hat{x} = Uz, \quad z = (U^T A^T AU)^{-1} U^T A^T b.$$

We now derive important optimality properties satisfied in these two special cases. For this purpose we first define a new inner product and norm related to a s.p.d. matrix A.

Definition 10.3.1. *For an s.p.d. matrix A we define a related A-inner product and A-norm, also called the* **energy norm**[49]*, by*

$$(u, v)_A = u^T A v, \qquad \|u\|_A = (u^T A u)^{1/2}, \tag{10.3.5}$$

It is easily verified that $\|u\|_A$ satisfies the conditions for a norm.

Lemma 10.3.2.

Let A by symmetric, positive definite and consider the case $\mathcal{L} = \mathcal{K}$ ($V = U$). Then $\hat{x} = U(U^T AU)^{-1}U^T b$ minimizes the energy norm of the error over all vectors $x \in \mathcal{K}$, i.e., $\hat{x}$ solves the problem

$$\min_{x\in\mathcal{K}} \|x - x^*\|_A, \quad x^* = A^{-1}b. \tag{10.3.6}$$

Proof. By (10.3.3) $\hat{x}$ satisfies $v^T(b - A\hat{x}) = 0$, $\forall v \in \mathcal{K}$. Let $\hat{e} = \hat{x} - x^*$ be the error in $\hat{x}$. Then for the error in $\hat{x} + v$, $v \in \mathcal{K}$ we have $e = \hat{e} + v$, and

$$\|e\|_A^2 = \hat{e}^T A\hat{e} + v^T Av + 2v^T A\hat{e}.$$

But here the last term is zero because $v^T A\hat{e} = v^T(A\hat{x} - b) = 0$. It follows that $\|e\|_A$ is minimum if $v = 0$. $\square$

A related result is obtained for the second case.

Lemma 10.3.3.

Let A be nonsingular and consider the case $\mathcal{L} = A\mathcal{K}$, ($V = AU$). Then $\hat{x} = U(U^T A^T AU)^{-1}U^T A^T b$ minimizes the 2-norm of the residual over all vectors $x \in \mathcal{K}$, i.e.,

$$\min_{x\in\mathcal{K}} \|b - Ax\|_2. \tag{10.3.7}$$

Proof. Using $x = Uz$ we have $\|b - Ax\|_2 = \|b - AUz\|_2$, which is minimized when z satisfies the normal equations $U^T A^T AUz = U^T A^T b$. This gives the desired result. $\square$

In an iterative method often *a sequence of projection steps* of the above form is taken. Then we need to modify the above algorithms slightly so that they can start from a given approximation x_k.[50]

[49] For some PDE problems this name has a physical relevance.

[50] In the rest of this chapter we will use vector notations and x_k will denote the kth approximation and not the kth component of x.

If we let $x = x_k + z$, then z satisfies the system $Az = r_k$, where $r_k = b - Ax_k$. In step k we now apply the above projection method to this system. Hence we require that $z \in \mathcal{K}$ and that $r_k - Az = b - A(x_k + z) \perp \mathcal{L}$. This gives the equations

$$r_k = b - Ax_k, \qquad z = (V^T AU)^{-1} V^T r_k, \qquad x_{k+1} = x_k + Uz. \tag{10.3.8}$$

for computing the new approximation x_{k+1}. A generic projection algorithm is obtained by starting from some x_0 (e.g., $x_0 = 0$), and repeatedly perform (10.3.8) for a sequence of subspaces $\mathcal{L} = \mathcal{L}_k$, $\mathcal{K} = \mathcal{K}_k$, $k = 1, 2, \ldots$.

10.3.2 The One-Dimensional Case

The simplest case of a projection method is when $m = 1$. Starting from some x_0, we take a sequence of steps. In the kth step we take $\mathcal{L}_k = \operatorname{span}(v_k)$, $\mathcal{K}_k = \operatorname{span}(u_k)$, and update x_k by

$$r_k = b - Ax_k, \quad \alpha_k = \frac{v_k^T r_k}{v_k^T A u_k}, \quad x_{k+1} = x_k + \alpha_k u_k, \tag{10.3.9}$$

where we have to require that $v_k^T A u_k \neq 0$. By construction the new residual r_{k+1} is orthogonal to v_k. Note that r_k can be computed recursively from

$$r_k = r_{k-1} - \alpha_{k-1} A u_{k-1}. \tag{10.3.10}$$

This expression is obtained by multiplying $x_k = x_{k-1} + \alpha_{k-1} u_{k-1}$ by A and using the definition $r_k = b - Ax_k$. Since Au_{k-1} is needed for computing α_{k-1} using the recursive residual will save one matrix times vector multiplication.

If A is s.p.d. and we take $v_k = u_k$, the above formulas become

$$r_k = b - Ax_k, \quad \alpha_k = \frac{u_k^T r_k}{u_k^T A u_k}, \quad x_{k+1} = x_k + \alpha_k u_k, \tag{10.3.11}$$

In this case x_{k+1} minimizes the quadratic functional

$$\phi(x) = \|x - x^*\|_A^2 = (x - x^*)^T A (x - x^*) \tag{10.3.12}$$

for all vectors of the form $x_k + \alpha_k u_k$.

The vectors u_k are often called **search directions**. Expanding the function $\phi(x_k + \alpha u_k)$ with respect to α, we obtain

$$\phi(x_k + \alpha u_k) = \phi(x_k) - \alpha u_k^T (b - Ax_k) + \frac{1}{2}\alpha^2 u_k^T A u_k. \tag{10.3.13}$$

Taking $\alpha = \omega \alpha_k$ where α_k is given by (10.3.11) we obtain

$$\phi(x_k + \omega \alpha_k u_k) = \phi(x_k) - \rho(\omega) \frac{(u_k^T r_k)^2}{u_k^T A u_k}, \quad \rho(\omega) = \frac{1}{2}\omega(2 - \omega), \tag{10.3.14}$$

which is a quadratic function of ω. In a projection step ($\omega = 1$) the line $x_k + \alpha u_k$ is tangent to the ellipsoidal level surface $\phi(x) = \phi(x_{k+1})$, and $\phi(x_k + \alpha_k u_k) < \phi(x_k)$ provided that $u_k^T r_k \neq 0$. More generally, if $u_k^T r_k \neq 0$ we have from symmetry that

$$\phi(x_k + \omega\alpha_k u_k) < \phi(x_k), \qquad 0 < \omega < 2.$$

For the error in $x_{k+1} = x_k + \omega\alpha_k u_k$ we have

$$\hat{x} - x_{k+1} = \hat{x} - x_k - \omega \frac{u_k^T r_k}{u_k^T A u_k} u_k = \left(I - \omega \frac{u_k u_k^T}{u_k^T A u_k} A \right) (\hat{x} - x_k).$$

This shows that the error in each step is transformed by a linear transformation $\hat{x} - x_{k+1} = B(\omega)(\hat{x} - x_k)$.

Example 10.3.2.

For the Gauss–Seidel method in the ith minor step the ith component of the current approximation x_k is changed so that the ith equation is satisfied, i.e., we take

$$x_k := x_k - \hat{\alpha} e_i, \qquad e_i^T (b - A(x_k - \hat{\alpha} e_i)) = 0,$$

where e_i is the ith unit vector. Hence the Gauss–Seidel method is equivalent to a sequence of one-dimensional modifications where the search directions are chosen equal to the unit vectors in cyclic order $e_1, \ldots, e_n, e_1, \ldots, e_n, \ldots$.

This interpretation can be used to prove convergence for the Gauss–Seidel (and more generally the SOR method) for the case when A is s.p.d..

Theorem 10.3.4.

If A is symmetric, positive definite then the SOR method converges for $0 < \omega < 2$, to the unique solution of $Ax = b$. In particular the Gauss–Seidel method, which corresponds to $\omega = 1$, converges.

Proof. In a minor step using search direction e_i the value of ϕ will decrease unless $e_i^T (b - Ax_k) = 0$, i.e., unless x_k satisfies the ith equation. A major step consists of a sequence of n minor steps using the search directions $e_1, \ldots, e_n$. Since each minor step effects a linear transformation of the error $y_k = \hat{x} - x_k$, in a major step it holds that $y_{k+1} = By_k$, for some matrix B. Here $\|By_k\|_A < \|y_k\|_A$ unless y_k is unchanged in *all* minor steps, $i = 1, \ldots, n$, which would imply that $y_k = 0$. Therefore if $y_k \neq 0$, then $\|By_k\|_A < \|y_k\|_A$, and thus $\|B\|_A < 1$. It follows that

$$\|B^n y_0\|_A \leq \|B\|_A^n \, \|y_0\|_A \to 0 \quad \text{when} \quad n \to \infty,$$

i.e., the iteration converges.

If we define the minor step as $x := \omega\hat{\alpha} e_i$, where ω is a fix relaxation factor, the convergence proof also holds. (We may even let ω vary with i, although the proof assumes that ω for the same i has the same value in all major steps.) This shows that the SOR method is convergent and by Theorem 10.1.7 this is equivalent to $\rho(B_\omega) < 1$, $0 < \omega < 2$. $\square$

We make two remarks about the convergence proof. First, it also holds if for the basis vectors $\{e_i\}_{i=1}^n$ we substitute an *arbitrary set of linearly independent vectors* $\{p_j\}_{i=1}^n$. Second, if A is a positive diagonal matrix, then we obtain the *exact* solution by the Gauss–Seidel method after n minor steps. Similarly, if A assumes diagonal form after a coordinate transformation with, $P = (p_1, \dots, p_n)$, i.e., if $P^TAP = D$, then the exact solution will be obtained in n steps using search directions $p_1, \dots, p_n$. Note that this condition is equivalent to the requirement that the vectors $\{p_j\}_{i=1}^n$ should be A-orthogonal, $p_i^TAp_j = 0$, $i \neq j$.

10.3.3 The Method of Steepest Descent

Assume that A is s.p.d. and consider the error functional (10.3.12). From the expansion (10.3.13) it is clear that the negative gradient of $\phi(x)$ with respect to x equals $-\nabla\phi(x) = b - Ax$. Hence the direction in which the function ϕ decreases most rapidly at the point x_k equals the residual $r_k = b - Ax_k$. The **method of steepest descent**[51] is a one-dimensional projection method with $v_k = u_k = r_k$. This leads to the iteration

$$r_k = b - Ax_k, \quad \alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}, \quad x_{k+1} = x_k + \alpha_k r_k. \tag{10.3.15}$$

It follows from (10.3.14) that when $r_k \neq 0$, it holds that $\phi(x_{k+1}) < \phi(x_k)$.

We now derive an expression for the rate of convergence of the steepest descent method. Denoting the error in x_k by $e_k = x_k - x^*$ we have

$$\begin{aligned}\|e_{k+1}\|_A^2 &= e_{k+1}^T A e_{k+1} = -r_{k+1}^T e_{k+1} = -r_{k+1}^T(e_k + \alpha_k r_k)\\ &= -(r_k - \alpha_k A r_k)^T e_k = e_k^T A e_k - \alpha_k r_k^T r_k,\end{aligned}$$

where we have used that $r_{k+1}^T r_k = 0$. Using the expression (10.3.15) for α_k we obtain

$$\|e_{k+1}\|_A^2 = \|e_k\|_A^2 \left(1 - \frac{r_k^T r_k}{r_k^T A r_k} \frac{r_k^T r_k}{r_k^T A^{-1} r_k}\right). \tag{10.3.16}$$

To estimate the right hand side we need the following result.

Lemma 10.3.5 (Kantorovich[52] inequality).

Let A be a real symmetric matrix with eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$. Then for any vector x it holds

$$\frac{(x^TAx)(x^TA^{-1}x)}{(x^Tx)^2} \leq \frac{1}{4}\left(\kappa^{1/2} + \kappa^{-1/2}\right)^2, \tag{10.3.17}$$

where $\kappa = \lambda_n/\lambda_1$ is the condition number of A.

Proof. (After D. Braess) Let $\mu = (\lambda_1\lambda_n)^{1/2}$ be the geometric mean of the eigenvalues and consider the symmetric matrix $B = \mu^{-1}A + \mu A^{-1}$. The eigenvalues of

[51] This method is attributed to Cauchy 1847.

[52] Leonid V. Kantorovich, 1912–1986, Moscow, 1975 Nobel Laureate in Economics.

B satisfy

$$\lambda_i(B) = \mu^{-1}\lambda_i + \mu\lambda_i^{-1} \le \kappa^{1/2} + \kappa^{-1/2}, \quad i = 1 : n.$$

Hence, by the Courant maximum principle, for any vector x it holds

$$x^T Bx = \mu^{-1}(x^T Ax) + \mu(x^T A^{-1}x) \le (\kappa^{1/2} + \kappa^{-1/2})(x^T x).$$

The left hand can be bounded using the simple inequality

$$(ab)^{1/2} \le \tfrac{1}{2}(\mu^{-1}a + \mu b), a, b > 0.$$

Squaring this and taking $a = x^T Ax$ and $b = x^T A^{-1}x$ the lemma follows. □

From (10.3.16) and Kantorovich's inequality it follows for the method of steepest descent that

$$\|e_{k+1}\|_A^2 \le \|e_k\|_A^2 \left(\frac{\kappa^{1/2} - \kappa^{-1/2}}{\kappa^{1/2} + \kappa^{-1/2}}\right)^2,$$

and hence

$$\|x - x_k\|_A \le \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x\|_A. \tag{10.3.18}$$

It can also be shown that asymptotically this bound is sharp. Hence, the asymptotic rate of convergence only depends *on the extreme eigenvalues of* A.

If the matrix A is ill-conditioned the level curves of ϕ are very elongated hyper-ellipsoids. Then the successive iterates x_k, $k = 0, 1, 2, \ldots$ will zig-zag slowly towards the minimum $x = A^{-1}b$ as illustrated in Figure 10.3.1 for a two dimensional case. Note that successive search directions are orthogonal.

Figure 10.3.1. *Convergence of the steepest descent method.*

Now consider the more general method where $u_0 = p_0 = r_0$ (i.e., the steepest descent direction) and the search directions $u_{k+1} = p_{k+1}$ are taken to be a linear combination of the negative gradient r_{k+1} and the previous search direction p_k, i.e.,

$$p_{k+1} = r_{k+1} + \beta_k p_k, \quad k = 0, 1, 2 \ldots. \tag{10.3.19}$$

Here the parameter β_k remains to be determined. (Note that $\beta_k = 0$ gives the method of steepest descent.) From (10.3.14) we know that to get $\phi(x_{k+1}) < \phi(x_k)$ we must have $p_k^T r_k \neq 0$. Replacing $(k+1)$ by k in (10.3.19) and multiplying by r_k^T, we obtain

$$r_k^T p_k = r_k^T r_k + \beta_{k-1} r_k^T p_{k-1} = r_k^T r_k, \tag{10.3.20}$$

since r_k is orthogonal to p_{k-1}. It follows that $r_k^T p_k = 0$ implies $r_k = 0$ and thus $x_k = A^{-1}b$. Hence unless x_k is the solution, the next iteration step is always defined, regardless of the value of the parameter β_k, and $\phi(x_{k+1}) < \phi(x_k)$. From (10.3.20) we also obtain the alternative expression

$$\alpha_k = (r_k^T r_k)/(p_k^T A p_k). \tag{10.3.21}$$

We shall discuss the choice of the parameter β_k in the next section.

Review Questions

1. Let $\phi(x) = \frac{1}{2}x^T Ax - x^T b$, where A is symmetric positive definite, and consider the function $\varphi(\alpha) = \phi(x_k + \alpha p_k)$, where p_k is a search direction and x_k the current approximation to $x = A^{-1}b$. For what value $\alpha = \alpha_k$ is $\varphi(\alpha)$ minimized? Show that for $x_{k+1} = x_k + \alpha_k p_k$ it holds that $b - Ax_{k+1} \perp p_k$.
2. Show that minimizing the quadratic form $\frac{1}{2}x^T Ax - x^T b$ along the search directions $p_i = e_i$, $i = 1 : n$ is equivalent to one step of the Gauss–Seidel method.
3. How are the search directions chosen in the method of steepest descent? What is the asymptotic rate of convergence of this method?

10.4 Krylov Subspace Methods

The **Lanczos method** and the **conjugate gradient method** of Hestenes and Stiefel were both published in 1952. In the beginning these methods were viewed primarily as direct methods, since (in exact arithmetic) they terminate in at most n steps for a system of order n. The methods came into disrepute after it was shown that in finite precision they could take more than $3n$–$5n$ steps before they actually converged. The way rounding errors influenced the methods were not well understood. The methods did therefore not come into wide use until twenty years later. The key was to realize that they should be used as iterative methods for solving large sparse systems. Today these methods, combined with preconditioning techniques, are the standard solvers for large symmetric (or Hermitian) systems.

10.4.1 The Conjugate Gradient Method

The Lanczos method and the conjugate gradient methods started the era of Krylov subspace iterative methods. We recall the definition in Sec. 9.8.3: Given a matrix A and a vector b the corresponding nested sequence of Krylov subspaces are defined by

$$\mathcal{K}_k(b, A) = \text{span}\,\{b, Ab, \ldots, A^{k-1}b\}, \quad k = 1, 2, \ldots. \tag{10.4.1}$$

The conjugate gradient method for solving a symmetric, positive definite linear system $Ax = b$, where $A \in \mathbf{R}^{n\times n}$ is a projection method such that

$$x_k \in \mathcal{K}_k(b, A), \qquad r_k = b - Ax_k \perp \mathcal{K}_k(b, A). \tag{10.4.2}$$

Set $r_0 = p_0 = b$, and use the recurrence (10.3.19) to generate p_{k+1}. Then a simple induction argument shows that the vectors p_k and r_k both will lie in the $\mathcal{K}_k(b, A)$. In the conjugate gradient method the parameter β_k is chosen to make p_{k+1} A-orthogonal or conjugate to the previous search direction, i.e.,

$$p_{k+1}^T A p_k = 0. \tag{10.4.3}$$

(A motivation to this choice is given in the second remark to Theorem 10.3.4.) Multiplying (10.3.19) by $p_k^T A$ and using (10.4.3) it follows that

$$\beta_k = -(p_k^T A r_{k+1})/(p_k^T A p_k). \tag{10.4.4}$$

We now prove the important result that this choice will in fact make p_{k+1} A-conjugate to *all* previous search directions!

Lemma 10.4.1.

In the conjugate gradient algorithm the residual vector r_k is orthogonal to all previous search directions and residual vectors

$$r_k^T p_j = 0, \quad j = 0, \ldots, k-1, \tag{10.4.5}$$

and the search directions are mutually A-conjugate

$$p_k^T A p_j = 0, \quad j = 0 : k-1. \tag{10.4.6}$$

Proof. We first prove the relations (10.4.5) and (10.4.6) jointly by induction. Clearly r_k is orthogonal to the previous search direction p_{k-1}, and (10.4.3) shows that also (10.4.6) holds for $j = k-1$, Hence these relations are certainly true for $k = 1$.

Assume now that the statements are true for some $k \geq 1$. From $p_k^T r_{k+1} = 0$, changing the index, and taking the scalar product with p_j, $0 \leq j < k$ we get

$$r_{k+1}^T p_j = r_k^T p_j - \alpha_k p_k^T A p_j.$$

From the induction hypothesis this is zero, and since $r_{k+1}^T p_k = 0$ it follows that (10.4.5) holds for $k := k+1$. Using equation (10.3.19), the induction hypothesis and equation (10.3.10) and then (10.3.19) again we find for $0 < j < k$

$$\begin{aligned} p_{k+1}^T A p_j &= r_{k+1}^T A p_j + \beta_k p_k^T A p_j = \alpha_j^{-1} r_{k+1}^T (r_j - r_{j+1}) \\ &= \alpha_j^{-1} r_{k+1}^T (p_j - \beta_{j-1} p_{j-1} - p_{j+1} + \beta_j p_j), \end{aligned}$$

which is zero by equation (10.4.5). For $j = 0$ we use $b = p_0$ in forming the last line of the equation. For $j = k$ we use (10.4.3), which yields (10.4.6). □

Since the vectors $p_0, \dots, p_{k-1}$ span the Krylov subspace $\mathcal{K}_k(b, A)$ the equation (10.4.5) shows that $r_k \perp \mathcal{K}_k(b, A)$. This relation shows that the conjugate gradient implements the projection method obtained by taking $\mathcal{K} = \mathcal{L} = \mathcal{K}_k(b, A)$. Hence from Lemma 10.3.2 we have the following *global minimization property.*

Theorem 10.4.2.

The vector x_k in the conjugate gradient method solves the minimization problem

$$\min_x \phi(x) = \frac{1}{2}\|x - x^*\|_A^2, \quad x \in \mathcal{K}_k(b, A) \tag{10.4.7}$$

From this property it follows directly that the "energy" norm $\|x - x^*\|_A$ in the CG method is monotonically decreasing. It can also be shown that the error norm $\|x - x_k\|_2$ is monotonically decreased (see Hestenes and Stiefel [206]).

Since the vectors $r_0, \dots, r_{k-1}$ span the Krylov subspace $\mathcal{K}_k(b, A)$ the following orthogonality relations also hold:

$$r_k^T r_j = 0, \quad j = 0 : k - 1. \tag{10.4.8}$$

Equation (10.4.8) ensures that in exact arithmetic the conjugate gradient method will terminate after at most n steps. For suppose the contrary is true. Then $r_k \neq 0$, $k = 0 : n$ and by (10.4.8) these $n+1$ nonzero vectors in $\mathbf{R}^n$ are mutually orthogonal and hence linearly independent, which is impossible. Hence the conjugate gradient method is in effect a direct method! However, as is now well know, round-off errors spoil the finite termination property and this aspect has little practical relevance.

From these relations, we can conclude that *the residuals vectors $r_0, r_1, \dots, r_k$ are the vectors that would be obtained from the sequence $b, Ab, \dots, A^k b$ by Gram–Schmidt orthogonalization.* This gives a connection to the Lanczos process described in Sec. 9.8.4, which will be further discussed below. The vectors $p_0, p_1, p_2, \dots,$ may be constructed similarly from the conjugacy relation (10.4.6).

An alternative expression for β_k is obtained by multiplying the recursive expression for the residual $r_{k+1} = r_k - \alpha_k A p_k$ by r_{k+1}^T and using the orthogonality (10.4.8) to get $r_{k+1}^T r_{k+1} = -\alpha_k r_{k+1}^T A p_k$. Equations (10.3.21) and (10.4.4) then yield

$$\beta_k = \|r_{k+1}\|_2^2 / \|r_k\|_2^2.$$

We observe that in this expression for β_k the matrix A is not needed. This property is important when the conjugate gradient method is extended to non-quadratic functionals.

We now summarize the conjugate gradient method. We have seen that there are alternative, mathematically equivalent formulas for computing r_k, α_k and β_k. However, these are not equivalent with respect to accuracy, storage and computational work. A comparison tends to favor the following version:

When a starting approximation $x_0 \neq 0$ is known then we can set $x = x_0 + z$, and apply the algorithm to the shifted system

$$Az = r_0, \quad r_0 = b - Ax_0.$$

This leads to the following algorithm:

Algorithm 10.1.
The Conjugate Gradient Method

$$
\begin{aligned}
& p_0 = r_0 = b - Ax_0; \\
& \textbf{for } k = 0, 1, 2, \ldots \textbf{ while } \|r_k\|_2 > tol \\
& \quad q_k = Ap_k; \\
& \quad \alpha_k = (r_k, r_k)/(p_k, q_k); \\
& \quad x_{k+1} = x_k + \alpha_k p_k; \\
& \quad r_{k+1} = r_k - \alpha_k q_k; \\
& \quad \beta_k = (r_{k+1}, r_{k+1})/(r_k, r_k); \\
& \quad p_{k+1} = r_{k+1} + \beta_k p_k; \\
& \textbf{end}
\end{aligned}
$$

Note that the matrix A only occurs in the matrix-vector operation Ap_k. Hence, the matrix A need not be explicitly available, and can be represented by a subroutine. Here the inner product used is $(p, q) = p^T q$. Four vectors x, r, p and Ap need to be stored. Each iteration step requires one matrix by vector product, two vector inner products, and three scalar by vector products. We remark that the computation of the inner products can be relative expensive since they cannot be parallelized.

By instead taking the inner product in the above algorithm to be $(p, q) = p^T Aq$ we obtain a related method that in each step minimizes the Euclidian norm of the residual over the same Krylov subspace. In this algorithm the vectors Ap_i, $i = 0, 1, \ldots$ are orthogonal. In addition, the residual vectors are required to be A-orthogonal, i.e., conjugate. Consequently this method is called the **conjugate residual method**. This algorithm requires one more vector of storage and one more vector update than the conjugate gradient method. Therefore, when applicable the conjugate gradient method is usually preferred over the conjugate residual method.

10.4.2 The Lanczos Connection

We will now exhibit the close connection between the conjugate gradient method and the Lanczos process described in Sec. 9.8.4. Recall that starting with a vector

v_1, the Lanczos process generates for $k = 1, 2, 3, \ldots$ a symmetric tridiagonal matrix

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & \beta_k & \alpha_k \end{pmatrix}.$$

and a matrix $V_k = (v_1, \ldots, v_k)$ with orthogonal columns spanning the Krylov subspace $\mathcal{K}_k(v_1, A)$, such that

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T. \tag{10.4.9}$$

In the context of solving the linear system $Ax = b$, the appropriate choice of starting vector is given by

$$\beta_1 v_1 = b, \quad \beta_1 = \|b\|_2. \tag{10.4.10}$$

In the kth step of the CG method we determine an approximation $x_k \in \mathcal{K}_k(b, A)$ such that $r_k = b - Ax_k$ is orthogonal to $\mathcal{K}_k(b, A)$. Since V_k is an orthogonal basis in $\mathcal{K}_k(b, A)$, we can set $x_k = V_k y_k$. Then using (10.4.9) we have

$$r_k = b - AV_k y_k = \beta_1 v_1 - V_k T_k y_k - \beta_{k+1}(e_k^T y_k) v_{k+1}. \tag{10.4.11}$$

Since $V_k^T r_k = 0$ multiplying (10.4.11) by V_k^T and using $V_k^T v_{k+1} = 0$ and $V_k^T v_1 = e_1$, gives

$$0 = V_k^T r_k = \beta_1 e_1 - T_k y_k,$$

that is y_k satisfies the tridiagonal system

$$T_k y_k = \beta_1 e_1. \tag{10.4.12}$$

In exact arithmetic $x_k = V_k y_k$, $k = 1, 2, \ldots$, is the same sequence of approximations as generated by the CG method. Further, the columns of V_k equal the first k residual vectors in the conjugate gradient method, normalized to unit length.

The Lanczos process stops if $\beta_{k+1} = \|r_k\|_2 = 0$ since then v_{k+1} is not defined. However, then we have $AV_k = V_k T_k$ and using (10.4.11)

$$0 = r_k = \beta_1 v_1 - V_k T_k y_k = b - AV_k y_k = b - Ax_k.$$

It follows that $Ax_k = b$, i.e. x_k is an exact solution.

There is one drawback with using the Lanczos process as outlined above. To compute $x_k = V_k y_k$ seems to require that all the vectors $y_1, \ldots, y_k$ are saved, wheras in the CG method a two-term recurrence relation is used to update x_k.

The recursion in the conjugate gradient method is obtained from (10.4.12) by computing the Cholesky factorization of $T_k = R_k^T R_k$. Suppose the Lanczos process stops for $k = l \leq n$. Then, since A is a positive definite matrix the matrix $T_l = V_l^T A V_l$ is also positive definite. Thus T_k, $k \leq l$, which is a principal submatrix of T_l, is also positive definite and its Cholesky factorization must exist.

So far we have discussed the Lanczos process in exact arithmetic. In practice, roundoff will cause the generated vectors to lose orthogonality. A possible remedy is to reorthogonalize each generated vector v_{k+1} to all previous vectors $v_k, \dots, v_1$. This is however very costly both in terms of storage and operations. The effect of finite precision on the Lanczos method is the same as for the CG method; it slows down convergence, but fortunately does not prevent accurate approximations to be found!

Assuming that $r_j \neq 0, = 1 : k$ we introduce for the conjugate gradient method the following matrices:

$$R_k = (\, r_1 \quad r_2 \quad \dots \quad r_k \,)\, S_k^{-1}, \qquad P_k = (\, p_1 \quad p_2 \quad \dots \quad p_k \,)\, S_k^{-1}, \tag{10.4.13}$$

$$L_k = \begin{pmatrix} 1 & & & & \\ -\sqrt{\beta_1} & 1 & & & \\ & -\sqrt{\beta_2} & 1 & & \\ & & \ddots & \ddots & \\ & & & -\sqrt{\beta_{k-1}} & 1 \end{pmatrix} \tag{10.4.14}$$

and the diagonal matrices

$$S_k = \mathrm{diag}\,(\, \|r_1\|_2 \quad \|r_2\|_2 \quad \dots \quad \|r_k\|_2 \,)\,, \tag{10.4.15}$$

$$D_k = \mathrm{diag}\,(\, \alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_k \,)\,. \tag{10.4.16}$$

Since the residual vectors r_j, $j = 1 : k$, are mutually orthogonal, R_n is an orthogonal matrix. Further, we have the relations

$$AP_nD_n = R_nL_n, \qquad P_nL_n = R_n.$$

Eliminating P_n from the first relation we obtain

$$AR_n = R_n(L_nD_n^{-1}L_n) = R_nT_n,$$

Hence this provides an orthogonal similarity transformation of A to symmetric tridiagonal form T_n.

10.4.3 Convergence of the CG Method

In a Krylov subspace method the approximations are of the form $x_k - x_0 \in \mathcal{K}_k(b, A)$, $k = 1, 2, \dots$. With $r_k = b - Ax_k$ it follows that $r_k - b \in A\mathcal{K}_k(b, A)$. Hence the residual vectors can be written

$$r_k = q_k(A)b,$$

where $q_k \in \tilde{\Pi}_k^1$, the set of polynomials q_k of degree k with $q_k(0) = 1$. Since

$$\phi(x) = \tfrac{1}{2}\|x - x^*\|_A^2 = \tfrac{1}{2}r^TA^{-1}r = \tfrac{1}{2}\|r\|_{A^{-1}}^2,$$

the optimality property in Theorem 10.4.2 can alternatively be stated as

$$\|r_k\|_{A^{-1}}^2 = \min_{q_k \in \tilde{\Pi}_k^1} \|q_k(A)b\|_{A^{-1}}^2. \tag{10.4.17}$$

Denote by $\{\lambda_i, v_i\}$, $i = 1, \dots, n$, the eigenvalues and eigenvectors of A. Since A is symmetric we can assume that the eigenvectors are orthonormal. Expanding the right hand side as

$$b = \sum_{i=1}^{n} \gamma_i v_i, \tag{10.4.18}$$

we have for *any* $q_k \in \tilde{\Pi}_k^1$

$$\|r_k\|_{A^{-1}}^2 \le \|q_k(A)b\|_{A^{-1}}^2 = b^T q_k(A)^T A^{-1} q_k(A) b = \sum_{i=1}^{n} \gamma_i^2 \lambda_i^{-1} q_k(\lambda_i)^2.$$

In particular, taking

$$q_n(\lambda) = \Big(1 - \frac{\lambda}{\lambda_1}\Big)\Big(1 - \frac{\lambda}{\lambda_2}\Big) \cdots \Big(1 - \frac{\lambda}{\lambda_n}\Big), \tag{10.4.19}$$

we get $\|r_n\|_{A^{-1}} = 0$. This is an alternative proof that the CG method terminates after at most n steps in exact arithmetic.

If the eigenvalues of A are distinct then q_n in (10.4.19) is the minimal polynomial of A (see Section 10.1.2). If A only has p distinct eigenvalues then the minimal polynomial is of degree p and CG converges in at most p steps for any vector b. Hence, CG is particularly effective when A has low rank! More generally, if the grade of b with respect to A equals m then only m steps are needed to obtain the exact solution. This will be the case if, e.g., in the expansion (10.4.18) $\gamma_i \neq 0$ only for m different values of i.

We stress that the finite termination property of the CG method shown above is only valid in exact arithmetic. In practical applications we want to obtain a good approximate solution x_k in far less than n iterations. We now use the optimality property (10.4.18) to derive an upper bound for the rate of convergence of the CG method considered as an iterative method. Let the set S contain all the eigenvalues of A and assume that for some $\tilde{q}_k \in \tilde{\Pi}_k^1$ we have

$$\max_{\lambda \in S} |\tilde{q}_k(\lambda)| \le M_k.$$

Then it follows that

$$\|r_k\|_{A^{-1}}^2 \le M_k^2 \sum_{i=1}^{n} \gamma_i^2 \lambda_i^{-1} = M_k^2 \|b\|_{A^{-1}}^2$$

or

$$\|x - x_k\|_A \le M_k \|x - x_0\|_A. \tag{10.4.20}$$

We now select a set S on the basis of some assumption regarding the eigenvalue distribution of A and seek a polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ such that $M_k = \max_{\lambda \in S} |\tilde{q}_k(\lambda)|$ is small.

A simple choice is to take $S = [\lambda_1, \lambda_n]$ and seek the polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ which minimizes

$$\max_{\lambda_1 \le \lambda \le \lambda_n} |q_k(\lambda)|.$$

The solution to this problem is known to be a shifted and scaled Chebyshev polynomial of degree k, see the analysis for Chebyshev semi-iteration in Sec. 10.2.4. It follows that

$$\|x - x_k\|_A < 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x - x_0\|_A. \tag{10.4.21}$$

where $\kappa = \lambda_n(A)/\lambda_1(A)$. Note that the convergence of the conjugate residual method can be analyzed using a similar technique.

Example 10.4.1.

For the model problem in Section 10.1.3 the extreme eigenvalues of $\frac{1}{4}A$ are $\lambda_{max} = 1 + \cos \pi h$, $\lambda_{min} = 1 - \cos \pi h$. It follows that

$$\kappa = \frac{1 + \cos \pi h}{1 - \cos \pi h} \approx \frac{1}{\sin^2 \pi h/2} \approx \frac{4}{(\pi h)^2}.$$

For $h = 1/100$ the number of iterations needed to reduce the initial error by a factor of 10^{-3} is then bounded by

$$k \approx \frac{1}{2} \log 2 \cdot 10^3 \sqrt{\kappa} \approx 242.$$

This is about the same number of iterations as needed with SOR using ω_{opt} to reduce the L_2-norm by the same factor. However, the conjugate gradient method is more general in that it does not require the matrix A to have “property A”.

The error estimate above tends to be pessimistic asymptotically. When there are gaps in the spectrum of A then, as the iterations proceeds, the effect of the smallest and largest eigenvalues of A are eliminated and the convergence then behaves according to a smaller ”effective” condition number. This behavior, called **superlinear convergence**, is in contrast to the Chebyshev semi-iterative method, which only takes the extreme eigenvalues of the spectrum into account and for which the error estimate in Section 10.2.4 tends to be sharp asymptotically.

We have seen that, in exact arithmetic, the conjugate gradient algorithm will produce the exact solution to a linear system $Ax = b$ in at most n steps. In the presence of rounding errors, the orthogonality relations in Theorem 10.3.4 will no longer be satisfied exactly. Indeed, orthogonality between residuals r_i and r_j, for $|i-j|$ is large, will usually be completely lost. Because of this, the finite termination property does not hold in practice.

The behavior of the conjugate gradient algorithm in finite precision is much more complex than in exact arithmetic. It has been observed that the bound (10.4.21) still holds to good approximation in finite precision. On the other hand a good approximate solution may not be obtained after n iterations, even though a large drop in the error sometimes occurs after step n. It has been observed that the conjugate gradient algorithm in finite precision behaves like the exact algorithm applied to a larger linear system $\hat{A}\hat{x} = \hat{b}$, where the matrix $\hat{A}$ has many eigenvalues distributed in tiny intervals about the eigenvalues of A. This means that $\kappa(\hat{A}) \approx \kappa(A)$, which explains why the bound (10.4.21) still applies. It can also be shown that even in finite precision $\|r_k\|_2 \to 0$, where r_k is the recursively computed residual in the algorithm. (Note that the norm of true residual $\|b - Ax_k\|_2$ cannot be expected to approach zero.) This means that a termination criterion $\|r_k\|_2 \leq \epsilon$ will eventually always be satisfied even if $\epsilon \approx u$, where u is the machine precision.

Example 10.4.2.

Consider Laplace equation

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$$

in the unit square $0 < x, y < 1$, with Dirichlet boundary conditions determined so that the solution is

$$u(x, y) = 2\Big((x - 1/2)^2 + (y - 1/2)^2\Big).$$

The Laplacian operator is approximated with the usual five-point operator with 32 mesh points in each direction; see Sec. sec10.1.1. This leads to a linear system of dimension $31^2 = 961$. The initial approximation is taken to be identically zero.

Table 10.4.1. *Maximum error for Example 10.4.2 using Chebyshev iteration with optimal parameters and the conjugate gradient algorithm.*

Iteration	Chebyshev	Conjugate gradient
1	$1.6 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$
2	$7.1 \cdot 10^{-4}$	$6.5 \cdot 10^{-4}$
3	$1.1 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$
4	$2.7 \cdot 10^{-7}$	$1.0 \cdot 10^{-7}$
5	$4.3 \cdot 10^{-9}$	$8.1 \cdot 10^{-10}$
6	$1.2 \cdot 10^{-10}$	$5.7 \cdot 10^{-12}$

In Table 10.4.2 we compare the maximum error using Chebyshev iteration with optimal parameters and the conjugate gradient algorithm. An initial estimate identically equal to zero is used. It is seen that the cg-algorithm yields a smaller error and there is no need to estimate the parameters a priori.

10.4.4 Symmetric Indefinite Systems

For symmetric positive definite matrices A the conjugate gradient method computes iterates x_k that satisfy the minimization property

$$\min_{x \in S_k} \|\hat{x} - x\|_A, \qquad S_k = x_0 + \mathcal{K}_k(b, A).$$

In case A is symmetric but indefinite $\|\cdot\|_A$ is no longer a norm. Hence the standard conjugate gradient method may break down. This is also true for the conjugate residual method.

A Krylov subspace method for symmetric indefinite systems was given by Paige and Saunders [300, 1975]. Using the Lanczos basis V_k they seek approximations $x_k = V_k y_k \in \mathcal{K}_k(b, A)$, which are stationary values of $\|\hat{x} - x_k\|_A^2$. These are given by the Galerkin condition

$$V_k^T(b - AV_k y_k) = 0.$$

This leads again to the tridiagonal system (10.4.12). However, when A is indefinite, although the Lanczos process is still well defined, the Cholesky factorization of T_k may not exist. Moreover, it may happen that T_k is singular at certain steps, and then y_k is not defined.

If the Lanczos process stops for some $k \le n$ then $AV_k = V_k T_k$. It follows that the eigenvalues of T_k are a subset of the eigenvalues of A, and thus if A is nonsingular so is T_k. Hence the problem with a singular T_k can only occur at intermediate steps.

To solve the tridiagonal system (10.4.12) Paige and Saunders suggest computing the LQ factorization

$$T_k = \bar{L}_k Q_k, \qquad Q_k^T Q_k = I,$$

where $\bar{L}_k$ is lower triangular and Q_k orthogonal. Such a factorization always exists and can be computed by multiplying T_k with a sequence of plane rotations from the right

$$T_k G_{12} \cdots G_{k-1,k} = \bar{L}_k = \begin{pmatrix} \gamma_1 & & & & \\ \delta_2 & \gamma_2 & & & \\ \epsilon_3 & \delta_3 & \gamma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \epsilon_k & \delta_k & \bar{\gamma}_k \end{pmatrix}.$$

The rotation $G_{k-1,k}$ is defined by elements c_{k-1} and s_{k-1}. The bar on the element $\bar{\gamma}_k$ is used to indicate that $\bar{L}_k$ differs from L_k, the $k \times k$ leading part of $\bar{L}_{k+1}$, in the (k, k) element only. In the next step the elements in $G_{k,k+1}$ are given by

$$\gamma_k = (\bar{\gamma}_k^2 + \beta_{k+1}^2)^{1/2}, \qquad c_k = \bar{\gamma}_k/\gamma_k, \qquad s_k = \beta_{k+1}/\gamma_k.$$

Since the solution y_k of $T_k y_k = \beta_1 e_1$ will change fully with each increase in k we write

$$x_k = V_k y_k = (V_k Q_k^T)\bar{z}_k = \bar{W}_k \bar{z}_k,$$

and let

$$\begin{aligned} \bar{W}_k &= (w_1, \dots, w_{k-1}, \bar{w}_k), \\ \bar{z}_k &= (\zeta_1, \dots, \zeta_{k-1}, \bar{\zeta}_k) = Q_k y_k. \end{aligned}$$

Here quantities without bars will be unchanged when k increases, and $\bar{W}_k$ can be updated with $\bar{T}_k$. The system (10.4.12) now becomes

$$\bar{L}_k \bar{z}_k = \beta_1 e_1, \qquad x_k^c = \bar{W}_k \bar{z}_k.$$

This formulation allows the v_i and w_i to be formed and discarded one by one.

In implementing the algorithm we should note that x_k^c need not be updated at each step, and that if $\bar{\gamma}_k = 0$, then $\bar{z}_k$ is not defined. Instead we update

$$x_k^L = W_k z_k = x_{k-1}^L + \zeta_k w_k,$$

where L_k is used rather than $\bar{L}_k$. We can then always obtain x_{k+1}^c when needed from

$$x_{k+1}^c = x_k^L + \bar{\zeta}_{k+1}\bar{w}_{k+1}.$$

This defines the SYMMLQ algorithm. In theory the algorithm will stop with $\beta_{k+1} = 0$ and then $x_k^c = x_k^L = x$. In practice it has been observed that β_{k+1} will rarely be small and some other stopping criterion based on the size of the residual must be used.

Paige and Saunders also derived an algorithm called MINRES, which is based on minimizing the Euclidian norm of the residual r_k. It should be noted that MINRES suffers more from poorly conditioned systems than SYMMLQ does.

10.4.5 Estimating Matrix Functionals

Let f be smooth function on a given real interval $[a, b]$ and consider the **matrix functional**

$$F(A) = u^T f(A)u, \tag{10.4.22}$$

where u is a given vector and $A \in \mathbf{R}^{n\times n}$ is a symmetric matrix. The evaluation of such a functional arises in many applications.

Example 10.4.3.

Let $\bar{x}$ be an approximate solution of a linear system $Ax = b$. If $r = b - A\bar{x}$ is the residual vector then the error $e = x - \bar{x}$ can be expressed as $e = A^{-1}x$. Thus the 2-norm of the error equals

$$\|e\|_2 = e^T e = r^T A^{-2} r = r^T f(A) r, \quad f(x) = x^{-2}. \tag{10.4.23}$$

Therefore the problem of computing an upper bound for $\|e\|_2$ given the residual vector is a special case of estimating a matrix functional.

Since A is a real symmetric matrix it has a spectral decomposition

$$A = Q\Lambda Q^T,$$

where Q is an orthonormal matrix whose columns are the normalized eigenvalues of A and Λ is a diagonal matrix containing the eigenvalues

$$a = \lambda_1 \le \lambda_2 \le \cdots \lambda_n = b.$$

Then by definition of a matrix function we have

$$F(A) = u^T Q f(\Lambda) Q^T u = \sum_{i=1}^{n} f(\lambda_i)\mu_j^2, \quad Q^T u = (\mu_1, \mu_2, \ldots, \mu_n)^T \tag{10.4.24}$$

In the following we assume that $\|u\|_2 = \|Q^T u\|_2 = 1$.

The last sum can be considered as a Riemann–Stieltjes integral

$$F(A) = \int_a^b f(t)\, d\mu(t),$$

where the measure μ is piecewise constant and defined by

$$\mu(t) = \begin{cases} 0, & \text{if } t < a; \\ \sum_{j=1}^{i} \mu_j^2, & \text{if } \lambda_i \le t < \lambda_{i+1}; \\ \sum_{j=1}^{n} \mu_j^2, & \text{if } \lambda_n \le t. \end{cases}$$

Review Questions

1. Define the Krylov space $\mathcal{K}_j(b, A)$. Show that it is invariant under (i) scaling τA. (ii) translation $A - sI$. How is it affected by an orthogonal similarity transformation $\Lambda = V^T A V$, $c = V^T b$?

2. What minimization problems are solved by the conjugate gradient method? How can this property be used to derive an upper bound for the rate of convergence of the conjugate gradient method.

3. Let the symmetric matrix A have eigenvalues λ_i and orthonormal eigenvectors v_i, $i = 1, \ldots, n$. If only $d < n$ eigenvalues are distinct, what is the maximum dimension of the Krylov space $\mathcal{K}_j(b, A)$?

Problems and Computer Exercises

1. Let λ_i, v_i be an eigenvalue and eigenvector of the symmetric matrix A.

(a) Show that if $v_i \perp b$, then also $v_i \perp \mathcal{K}_j(b, A)$, for all $j > 1$.

(b) Show that if b is orthogonal against p eigenvectors, then the maximum dimension of $\mathcal{K}_j(b, A)$ is at most $n - p$. Deduce that the the conjugate gradient method converges in at most $n - p$ iterations.

2. Let $A = I + BB^T \in \mathbf{R}^{n \times n}$, where B is of rank p. In exact arithmetic, how many iterations are at most needed to solve a system $Ax = b$ with the conjugate gradient method?

3. Write down explicitly the conjugate residual method. Show that in this algorithm one needs to store the vectors x, r, Ar, p and Ap.

4. SYMMLQ is based on solving the tridiagonal system (10.4.9) using an LQ factorization of T_k. Derive an alternative algorithm, which solves this system with Gaussian elimination with partial pivoting.

10.5 Iterative Least Squares Methods.

10.5.1 Introduction

In this section we consider the iterative solution of a nonsymmetric linear system, or more generally, a large sparse least squares problems

$$\min_x \|Ax - b\|_2, \quad A \in \mathbf{R}^{m\times n}. \tag{10.5.1}$$

We assume in the following, unless otherwise stated, that A has full column rank, so that this problem has a unique solution.

One way to proceed would be to form the positive definite system of normal equations

$$A^TAx = A^Tb.$$

Then any iterative method for symmetric positive definite linear systems can be used.

The symmetrization of A by premultiplication by A^T is costly. It is equivalent to n iterations for the original system. However, it is easy to see that the explicit formation of the matrix A^TA can usually be avoided by using the **factored form** of the normal equations

$$A^T(Ax - b) = 0 \tag{10.5.2}$$

of the normal equations. Working only with A and A^T separately has two important advantages. First, as has been much emphasized for direct methods, a small perturbation in A^TA, e.g., by roundoff, may change the solution much more than perturbations of similar size in A itself. Second, we avoid the fill which can occur in the formation of A^TA. The matrix A^TA often has many more nonzero elements than A and can be quite dense even when A is sparse.

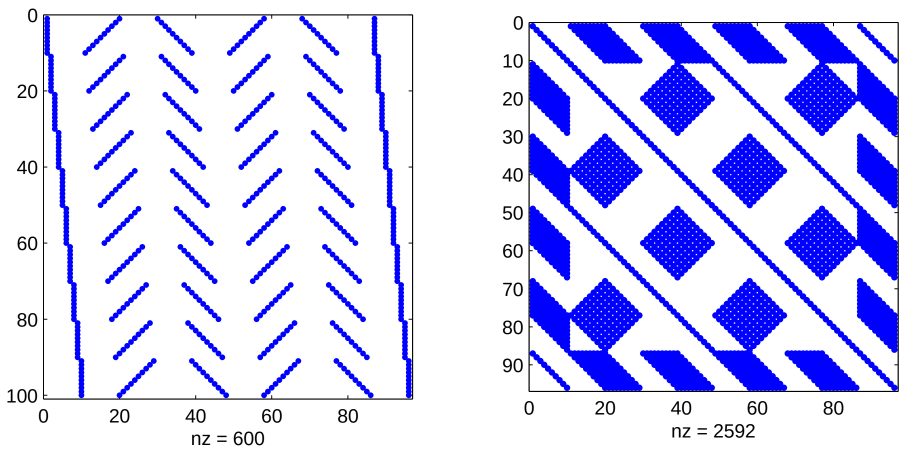

Figure 10.5.1. *Structure of A (left) and A^TA (right) for a model problem.*

Iterative methods can be used also for computing a minimum norm solution of a (consistent) underdetermined system,

$$\min \|y\|_2, \qquad A^T y = c.$$

If A^T has full row rank the unique solution satisfies the normal equations of the second kind

$$y = Az, \qquad A^T(Az) = c. \tag{10.5.3}$$

Again the explicit formation of the cross-product matrix A^TA should be avoided. Another possible approach is to use an iterative method applied to the augmented system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}; \tag{10.5.4}$$

see Theorem 1.1.5. This also avoids forming the normal equations, but has the drawback that since the augmented system is symmetric indefinite many standard iterative methods cannot be applied.

If A is square and nonsingular, then the normal equations are symmetric and positive definite with solution $x = A^{-1}b$. Hence a natural extension of iterative methods for symmetric positive definite systems to general nonsingular, non-symmetric linear systems $Ax = b$ is to to apply them to the normal equations of first or second kind. A severe drawback can be that $\kappa(A^TA) = \kappa(AA^T) = \ kappa^2(A)$, i.e. *the condition number is squared* compared to the original system $Ax = b$. From the estimate (10.4.21) we note that this can lead to a substantial decrease in the rate of convergence.

For some classes of sparse least squares problems fill-in will make sparse direct methods prohibitively costly in terms of storage and operations. There are problems where A is large and sparse but A^TA is almost dense; compare Figure 10.5.1. Then the Cholesky factor R will in general also be dense. This rules out the use of sparse direct methods based on QR decomposition. For an example, consider the case when A has a *random sparsity structure*, such that an element a_{ij} is nonzero with probability $p < 1$. Ignoring numerical cancellation it follows that $(A^TA)_{jk} \neq 0$ with probability

$$q = 1 - (1 - p^2)^m \approx 1 - e^{-mp^2}.$$

Therefore A^TA will be almost dense when $mp \approx m^{1/2}$, i.e., when the average number of nonzero elements in a column equals about $m^{1/2}$. This type of structure is common in reconstruction problems. An example is the inversion problem for the velocity structure for the Central California Microearthquake Network, for which (in 1980) $m = 500{,}000$, $n = 20{,}000$, and A has about 10^7 nonzero elements with a very irregular structure. The matrix A^TA will be almost dense.

10.5.2 Landweber's and Cimmino's methods

The non-stationary Richardson iteration applied to the normal equations $A^TAx = A^Tb$ can be written in the form

$$x^{(k+1)} = x^{(k)} + \omega_k A^T(b - Ax^{(k)}), \quad k = 1, 2, \dots,$$

This method is often referred to as **Landweber's method** [252]. It can be shown that this methods is convergent provided that for some $\epsilon > 0$ it holds

$$0 < \epsilon < \omega_k < (2 - \epsilon)/\sigma_{\max}(A), \quad \forall k.$$

An important things to notice in the implementation is that to avoid numerical instability and fill-in, the matrix $A^T A$ should not be *explicitly* computed.

The eigenvalues of the iteration matrix $G = I - \omega A^T A$ equal

$$\lambda_k(G) = 1 - \omega \sigma_k^2, \quad k = 1 : n,$$

where σ_k are the singular values of A. From this it can be shown that Richardson's method converges to the least squares solution $x = A^\dagger b$ if

$$x^{(0)} \in \mathcal{R}(A^T), \qquad 0 < \omega < 2/\sigma_1^2(A).$$

Cimmino ([69]) introduced a method that is related the Landweber's method. Consider a square nonsingular system $Ax = b$, where the rows of A are $a_1^T, \ldots, a_n^T$. The solution $x = A^{-1}b$ is then equal to the intersection of the n hyper-planes

$$a_i^T x = b_i, \quad i = 1 : n.$$

In **Cimmino's method** one considers the reflections of an initial approximation $x^{(0)}$, with respect to these hyper-planes

$$x_i^{(0)} = x^{(0)} + 2\frac{(b_i - a_i^T x^{(0)})}{\|a_i\|_2} a_i, \quad i = 1 : n. \tag{10.5.5}$$

The next approximation is then taken to be

$$x^{(1)} = \frac{1}{\mu} \sum_{i=1}^n m_i x_i^{(0)}, \quad \mu = \sum_{i=1}^n m_i.$$

This can be interpreted as the center of gravity of n masses m_i placed at the points $x_i^{(0)}$. Cimmino noted that the initial point $x^{(0)}$ and its reflections with respect to the n hyperplanes (10.5.5) all lie on a hyper-sphere the center of which is the solution of the linear system. Because the center of gravity of the system of masses m_i must fall inside this hyper-sphere it follows that

$$\|x^{(1)} - x\|_2 < \|x^{(0)} - x\|_2,$$

that is, the error is reduced. Therefore Cimmino's method converges.

In matrix form Cimmino's method can be written as

$$x^{(k+1)} = x^{(k)} + \frac{2}{\mu} A^T D(b - Ax^{(k)}), \tag{10.5.6}$$

where $D = \operatorname{diag}(d_1, \ldots, d_n)$, where $d_i = m_i/\|a_i\|_2^2$. In particular, with $m_i = \|a_i\|_2$ we get Landweber's method with $\omega = 2/\mu$. It follows that Cimmino's method also converges for singular and inconsistent linear systems.

10.5.3 Jacobi's and Gauss–Seidel's Methods

Assume that all columns in $A = (a_1, \dots, a_n) \in \mathbf{R}^{m\times n}$ are nonzero, and set

$$d_j = a_j^T a_j = \|a_j\|_2^2 > 0. \tag{10.5.7}$$

In the jth minor step of **Jacobi's** method equation for solving the normal equations $A^T(Ax-b) = 0$ we compute $x_j^{(k+1)}$ so that the jth equation $a_j^T(Ax-b) = 0$ is satisfied, that is

$$x_j^{(k+1)} = x_j^{(k)} + a_j^T(b - Ax^{(k)})/d_j, \quad j = 1:n. \tag{10.5.8}$$

Thus Jacobi's method can be written in matrix form as

$$x^{(k+1)} = x^{(k)} + D_A^{-1}A^T(b - Ax^{(k)}), \tag{10.5.9}$$

where

$$D_A = \mathrm{diag}\,(d_1, \dots, d_n) = \mathrm{diag}\,(A^TA).$$

Note that Jacobi's method is symmetrizable, since

$$D_A^{1/2}(I - D_A^{-1}A^TA)D_A^{-1/2} = I - D_A^{-1/2}A^TAD_A^{-1/2}.$$

The Gauss–Seidel method is a special case of the following class of **residual reducing** methods. Let $p_j \notin \mathcal{N}(A)$, $j = 1, 2, \dots$, be a sequence of nonzero n-vectors and compute a sequence of approximations of the form

$$x^{(j+1)} = x^{(j)} + \alpha_j p_j, \qquad \alpha_j = p_j^T A^T(b - Ax^{(j)})/\|Ap_j\|_2^2. \tag{10.5.10}$$

It is easily verified that $r^{(j+1)} \perp Ap_j = 0$, where $r_j = b - Ax^{(j)}$, and by Phythagoras' theorem

$$\|r^{(j+1)}\|_2^2 = \|r^{(j)}\|_2^2 - |\alpha_j|^2\|Ap_j\|_2^2 \le \|r^{(j)}\|_2^2.$$

This shows that this class of methods (10.5.10) is residual reducing. For a square matrix A method (10.5.11) was developed by de la Garza [91, 1951]. This class of residual reducing projection methods was studied by Householder and Bauer [219, 1960].

If A has linearly independent columns we obtain the Gauss–Seidel method for the normal equations by taking p_j in (10.5.10) equal to the unit vectors e_j in cyclic order. Then if $A = (a_1, a_2, \dots, a_n)$, we have $Ap_j = Ae_j = a_j$. An iteration step in the Gauss–Seidel method consists of n minor steps where we put $z^{(1)} = x^{(k)}$, and $x^{(k+1)} = z^{(n+1)}$ is computed by

$$z^{(j+1)} = z^{(j)} + e_j a_j^T r^{(j)}/d_j, \qquad r^{(j)} = b - Az^{(j)}, \tag{10.5.11}$$

$j = 1:n$. In the jth minor step only the jth component of $z^{(j)}$ is changed, and hence the residual $r^{(j)}$ can be cheaply updated. With $r^{(1)} = b - Ax^{(k)}$ we obtain the simple recurrence

$$\begin{aligned} \delta_j = a_j^T r^{(j)}/d_j, \qquad & z^{(j+1)} = z^{(j)} + \delta_j e_j, \\ & r^{(j+1)} = r^{(j)} - \delta_j a_j, \quad j = 1:n. \end{aligned} \tag{10.5.12}$$

Note that in the jth minor step only the jth column of A is accessed. and that it can be implemented without forming the matrix A^TA explicitly. In contrast to the Jacobi method the Gauss–Seidel method is not symmetrizable and the ordering of the columns of A will influence the convergence.

The Jacobi method has the advantage over Gauss–Seidel's method that it is more easily adapted to parallel computation, since it just requires a matrix-vector multiplication. Further, it does not require A to be stored (or generated) columnwise, since products of the form Ax and A^Tr can conveniently be computed also if A can only be accessed by rows. In this case, if $a_1^T, \dots, a_m^T$ are the rows of A, then we have

$$(Ax)_i = a_i^T x, \quad i = 1, \dots, n, \quad A^T r = \sum_{i=1}^m r_i a_i,$$

where r_i is the ith component of the residual r.

The **successive over-relaxation (SOR) method** for the normal equations $A^TAx = A^Tb$ is obtained by introducing an **relaxation parameter** ω in the Gauss–Seidel method (10.5.13),

$$\begin{aligned} \delta_j = \omega a_j^T r^{(j)}/d_j, \qquad & z^{(j+1)} = z^{(j)} + \delta_j e_j, \\ & r^{(j+1)} = r^{(j)} - \delta_j a_j, \quad j = 1 : n. \end{aligned} \tag{10.5.13}$$

The SOR method always converges when A^TA is positive definite and ω satisfies $0 < \omega < 2$. The SOR method shares with the Gauss–Seidel method the advantage of simplicity and small storage requirements. However, when A^TA does not have Young's property A, the rate of convergence may be slow for any choice of ω. Then the SSOR method is to be preferred. This is easily implemented by following each forward sweep (10.5.13) with a backward sweep $j = n : -1 : 1$.

We now consider normal equations of second type $y = Az$, where $A^T(Az) = c$. In Jacobi's method $z_j^{(k)}$ is modified in the jth minor step so that the jth equation $a_j^T Az = c_j$ is satisfied, that is

$$z_j^{(k+1)} = z_j^{(k)} + (c_j - a_j^T(Az^{(k)}))/d_j, \quad j = 1 : n, \tag{10.5.14}$$

where as before $d_j = \|a_j\|_2^2$. Multiplying by A and setting $y^{(k)} = Az^{(k)}$, the iteration becomes in matrix form

$$y^{(k+1)} = y^{(k)} + AD_A^{-1}(c - A^T y^{(k)}), \quad D_A = \operatorname{diag}(A^TA). \tag{10.5.15}$$

The Gauss–Seidel method for solving the normal equations of second kind can also be implemented without forming A^TA. It is a special case of a family of **error reducing** methods defined as follows: Let $p_i \notin \mathcal{N}(A)$, $i = 1, 2, \dots$, be a sequence of nonzero n-vectors and compute approximations of the form

$$y^{(j+1)} = y^{(j)} + \omega_j A p_j, \qquad \omega_j = p_j^T(c - A^T y^{(j)})/\|Ap_j\|_2^2. \tag{10.5.16}$$

If the system $A^T y = c$ is consistent there is a unique solution y of minimum norm. If we denote the error by $e^{(j)} = y - y^{(j)}$, then by construction $e^{(j+1)} \perp Ap_i$. By Phythagoras' theorem it follows that

$$\|e^{(j+1)}\|_2^2 = \|e^{(j)}\|_2^2 - |\alpha_j|^2 \|Ap_j\|_2^2 \leq \|e^{(j)}\|_2^2,$$

i.e. this class of methods is error reducing.

We obtain the Gauss–Seidel method by taking p_j to be the unit vectors e_j in cyclic order. Then $Ap_j = a_j$, where a_j is the jth column of A. The iterative method (10.5.16) takes the form

$$y^{(j+1)} = y^{(j)} + a_j(c_j - a_j^T y^{(j)})/d_j, \quad j = 1; n. \tag{10.5.17}$$

This shows that if we take $x^{(0)} = Ay^{(0)}$, then for an arbitrary $y^{(0)}$ (10.5.17) is equivalent to the Gauss–Seidel method for (10.5.3). For the case of a square matrix A this method was originally devised by Kaczmarz [228, 1937].

The SOR method applied to the normal equations of the second kind can be obtained by introducing an acceleration parameter ω, i.e.

$$y^{(j+1)} = y^{(j)} + \omega a_j(c_j - a_j^T y^{(j)})/d_j, \quad j = 1 : n. \tag{10.5.18}$$

To obtain the SSOR method we perform a backward sweep $j = n : -1 : 1$ after each forward sweep in (10.5.18).

10.5.4 Krylov Subspace Methods for Least Squares

The implementation of CG applied to the normal equations of the first kind becomes as follows:

Algorithm 10.2.
CGLS

$r_0 = b - Ax_0; \quad p_0 = s_0 = A^T r_0;$

for $k = 0, 1, \ldots$ **while** $\|r_k\|_2 > \epsilon$ **do**

$\quad q_k = Ap_k;$

$\quad \alpha_k = \|s_k\|_2^2 / \|q_k\|_2^2;$

$\quad x_{k+1} = x_k + \alpha_k p_k;$

$\quad r_{k+1} = r_k - \alpha_k q_k;$

$\quad s_{k+1} = A^T r_{k+1};$

$\quad \beta_k = \|s_{k+1}\|_2^2 / \|s_k\|_2^2;$

$\quad p_{k+1} = s_{k+1} + \beta_k p_k;$

end

Note that it is important for the stability that the residuals $r_k = b - Ax_k$ and *not* the residuals $s_k = A^T(b - Ax_k)$ are recurred. The method obtained by applying CG to the normal equations of the second kind is also known as **Craig's Method**. This method can only be used for consistent problems, i.e., when $b \in \mathcal{R}(A)$. It can also be used to compute the (unique) minimum norm solution of an underdetermined system, $\min \|x\|_2$, subject to $Ax = b$, where $A \in \mathbf{R}^{m\times n}$, $m < n$.

Craig's method (CGNE) can be implemented as follows:

Algorithm 10.3.
CGNE

$$
\begin{aligned}
& r_0 = b - Ax_0; \quad p_0 = A^T r_0; \\
& \textbf{for } k = 0, 1, \dots \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do} \\
& \qquad \alpha_k = \|r_k\|_2^2 / \|p_k\|_2^2; \\
& \qquad x_{k+1} = x_k + \alpha_k p_k; \\
& \qquad r_{k+1} = r_k - \alpha_k A p_k; \\
& \qquad \beta_k = \|r_{k+1}\|_2^2 / \|r_k\|_2^2; \\
& \qquad p_{k+1} = A^T r_{k+1} + \beta_k p_k; \\
& \textbf{end}
\end{aligned}
$$

Both CGLS and CGNE will generate iterates in the shifted Krylov subspace,

$$x_k \in x_0 + \mathcal{K}_k(A^T r_0, A^T A).$$

From the minimization property we have for the iterates in CGLS

$$\|x - x_k\|_{A^T A} = \|r - r_k\|_2 < 2\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^k \|r_0\|_2,$$

where $\kappa = \kappa(A)$. Similarly for CGNE we have

$$\|y - y_k\|_{AA^T} = \|x - x_k\|_2 < 2\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^k \|x - x_0\|_2.$$

For consistent problems the method CGNE should in general be preferred.

The main drawback with the two above methods is that they often converge very slowly, which is related to the fact that $\kappa(A^T A) = \kappa^2(A)$. Note, however, that in some special cases both CGLS and CGNE may converge much faster than alternative methods. For example, when A is orthogonal then $A^T A = AA^T = I$ and both methods converge in one step!

As shown by Paige and Saunders the Golub–Kahan bidiagonalization process developed in Section 9.9.4 can be used for developing methods related to CGLS and CGNE for solving the linear least squares problem (10.5.2) and the minimum norm problem (10.5.3), respectively.

To compute a sequence of approximate solutions to the least squares problem we start the recursion (9.9.19)–(9.19.20) by

$$\beta_1 u_1 = b - Ax_0, \qquad \alpha_1 v_1 = A^T u_1, \tag{10.5.19}$$

and for $j = 1, 2, \ldots$ compute

$$\begin{aligned} \beta_{j+1} u_{j+1} &= A v_j - \alpha_j u_j, \\ \alpha_{j+1} v_{j+1} &= A^T u_{j+1} - \beta_{j+1} v_j, \end{aligned} \tag{10.5.20}$$

where $\beta_j \geq 0$ and $\alpha_j \geq 0$, $j \geq 1$, are determined so that $\|u_j\|_2 = \|v_j\|_2 = 1$.

After k steps we have computed orthogonal matrices

$$V_k = (v_1, \ldots, v_k), \qquad U_{k+1} = (u_1, \ldots, u_{k+1})$$

and a rectangular lower bidiagonal matrix

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} \in \mathbf{R}^{(k+1)\times k}. \tag{10.5.21}$$

The recurrence relations (10.5.19)–(10.5.20) can be written in matrix form as

$$U_{k+1} \beta_1 e_1 = r_0, \quad r_0 = b - Ax_0, \tag{10.5.22}$$

where e_1 denotes the first unit vector, and

$$AV_k = U_{k+1} B_k, \qquad A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \tag{10.5.23}$$

We now seek an approximate solution $x_k \in \mathcal{K}_k = \mathcal{K}_k(A^T r_0, A^T A)$. From the recursions (10.5.19)–(10.5.20) it follows that $\mathcal{K}_k = \operatorname{span}(V_k)$ and so we write

$$x_k = x_0 + V_k y_k. \tag{10.5.24}$$

Multiplying the first equation in (10.5.23) by y_k we obtain $Ax_k = AV_k y_k = U_{k+1} B_k y_k$, and then from (10.5.22)

$$b - Ax_k = U_{k+1} t_{k+1}, \qquad t_{k+1} = \beta_1 e_1 - B_k y_k. \tag{10.5.25}$$

Using the orthogonality of U_{k+1} and V_k, which holds in exact arithmetic, it follows that $\|b - Ax_k\|_2$ is minimized over all $x_k \in \operatorname{span}(V_k)$ by taking y_k to be the solution to the least squares problem

$$\min_{y_k} \|B_k y_k - \beta_1 e_1\|_2. \tag{10.5.26}$$

This forms the basis for the algorithm LSQR. Note the special form of the right-hand side, which holds because the starting vector was taken as b. Now $x_k = V_k y_k$

solves $\min_{x_k\in\mathcal{K}_k} \|Ax-b\|_2$, where $\mathcal{K}_k = \mathcal{K}_k(A^Tb, A^TA)$. Thus *mathematically* LSQR generates the same sequence of approximations as Algorithm 10.5.4 CGLS.

To solve (10.5.26) stably we need the QR factorization $Q_k^T B_k = R_k$. This can be computed by premultiplying B_k by a sequence Givens transformations, which are also applied to the right hand side e_1,

$$G_{k,k+1}G_{k-1,k}\cdots G_{12}(B_k \; e_1) = \begin{pmatrix} R_k & d_k \\ 0 & \rho_k \end{pmatrix}.$$

Here the rotation $G_{j,j+1}$ is used to zero the element β_{j+1}. It is easily verified that R_k is an upper bidiagonal matrix. The least squares solution y_k and the norm of the corresponding residual are then obtained from

$$R_k y_k = \beta e_1, \qquad \|b - Ax_k\|_2 = |\rho_k|.$$

Note that the whole vector y_k differs from y_{k-1}. An updating formula for x_k can be derived using an idea due to Paige and Saunders. With $W_k = V_k R_k^{-1}$ we can write

$$\begin{aligned} x_k &= x_0 + V_k y_k = x_0 + \beta_1 V_k R_k^{-1} d_k = x_0 + \beta_1 W_k d_k \\ &= x_0 + \beta_1 (W_{k-1}, \; w_k) \begin{pmatrix} d_{k-1} \\ \tau_k \end{pmatrix} = x_{k-1} + \beta_1 \tau_k w_k. \end{aligned} \tag{10.5.27}$$

Consider now the minimum norm problem for a consistent system $Ax = b$. Let L_k be the lower bidiagonal matrix formed by the first k rows of B_k

$$L_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{pmatrix} \in \mathbf{R}^{k\times k}. \tag{10.5.28}$$

The relations (10.5.23) can now be rewritten as

$$AV_k = U_k L_k + \beta_{k+1} u_{k+1} e_k^T, \qquad A^T U_k = V_k L_k^T. \tag{10.5.29}$$

The iterates x_k in Craig's method can be computed as

$$L_k y_k = \beta_1 e_1, \qquad x_k = V_k z_k. \tag{10.5.30}$$

Using (10.5.29) and (10.5.19) it follows that the residual vector satisfies

$$r_k = b - AV_k z_k = -\beta_{k+1} u_{k+1} (e_k^T z_k) = -\beta_{k+1} \eta_k u_{k+1},$$

and hence $U_k^T r_k = 0$. It can be shown that if $r_{k-1} \neq 0$ then $\alpha_k \neq 0$. Hence the vectors y_k and x_k can recursively be formed using

$$\eta_k = -\frac{\beta_k}{\alpha_k}\eta_{k-1}, \qquad x_k = x_{k-1} + \eta_k v_k.$$

10.5.5 Iterative Regularization.

In Section 8.4.1 we considered the solution of ill-posed linear problems by truncated SVD or Tikhonov regularization. When linear systems are derived from two- and three-dimensional ill-posed problems such direct methods become impractical. Instead iterative regularization algorithms can be used. Another application for such iterative methods are linear systems coming from the discretization of convolution-type integral equations. Then typically the matrix A is a Toeplitz matrix and matrix-vector products Ax and A^Ty can be computed in $O(n \log_2 n)$ multiplications using the fast Fourier transform;

In iterative methods for computing a regularized solution to the least squares problem (10.5.1), regularization is achieved by terminating the iterations before the unwanted irregular part of the solution has converged. Thus the regularization is controlled by the number of iterations carried out.

One of the earliest methods of the first class was proposed by Landweber [252, 1951], who considered the iteration

$$x_{k+1} = x_k + \omega A^T(b - Ax_k), \quad k = 0, 1, 2, \ldots. \tag{10.5.31}$$

Here ω is a parameter that should be chosen so that $\omega \approx 1/\sigma_1^2(A)$. The approximations x_k here will seem to converge in the beginning, before they deteriorate and finally diverge. This behavior is often called **semi convergence**. It can be shown that terminating the iterations with x_k gives behavior similar to truncating the singular value expansion of the solution for $\sigma_i \le \mu \sim k^{-1/2}$. Thus this method produces a *sequence* of less and less regularized solutions. Note that Landweber's method is equivalent to Richardson's stationary first-order method applied to the normal equations $A^T(Ax - b) = 0$.

More generally we can consider the iteration

$$x_{k+1} = x_k + p(A^TA)A^T(b - Ax_k), \tag{10.5.32}$$

where $p(\lambda)$ is a polynomial or rational function of λ. An important special case is the the **iterated Tikhonov method**

$$x_{k+1} = x_k + (A^TA + \mu^2 I)^{-1}A^T(b - Ax_k), \tag{10.5.33}$$

which corresponds to taking $p(\lambda) = (\lambda + \mu^2)^{-1}$.

Assume that $x_0 = 0$ in the iteration (10.5.32), which is no restriction. Then the kth iterate can be expressed in terms of the SVD of A. With $A = U\Sigma V^T$, $U = (u_1, \ldots, u_m)$, $V = (v_1, \ldots, v_n)$, we have

$$x_k = \sum_{i=1}^n \varphi_k(\sigma_i^2)\frac{u_i^Tb}{\sigma_i}v_i, \qquad \varphi_k(\lambda) = 1 - (1 - \lambda p(\lambda))^k, \tag{10.5.34}$$

where φ is called the **filter factors** after k iterations. From (10.5.34) it follows that the effect of terminating the iteration with x_k is to damp the component of the solution along v_i by the factor $\varphi_k(\sigma_i^2)$. For example, the filter function for the Landweber iteration is

$$\varphi_k(\lambda) = 1 - (1 - \omega\lambda)^k.$$

From this it is easily deduced that, after k iterations only the components of the solution corresponding to $\sigma_i \geq 1/k^{1/2}$ have converged.

We remark that the iteration (10.5.32) can be performed more efficiently using the factorized polynomial

$$1 - \lambda p(\lambda) = \prod_{i=1}^{d} (1 - \alpha_j \lambda).$$

One iteration in (10.5.32) can then be performed in d minor steps in the *nonstationary* Landweber iteration

$$x_{j+1} = x_j + \gamma_j A^T (b - A x_j), \qquad j = 0, 1, \ldots, d-1. \tag{10.5.35}$$

Assume that $\sigma_1 = \beta^{1/2}$, and that our object is to compute an approximation to the truncated singular value solution with a cut-off for singular values $\sigma_i \leq \sigma_c = \alpha^{1/2}$. Then it is well known that in a certain sense the optimal choice of the parameters in (10.5.35) are $\gamma_j = 1/\xi_j$, where

$$\xi_j = \frac{1}{2}(\alpha + \beta) + \frac{1}{2}(\alpha - \beta) x_k, \qquad x_k = \cos\left(\frac{\pi}{2}\frac{2j+1}{k}\right), \tag{10.5.36}$$

are the zeros of the Chebyshev polynomial of degree k on the interval $[\alpha, \beta]$. This choice gives a filter function $R(t)$ of degree k with $R(0) = 0$, and of least maximum deviation from one on $[\alpha, \beta]$. Thus there is no need to construct $p(\lambda)$ first in order to get the parameters γ_j in (10.5.35). Note that we have to pick α in advance, but it is possible to vary the regularization by using a decreasing sequence $\alpha = \alpha_1 > \alpha_2 > \alpha_3 > \cdots$.

Using standard results for Chebyshev polynomials it can be shown that if $\alpha \ll \beta$, then k steps in the iteration (10.5.35)–(10.5.36) reduce the regular part of the solution by the factor

$$\delta_k \approx 2e^{-2k(\alpha/\beta)^{1/2}}. \tag{10.5.37}$$

From this it follows that the cut-off σ_c for this method is related to the number of iteration steps k in (10.5.35) by $k \approx 1/\sigma_c$. This is a great improvement over Landweber's method, for which $k \approx (1/\sigma_c)^2$.

It is important to note that as it stands the iteration (10.5.35) with parameters (10.5.36) suffers severely from roundoff errors. This instability can be overcome by a reordering of the parameters ξ_j; see [6, 1972].

The CGLS (LSQR) and CGNE methods (Sec. 10.5.4) are well suited for computing regularized solutions, since they tend to converge quickly to the solution corresponding to the dominating singular values. With the smooth initial solution $x_0 = 0$ they generate a sequence of approximations x_k, $k = 1, 2, \ldots$, which minimize the quadratic form $\|Ax - b\|_2^2$ over the Krylov subspace

$$w_k \in \mathcal{K}_k(A^T A, A^T b).$$

CGLS and CGNE often converge much more quickly than competing iterative methods to the optimal solution of an ill-posed problem. Under appropriate conditions it can be dramatically faster; However, after the optimal number of iterations the CG method *diverges* much more rapidly than other methods, and hence it is essential to stop the iterations after the optimal number of steps. This is difficult since an a priori choice of the number of iterations as a function of the error level in the data is not possible. A complete understanding of the regularizing effect of Krylov subspace methods is still lacking.

The difficulty in finding reliable stopping rules for Krylov subspace methods can partly be solved by combining them with an inner regularizing algorithm. For example, the CGLS method can be applied to the regularized problem

$$\min_x \left\| \begin{pmatrix} A \\ \mu I_n \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2, \tag{10.5.38}$$

which has to be solved for several values of μ. This can be efficiently implemented using on the Golub–Kahan bidiagonalization (see Sec. 8.4.5) in the LSQR method. The kth approximation is taken to be $x_k(\mu) = V_k y_k(\mu)$, where $y_k(\mu)$ is the solution to

$$\min_{y_k} \left\| \begin{pmatrix} B_k \\ \mu I_k \end{pmatrix} y_k - \beta_1 \begin{pmatrix} e_1 \\ 0 \end{pmatrix} \right\|_2,$$

Since B_k is bidiagonal this can be solved in $O(k^2)$ flops for any given value of μ.

$$y_k(\mu) = \beta_1 \sum_{i=1}^{k} \frac{\omega_i p_{1i}}{\omega_i^2 + \mu^2} q_i,$$

Note that as in LSQR the vectors $v_1, \dots, v_k$ need to be saved or recomputed to construct $x_k(\mu)$. However, this need not be done except at the last iteration step.

Review Questions

1. To be written.

Problems and Computer Exercises

1. To be written.

10.6 Nonsymmetric Problems

An ideal conjugate gradient-like method for nonsymmetric systems would be characterized by one of the properties (10.4.7) or (10.4.11). We would also like to be able to base the implementation on a short vector recursion. Unfortunately, it turns out that such an ideal method essentially can only exist for matrices of very special form. In particular, a two term recursion like in the CG method is only possible in case A either has a minimal polynomial of degree ≤ 1, or is Hermitian, or is of the form

$$A = e^{i\theta}(B + \rho I), \qquad B = -B^H,$$

where θ and ρ are real. Hence the class essentially consists of shifted and rotated Hermitian matrices.

There are several possibilities for generalizing the conjugate gradient method to nonsymmetric systems. One simple approach is to apply the conjugate gradient method to the symmetrize system of normal equations. Since these usually have much higher spectral condition convergence of these methods can be very slow.

We can maintain the three-term relation by

10.6.1 Arnoldi's Method and GMRES

A serious drawback with using methods based on the normal equations is that they often converge very slowly, which is related to the fact that the singular values of $A^T A$ are the square of the singular values of A. There also are applications where it is not possible to compute matrix-vector products $A^T x$—note that A may only exist as subroutine for computing Ax.

We now consider a method for solving a general nonsymmetric system $Ax = b$ based on the Arnoldi process (see Section 9.9.6) with the starting vector

$$v_1 = b/\beta_1, \quad b = b - Ax_0, \quad \beta_1 = \|b\|_2,$$

In the following implementation of the Arnoldi process we perform the orthogonalization by the modified Gram-Schmidt method.

Algorithm 10.4.
The Arnoldi Process.

$\beta_1 = \|b\|_2; \quad v_1 = b/\beta_1;$
for $k = 1 : n$ **do**
  $z_k = Av_k;$
  for $i = 1 : k$ **do**
    $h_{ik} = z_k^T v_i;$
    $z_k = z_k - h_{ik} v_i;$
  end
  $h_{k+1,k} = \|z_k\|_2;$
  if $|h_{k+1,k}| < \epsilon$, *break* **end**
  $v_{k+1} = z_k / h_{k+1,k};$
end

In exact arithmetic the result after k steps is a matrix $V_k = (v_1, \dots, v_k)$, that (in exact arithmetic) gives an orthogonal basis for the Krylov subspace

$$\mathcal{K}_k(b, A) = \operatorname{span}(b, Ab, \dots, A^{k-1}b),$$

and a related square Hessenberg matrix $H_k = (h_{ij}) \in \mathbf{R}^{k \times k}$. Further we have

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T = V_{k+1} \bar{H}_k, \tag{10.6.1}$$

where

$$\bar{H}_k = \begin{pmatrix} H_k \\ h_{k+1,k} e_k^T \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ & \ddots & \ddots & \vdots \\ & & h_{k,k-1} & h_{kk} \\ & & & h_{k+1,k} \end{pmatrix} \in \mathbf{R}^{(k+1) \times k}. \tag{10.6.2}$$

We seek at step k an approximate solution of the form

$$x_k = x_0 + V_k y_k \in x_0 + \mathcal{K}_k(b, A), \tag{10.6.3}$$

There are two different ways to choose the approximation x_k. In the **full orthogonalization method (FOM)** x_k is determined by the Galerkin condition

$$r_k \perp \mathcal{K}_k(b, A), \quad r_k = b - Ax_k.$$

Using (10.6.1) the residual r_k can be expressed as

$$r_k = b - AV_k y_k = \beta_1 v_1 - V_k H_k y_k - h_{k+1,k} v_{k+1} e_k^T. = V_{k+1}(\beta_1 e_1 - \bar{H}_k y_k). \tag{10.6.4}$$

Using this the Galerkin condition gives $V_k^T r_k = \beta_1 e_1 - H_k y_k = 0$. Hence $y_k = \beta_1 H_k^{-1} e_1$ is obtained as the solution of a linear system of Hessenberg form. (Note that H_k is nonsingular if $\mathcal{K}_k(b, A)$ has full rank).

In the **generalized minimum residual (GMRES) method** y_k is chosen so that $\|b - Ax_k\|_2$ is minimized. Notice that this ensures that $\|r_k\|_2$ is monotonically decreasing as the iteration proceeds. Since (in exact arithmetic) V_{k+1} has orthogonal columns, $\|r_k\|_2$ is minimized by taking y_k to be the solution of the least squares problem

$$\min_{y_k} \|\beta_1 e_1 - \bar{H}_k y_k\|_2. \tag{10.6.5}$$

The Arnoldi process breaks down at step k if and only if $A^k b \in \mathcal{K}_k(b, A)$. Then z_k vanishes, $h_{k+1,k} = 0$ and $AV_k = V_k H_k$. Since $\operatorname{rank}(AV_k) = \operatorname{rank}(V_k) = k$ the matrix H_k is nonsingular. Then

$$r_k = V_k(\beta_1 e_1 - H_k y_k) = 0, \quad y_k = \beta_1 H_k^{-1} e_1,$$

and $x_k = x_0 + V_k y_k$ is the solution of $Ax = b$. This shows the important property (in exact arithmetic) that *GMRES does not break down before the exact solution is found.* It follows that GMRES terminates in at most n steps.

We now discuss the implementation of GMRES. To solve (10.6.5) we compute the QR factorization of the Hessenberg matrix $\bar{H}_k$. This can be done by using a sequence of k plane rotations. Let

$$Q_k^T(\bar{H}_k \ e_1) = \begin{pmatrix} R_k & d_k \\ 0 & \rho_k \end{pmatrix}, \quad Q_k^T = G_{k,k+1}G_{k-1,k}\cdots G_{12}, \tag{10.6.6}$$

where $G_{j+1,j}$ is chosen to zero the subdiagonal element $h_{j+1,j}$. Then the solution to (10.6.5) and its residual is given by

$$R_k y_k = \beta_1 d_k, \qquad \|r_k\|_2 = \beta_1|\rho_k|. \tag{10.6.7}$$

The iterations can be stopped as soon as $|\rho_k|$ is smaller than a prescribed tolerance.

Since $\bar{H}_{k-1}$ determines the first $k-1$ Givens rotations and $\bar{H}_k$ is obtained from $\bar{H}_{k-1}$ by adding the kth column, it is possible to save work by *updating the QR factorization* (10.6.6) at each step of the Arnoldi process. To derive the updating formulas for step $j = k$ we write

$$Q_k^T\bar{H}_k = G_{k,k+1}\begin{pmatrix} Q_{k-1}^T & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \bar{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{pmatrix} = \begin{pmatrix} R_{k-1} & c_{k-1} \\ 0 & \gamma_k \\ 0 & 0 \end{pmatrix},$$

We first apply the previous rotations to h_k giving

$$Q_{k-1}^T h_k = G_{k-1,k}\cdots G_{12}h_k = \begin{pmatrix} c_{k-1} \\ \delta_k \end{pmatrix}, \tag{10.6.8}$$

The rotation $G_{k,k+1}$ is determined by

$$G_{k,k+1}\begin{pmatrix} \delta_k \\ h_{k+1,k} \end{pmatrix} = \begin{pmatrix} \gamma_k \\ 0 \end{pmatrix}. \tag{10.6.9}$$

and gives the last element in the kth column in R_k.

Proceeding similarly with the right hand side, we have

$$Q_k^T e_1 = G_{k,k+1}\begin{pmatrix} Q_{k-1}^T e_1 \\ 0 \end{pmatrix} = G_{k,k+1}\begin{pmatrix} d_{k-1} \\ \rho_{k-1} \\ 0 \end{pmatrix} = \begin{pmatrix} d_{k-1} \\ \tau_k \\ \rho_k \end{pmatrix} \equiv \begin{pmatrix} d_k \\ \rho_k \end{pmatrix}. \tag{10.6.10}$$

(Note that the different dimensions of the unit vectors e_1 above is not indicated in the notation.) The first $k-1$ elements in $Q_k^T e_1$ are not changed.

The approximate solution can be obtained from $x_k = x_0 + V_k y_k$. Note that the whole vector y_k differs from y_{k-1} and therefore all the vectors $v_1, \ldots, v_k$ needs to be saved. Since $\|r_k\|_2 = |\rho_k|$ is available without forming x_k, this expensive operation can be delayed until until GMRES has converged, i.e., when ρ_k is small enough.

Alternatively, an updating formula for x_k can be derived as follows: Set $W_k R_k = V_k$, which can be written

$$(W_{k-1}, \ w_k)\begin{pmatrix} R_{k-1} & c_{k-1} \\ 0 & \gamma_k \end{pmatrix} = (V_{k-1}, \ v_k).$$

Equating the first block columns gives $W_{k-1}R_{k-1} = V_{k-1}$, which shows that the first $k-1$ columns of W_k equal W_{k-1}. Equating the last columns and solving for w_k we get

$$w_k = (v_k - W_{k-1}r_{k-1})/\gamma_k \tag{10.6.11}$$

Then from (10.5.27) $x_k = x_{k-1} + \beta_1 \tau_k w_k$. Note that if this formula is used we only need the last column of the matrix R_k. (We now need to save W_k but not R_k.)

The steps in the resulting GMRES algorithm can now be summarized as follows:

1. Obtain last column of $\bar{H}_k$ from the Arnoldi process and apply old rotations $g_k = G_{k-1,k} \cdots G_{12} h_k$.
2. Determine rotation $G_{k,k+1}$ and new column in R_k, i.e., c_{k-1} and γ_k according to (10.6.9). This also determines τ_k and $|\rho_k| = \|r_k\|_2$.
3. If x_{k-1} is recursively updated, then compute w_k using (10.6.10) and x_k from (10.5.27).

Suppose that the matrix A is diagonalizable,

$$A = X\Lambda X^{-1}, \quad \Lambda = \mathrm{diag}(\lambda_i).$$

Then, using the property that the GMRES approximations minimize the Euclidian norm of the residual $r_k = b - Ax_k$ in the Krylov subspace $\mathcal{K}_k(b, A)$, it can be shown that

$$\frac{||r_k||_2}{||b||_2} \le \kappa_2(X) \min_{q_k} \max_{i=1,2,\ldots,n} |q_k(\lambda_i)|, \tag{10.6.12}$$

where q_k is a polynomial of degree $\le k$ and $q_k(0) = 1$. The proof is similar to the convergence proof for the conjugate gradient method in Section 10.4.2. This results shows that if A has $p \le n$ distinct eigenvalues then, as for CG in the symmetric case, GMRES converges in at most p steps. If the spectrum is clustered in p clusters of sufficiently small diameters, then we can also expect GMRES to provide accurate approximations after about p iterations.

Because of the factor $\kappa_2(X)$ in (10.6.12) an upper bound for the rate of convergence can no longer be deduced from the spectrum $\{\lambda_i\}$ of A alone. In the special case that A is normal we have $\kappa_2(X) = 1$, and the convergence is related to the complex approximation problem

$$\min_{q_k} \max_{i=1,2,\ldots,n} |q_k(\lambda_i)|, \quad q_k(0) = 1.$$

Because complex approximation problems are harder than real ones, no simple results are available even for this special case.

In practice it is often observed that GMRES (like the CG method) has a so-called superlinear convergence. By this we mean that the rate of convergence improves as the iteration proceeds. It has been proved that this is related to the convergence of Ritz values to exterior eigenvalues of A. When this happens GMRES converges from then on as fast as for a related system in which these eigenvalues and their eigenvector components are missing.

The memory requirement of GMRES increases linearly with the number of steps k and the cost for orthogonalizing the vector Av_k is proportional to k^2. In practice the number of steps taken by GMRES must therefore often be limited. by **restarting** GMRES after each m iterations, where in practice typically m is between 10 and 30. We denote the corresponding algorithm GMRES(m). GMRES(m) cannot break down (in exact arithmetic) before the true solution has been produced, but for $m < n$ GMRES may never converge.

Since restarting destroys the accumulated information about the eigenvalues of A the superlinear convergence is usually lost. This loss can be compensated for by extracting form the computed Arnoldi factorization an approximate invariant subspace of A associated with the small eigenvalues. This is then used to precondition the restarted iteration.

If GMRES is applied to a real symmetric indefinite system, it can be implemented with a three-term recurrence, which avoids the necessity to store all basis vectors v_j. This leads to the method MINRES by Paige and Saunders mentioned in Section 10.6.1.

10.6.2 Lanczos Bi-Orthogonalization

The GMRES method is related to the reduction of a nonsymmetric matrix $A \in \mathbf{R}^{n\times n}$ to Hessenberg form by an orthogonal similarity $H = Q^TAQ$. It gives up the short recurrences of the CG method. Another possible generalization proposed by Lanczos [250] is related to the reduction of to tridiagonal form by a general similarity transformation.

Hence the stability of this process cannot be guaranteed, and this reduction is in general not advisable.

Assume that A can be reduced to tridiagonal form

$$W^TAV = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \gamma_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_n \\ & & & \gamma_n & \alpha_n \end{pmatrix},$$

where $V = (v_1, \dots, v_n)$ and $W = (w_1, \dots, w)$, are nonsingular and $W^TV = I$. The two vector sequences $\{v_1, \dots, v_n\}$ and $\{w_1, \dots, w\}$, then are **bi-orthogonal**, i.e.,

$$w_i^T v_j = \begin{cases} 1, & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{10.6.13}$$

Comparing columns in $AV = VT$ and $A^TW = WT^T$ we find (with $v_0 = w_0 = 0$) the recurrence relations

$$\gamma_{k+1}v_{k+1} = \tilde{v}_{k+1} = (A - \alpha_k I)v_k - \beta_k v_{k-1}, \tag{10.6.14}$$

$$\beta_{k+1}w_{k+1} = \tilde{w}_{k+1} = (A^T - \alpha_k I)w_k - \gamma_k w_{k-1}, \tag{10.6.15}$$

Multiplying equation (10.6.14) by w_k^T, and using the bi-orthogonality we have

$$\alpha_k = w_k^T A v_k.$$

To satisfy the bi-orthogonality relation (10.6.15) for $i = j = k+1$ it suffices to choose γ_{k+1} and β_{k+1} so that.

$$\gamma_{k+1}\beta_{k+1} = \tilde{w}_{k+1}^T \tilde{v}_{k+1}.$$

Hence there is some freedom in choosing these scale factors.

If we denote

$$V_k = (v_1, \ldots, v_k), \qquad W_k = (w_1, \ldots, w_k),$$

then we have $W_k^T A V_k = T_k$, and the recurrences in this process can be written in matrix form as

$$A V_k = V_k T_k + \gamma_{k+1} v_{k+1} e_k^T, \tag{10.6.16}$$

$$A^T W_k = W_k T_k^T + \beta_{k+1} w_{k+1} e_k^T. \tag{10.6.17}$$

By construction these vector sequences form basis vectors for the two Krylov spaces

$$\mathcal{R}(V_k) = \mathcal{K}_k(v_1, A), \qquad \mathcal{R}(W_k) = \mathcal{K}_k(w_1, A^T). \tag{10.6.18}$$

We summarize the algorithm for generating the two sequences of vectors $v_1, v_2, \ldots$ and $w_1, w_2, \ldots$:

Algorithm 10.5.

The Lanczos Bi-orthogonalization Process. Let v_1 and w_1 be two vectors such that $w_1^T v_1 = 1$. The following algorithm computes in exact arithmetic after k steps a symmetric tridiagonal matrix $T_k = \mathrm{trid}\,(\gamma_j, \alpha_j, \beta_{j+1})$ and two matrices W_k and V_k with bi-orthogonal columns spanning the Krylov subspaces $\mathcal{K}_k(v_1, A)$ and $\mathcal{K}_k(w_1, A^T)$:

$w_0 = v_0 = 0;$
$\beta_1 = \gamma_1 = 0;$
for $j = 1, 2, \ldots$
  $\alpha_j = w_j^T A v_j;$
  $v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1};$
  $w_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1};$
  $\delta_{j+1} = |w_{j+1}^T v_{j+1}|^{1/2};$
  if $\delta_{j+1} = 0$ **then** *exit*;
  $\beta_{j+1} = (w_{j+1}^T v_{j+1})/\delta_{j+1};$
  $v_{j+1} = v_{j+1}/\delta_{j+1};$
  $w_{j+1} = w_{j+1}/\beta_{j+1};$
end

Note that if $A = A^T$, $w_1 = v_1$, and we take $\beta_k = \gamma_k$, then the two sequences generated will be identical. The process then is equivalent to the symmetric Lanczos process.

There are two cases when the above algorithm breaks down. The first occurs when either $\tilde{v}_{k+1}$ or $\tilde{w}_{k+1}$ (or both) is null. In this case it follows from (10.6.16)–(10.6.17) that an invariant subspace has been found; if $v_{k+1} = 0$, then $AV_k = V_k T_k$ and $\mathcal{R}(V_k)$ is an A-invariant subspace. If $w_{k+1} = 0$, then $A^T W_k = W_k T_k^T$ and $\mathcal{R}(W_k)$ is an A^T-invariant subspace. This is called regular termination. The second case, called serious breakdown, occurs when $\tilde{w}_k^T \tilde{v}_k = 0$, with neither $\tilde{v}_{k+1}$ nor $\tilde{w}_{k+1}$ null.

10.6.3 Bi-Conjugate Gradient Method and QMR

We now consider the use of the nonsymmetric Lanczos process for solving a linear system $Ax = b$. Let x_0 be an initial approximation. Take $\beta_1 v_1 = r_0$, $\beta_1 = \|r_0\|_2$, and $w_1 = v_1$. We seek an approximate solution x_k such that

$$x_k - x_0 = V_k y_k \in \mathcal{K}_k(r_0, A).$$

For the residual we then have

$$r_k = b - Ax_k = \beta_1 v_1 - AV_k y_k,$$

Here y_k is determined so that the Galerkin condition $r_k \perp \mathcal{K}_k(w_1, A^T)$ is satisfied, or equivalently $W_k^T r_k = 0$. Using (10.6.16) and the bi-orthogonality conditions $W_k^T V_k = 0$ this gives

$$W_k^T(\beta_1 v_1 - AV_k y_k) = \beta_1 e_1 - T_k y_k = 0. \qquad (10.6.19)$$

Hence, if the matrix T_k is nonsingular x_k is determined by solving the tridiagonal system $T_k y_k = \beta_1 e_1$ and setting $x_k = x_0 + V_k y_k$.

If A is symmetric, this method becomes the SYMMLQ method, see Sec. 10.6.1. We remark again that in the nonsymmetric case this method can break down without producing a good approximate solution to $Ax = b$. In case of a serious breakdown, it is necessary to restart from the beginning with a new starting vector r_0. As in SYMMLQ the matrix T_k may be singular for some k and this is an additional cause for breakdown.

The Lanczos bi-orthogonalization algorithm is the basis for several iterative methods for nonsymmetric systems. The method can be written in a form more like the conjugate gradient algorithm, which is called the **bi-conjugate gradient** or Bi-CG method. The algorithm was first proposed by Lanczos [251] and later in conjugate gradient form by Fletcher [132]. The Bi-CG algorithm can be derived from Algorithm 10.5.4 in exactly the same way as the CG method was derived from the Lanczos algorithm. The algorithm solves not only the original system $Ax = b$ but also a dual linear system $A^T \tilde{x} = \tilde{b}$, although the dual system usually is ignored in the derivation of the algorithm.

To derive th Bi-CG algorithm from the Lanczos bi-orthogonalization we introduce the LU decomposition

$$T_k = L_k U_k,$$

and write

$$x_k = x_0 + V_k T_k^{-1}(\beta e_1) = x_0 + P_k L_k^{-1}(\beta e_1),$$

where $P_k = V_k U_k^{-1}$. Notice that x_k can be obtained by updating x_{k-1} as in the CG method.

Define similarly the matrix $\tilde{P}_k = W_k L_k^{-T}$. Then the columns of P_k and $\tilde{P}_k$ are A-conjugate, since

$$\tilde{P}_k^T A P_k = L_k^{-1} W_k^T A V_k U_k^{-1} = L_k^{-1} T_k U_k^{-1} = I.$$

Algorithm 10.6.
Bi-conjugate Gradient Algorithm Set $r_0 = b - Ax_0$ and choose $\tilde{r}_0$ so that $(r_=0, \tilde{r}_0) \neq 0$.

$$
\begin{aligned}
&p_0 = r_0; \quad \tilde{p}_0 = \tilde{r}_0;\\
&\rho_0 = (\tilde{r}_0, r_0);\\
&\textbf{for } j = 0, 1, 2, \ldots\\
&\qquad v_j = Ap_j; \quad \alpha_j = \rho_j / (\tilde{p}_j, v_j);\\
&\qquad x_{j+1} = x_j + \alpha_j p_j;\\
&\qquad r_{j+1} = r_j - \alpha_j v_j;\\
&\qquad \tilde{r}_{j+1} = \tilde{r}_j - \alpha_j (A^T \tilde{p}_j);\\
&\qquad \rho_{j+1} = (\tilde{r}_{j+1}, r_{j+1});\\
&\qquad \beta_j = \rho_{j+1} / \rho_j;\\
&\qquad p_{j+1} = r_{j+1} + \beta_j p_j;\\
&\qquad \tilde{p}_{j+1} = \tilde{r}_{j+1} + \beta_j \tilde{p}_j;\\
&\textbf{end}
\end{aligned}
$$

The vectors r_j and $\tilde{r}_j$ are in the same direction as v_{j+1} and w_{j+1}, respectively. Hence they form a biorthogonal sequence. Note that Bi-CG has the computational advantage over CGNE that the most time-consuming operations Ap_j and $A^T \tilde{p}_j$ can be carried out in parallel.

One can encounter convergence problems with Bi-CG, since for general matrices the bilinear form

$$[x, y] = (\psi(A^T)x, \psi(A)y)$$

used to define bi-orthogonality, does not define an inner product. Therefore if $\tilde{r}_0$ is chosen unfavorably, it may occur that ρ_j or $(\tilde{p}_j, v_j)$ is zero (or very small), without convergence having taken place.

Nothing is minimized in the Bi-CG and related methods, and for a general unsymmetric matrix A there is no guarantee that the algorithm will not break down or be unstable. On the contrary, it has been observed that sometimes convergence can be as fast as for GMRES. However, the convergence behavior can be very irregular, and as remarked above, breakdown occurs. Sometimes, breakdown can be avoided by a restart at the iteration step immediately before the breakdown step.

A related method called the **Quasi-Minimal Residual (QMR) method** can be developed as follows. After k steps of the nonsymmetric Lanczos process we have from the relation (10.6.16) that

$$AV_k = V_{k+1}\hat{T}_k, \quad \hat{T}_k = \begin{pmatrix} T_k \\ \gamma_{k+1}e_k^T \end{pmatrix},$$

where $\hat{T}_k$ is an $(k+1)\times k$ tridiagonal matrix. We can now proceed as was done in developing GMRES. If we take $v_1 = \beta r_0$, the the residual associated with with an approximate solution of the form $x_k = x_0 + V_k y$ is given by

$$\begin{aligned} b - Ax_k &= b - A(x_0 + V_k y) = r_0 - AV_k y \\ &= \beta v_1 - V_{k+1}\hat{T}_k y = V_{k+1}(\beta e_1 - \hat{T}_k y). \end{aligned} \tag{10.6.20}$$

Hence the norm of the residual vector is

$$\|b - Ax_k\|_2 = \|V_{k+1}(\beta e_1 - \hat{T}_k y)\|_2.$$

If the matrix V_{k+1} had orthonormal columns then the residual norm would become $\|(\beta e_1 - \hat{T}_k y)\|_2$, as in GMRES, and a least squares solution in the Krylov subspace could be obtained by solving

$$\min_y \|\beta e_1 - \hat{T}_k y\|_2.$$

for y_k and taking $x_k = x_0 + V_k y_k$.

Recent surveys on progress in iterative methods for non-symmetric systems are given by Freund, Golub and Nachtigal [141, 1991] and Golub and van der Vorst [182]. There is a huge variety of methods to choose from. Unfortunately in many practical situations it is not clear what method to select. In general there is no best method. In [286] examples are given which show that, depending on the linear system to be solved, each method can be clear winner or clear loser! Hence insight into the characteristics of the linear system is needed in order to discriminate between methods. This is different from the symmetric case, where the rate of convergence can be deduced from the spectral properties of the matrix alone.

10.6.4 Transpose-Free Methods

A disadvantage of the methods previously described for solving non-symmetric linear systems is that they require subroutines for the calculation of both Ax and $A^T y$ for arbitrary vectors x and y. If the data structure favors the calculation of Ax then it is often less favorable for the calculation of $A^T y$. Moreover, for some problems deriving from ordinary differential equations the rows of A arise naturally from a finite difference approximation and the matrix product Ax may be much more easily computed than $A^T y$. These consideration has lead to the development of "transpose-free" methods

The first of the transpose-free iterative methods **Bi-CGS**, due to Sonneveld [339], is a modification of the Bi-CG algorithm.. Here CGS stands for "conjugate gradient squared". The key observation behind this algorithm is the following property of the vectors generated in the Bi-CG algorithm. Taking into account that $p_0 = r_0$, it is easily showed that there are polynomials $\phi_j(x)$ and $psi_j(x)$ of degree such that for $j = 1, 2, 3, \ldots,$

$$\begin{aligned} r_j &= \phi_j(A)\, r_0, & \tilde{r}_j &= \phi_j(A^T)\, \tilde{r}_0, \\ p_j &= \psi_j(A)\, r_0, & \tilde{p}_j &= \psi_j(A^T)\, \tilde{r}_0. \end{aligned}$$

That is r_j and $\tilde{r}_j$ are obtained by premultiplication by *the same polynomial* $\phi(t)$ in A and A^T, respectively. The same is true for p_j and $\tilde{p}_j$ for the polynomial $\psi(t)$. Using the fact that the polynomial of a transpose matrix is the transpose of the polynomial, it follows that the quantities needed in the Bi-CG algorithm can be expressed as

$$(\tilde{r}_j, r_j) = (\tilde{r}_0, \phi_j^2(A)\, r_0), \qquad (\tilde{p}_j, Ap_j) = (\tilde{p}_0, \psi_j^2(A)\, r_0).$$

Therefore, if we somehow could generate the vectors $\phi_j(A)]^2 r_0$ and $\psi_j(A)]^2 p_0$ directly, then no products with A^T would be required. To achieve this we note that from the Bi-CG algorithm we have the relations, $\phi_0(A) = \psi_0(A) = I$,

$$\phi_{j+1}(A) = \phi_j(A) - \alpha_j A\, \psi_j(A), \tag{10.6.21}$$
$$\psi_{j+1}(A) = \phi_{j+1}(A) + \beta_j \psi_j(A), \tag{10.6.22}$$

Squaring these relations we obtain

$$\begin{aligned} \phi_{j+1}^2 &= \phi_j^2 - 2\alpha_j A\, \phi_j \psi_j + \alpha_j^2 A^2 \psi_j^2, \\ \psi_{j+1}^2 &= \phi_{j+1}^2 + 2\beta_j \phi_{j+1} \psi_j + \beta_j^2 \psi_j^2. \end{aligned}$$

where we have omitted the argument A. For the first cross product term we have using (10.6.22)

$$\phi_j \psi_j = \phi_j(\phi_j + \beta_{j-1}\psi_{j-1}) = \phi_j^2 + \beta_{j-1}\phi_j \psi_{j-1}.$$

From this and (10.6.21) we get for the other cross product term

$$\phi_{j+1}\psi_j = (\phi_j - \alpha_j A\, \psi_j)\psi_j = \phi_j \psi_j - \alpha_j A\, \psi_j^2 = \phi_j^2 + \beta_{j-1}\phi_j \psi_{j-1} - \alpha_j A\, \psi_j^2.$$

Summarizing, we now have the three recurrence relations, which are the basis of the Bi-CGS algorithm:

$$\begin{aligned} \phi_{j+1}^2 &= \phi_j^2 - \alpha_j A\, (2\phi_j^2 + 2\beta_{j-1}\phi_j \psi_{j-1} - \alpha_j A\, \psi_j^2), \\ \phi_{j+1}\psi_j &= \phi_j^2 + \beta_{j-1}\phi_j \psi_{j-1} - \alpha_j A\, \psi_j^2 \\ \psi_{j+1}^2 &= \phi_{j+1}^2 + 2\beta_j \phi_{j+1}\psi_j + \beta_j^2 \psi_j^2. \end{aligned}$$

If we now define

$$r_j = \phi_j^2(A)r_0, \qquad q_j = \phi_{j+1}(A)\psi_j(A)r_0, \qquad p_j = \psi_j^2(A)r_0. \tag{10.6.23}$$

we get

$$r_{j+1} = r_j - \alpha_j A\,(2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j A\,p_j), \tag{10.6.24}$$

$$q_j = r_j + \beta_{j-1}q_{j-1} - \alpha_j A\,p_j, \tag{10.6.25}$$

$$p_{j+1} = r_{j+1} + 2\beta_j q_j + \beta_j^2 p_j. \tag{10.6.26}$$

These recurrences can be simplified by introducing the auxiliary vectors

$$u_j = r_j + \beta_{j-1}q_{j-1}, \qquad d_j = u_j + q_j. \tag{10.6.27}$$

The resulting algorithm is given below.

Algorithm 10.7.
Bi-CGS Algorithm Set $r_0 = b - Ax_0$ and choose $\tilde{r}_0$ so that $(r_0, \tilde{r}_0) \neq 0$.

$$
\begin{aligned}
&p_0 = u_0 = r_0; \quad \rho_0 = (\tilde{r}_0, r_0);\\
&\textbf{for } j = 0, 1, 2, \dots\\
&\qquad v_j = Ap_j; \quad \alpha_j = \rho_j/(\tilde{r}_0, v_j);\\
&\qquad q_j = u_j - \alpha_j v_j;\\
&\qquad d_j = u_j + q_j;\\
&\qquad x_{j+1} = x_j + \alpha_j d_j;\\
&\qquad r_{j+1} = r_j - \alpha_j A\,d_j;\\
&\qquad \rho_{j+1} = (\tilde{r}_0, r_{j+1});\\
&\qquad \beta_j = \rho_{j+1}/\rho_j;\\
&\qquad u_{j+1} = r_{j+1} + \beta_j q_j;\\
&\qquad p_{j+1} = u_{j+1} + \beta_j(q_j + \beta_j p_j);\\
&\textbf{end}
\end{aligned}
$$

There are now two matrix-vector multiplications with A in each step. When Bi-CG converges well we can expect Bi-CGS to converge about twice as fast.

Although the Bi-CGS algorithm often is competitive with other methods such as GMRES, a weak point of Bi-CGS is that the residual norms may behave very erratically, in particular when the iteration is started close to the solution. For example, although the norm of of the vector $\psi_j(A)r_0$ is small it may happen that $\|\psi_j^2(A)r_0\|$ is much bigger than $\|r_0\|$. This may even lead to such severe cancellation that the accuracy of the computed solution is spoilt.

This problem motivated the development a stabilized version called Bi-CGSTAB by van der Vorst [367]), which is more smoothly converging. Instead of computing the residuals $\psi_j^2(A)r_0$, this algorithm uses

$$r_j = \chi_j(A)\psi_j(A)r_0, \quad \chi_j(t) = (1 - \omega_1 t)(1 - \omega_2 t)\cdots(1 - \omega_j t), \tag{10.6.28}$$

where the constants ω_j are determined so that $\|r_j\|_2$ is minimized as a function of ω_j.

From the orthogonality property $(\psi_i(A)r_0, \chi_j(A)r_0) = 0$, for $j < i$, it follows that Bi-CGSTAB is a finite method, i.e. in exact arithmetic it will converge in at most n steps.

Algorithm 10.8.
Bi-CGSTAB Algorithm Let x_0 be an initial guess, $r_0 = b - Ax_0$ and choose $\tilde{r}_0$ so that $(\tilde{r}_0, r_0) \neq 0$.

$$
\begin{aligned}
& p_0 = u_0 = r_0; \quad \rho_0 = (\tilde{r}_0, r_0); \\
& \textbf{for } j = 0, 1, 2, \dots \\
& \qquad v_j = Ap_j; \quad \alpha_j = \rho_j/(\tilde{r}_0, v_j); \\
& \qquad s_j = r_j - \alpha_j v_j; \\
& \qquad t_j = Ap_j; \\
& \qquad \omega_j = (t_j, s_j)/(t_j, t_j); \\
& \qquad q_j = u_j - \alpha_j v_j; \\
& \qquad d_j = u_j + q_j; \\
& \qquad x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j; \\
& \qquad r_{j+1} = s_j - \omega_j t_j; \\
& \qquad \rho_{j+1} = (\tilde{r}_0, r_{j+1}); \\
& \qquad \beta_j = (\rho_{j+1}/\rho_j)(\alpha_j/\omega_j); \\
& \qquad p_{j+1} = r_{j+1} + \beta_j(p_j - \omega_j v_j); \\
& \textbf{end}
\end{aligned}
$$

As for Bi-CGS this algorithm requires two matrix-vector products with A.

Review Questions

1. What optimality property does the residual vectors $r_k = b - Ax_k$ in the GMRES method satisfy. In what subspace does the vector $r_k - r_0$ lie?
2. In Lanczos bi-orthogonalization bases for two different Krylov subspaces are computed. Which subspaces and what property has these bases?
3. (a) The bi-conjugate gradient (Bi-CG) method is based on the reduction of $A \in \mathbf{C}^{n\times n}$ to tridiagonal form by a general similarity transformation.

   (b) What are the main advantages and drawbacks of the Bi-CG method compared to GMRES.

   c) How are the approximations x_k defined in QMR?

Problems and Computer Exercises

1. Derive Algorithms CGLS and CGNE by applying the conjugate gradient algorithm to the normal equations $A^TAx = A^Tb$ and $AA^Ty = b$, $x = A^Ty$, respectively.

2. Consider using GMRES to solve the system $Ax = b$, where

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

using $x_0 = 0$. Show that $x_1 = 0$, and that therefore GMRES(1) will never produce a solution.

10.7 Preconditioned Iterative Methods

Preconditioned iterative methods can be viewed as a compromise between a direct and iterative solution method. General purpose techniques for constructing preconditioners have made iterative methods successful in many industrial applications.

The term "preconditioning" dates back to Turing in 1948, and is in general taken to mean the transformation of a problem to a form that can more efficiently be solved. In order to be effective iterative methods must usually be combined with a (nonsingular) preconditioning matrix M, which in some sense is an approximation to A. The original linear system $Ax = b$ is then transformed by considering the **left-preconditioned system**

$$M^{-1}Ax = M^{-1}b. \tag{10.7.1}$$

or **right-preconditioned system**

$$AM^{-1}u = b, \qquad u = Mx. \tag{10.7.2}$$

The idea is to choose M so that the rate of convergence of the iterative method is improved. Note that the product $M^{-1}A$ (or AM^{-1}) should never be formed. The preconditioned iteration is instead implemented by forming matrix vector products with A and M^{-1} separately. Since forming $u = M^{-1}v$ for an arbitrary vector v is equivalent to solving a linear system $Mu = v$, the inverse M^{-1} is not needed either.

Often the rate of convergence depends on the spectrum of the transformed matrix. Since the eigenvalues of $M^{-1}A$ and AM^{-1} are the same, we see that the main difference between these two approaches is that the actual residual norm is available in the right-preconditioned case.

If A is symmetric, positive definite, the preconditioned system should also have this property. To ensure this we take M symmetric, positive definite and consider a **split preconditioner**. Let $M = LL^T$ where L is the Cholesky factor of M, and set $\tilde{A}\tilde{x} = \tilde{b}$, where

$$\tilde{A} = L^{-1}AL^{-T}, \qquad \tilde{x} = L^Tx, \qquad \tilde{b} = L^{-1}b. \tag{10.7.3}$$

Then $\tilde{A}$ is symmetric positive definite. Note that the spectrum of $A = L^{-1}AL^{-T}$ is the same as for $L^{-T}L^{-1}A = M^{-1}A$.

A preconditioner should typically satisfy the following conditions:

(i) $M^{-1}A = I + R$, where $\|R\|$ is small.

(ii) Linear systems of the form $Mu = v$ should be easy to solve.

(iii) $\mathrm{nz}(M) \approx \mathrm{nz}(A)$.

Condition (i) implies fast convergence, (ii) that the arithmetic cost of preconditioning is reasonable, and (iii) that the storage overhead is not too large. Obviously these conditions are contradictory and a compromise must be sought. For example, taking $M = A$ is optimal in the sense of (i), but obviously this choice is ruled out by (ii).

The choice of preconditioner is strongly problem dependent and possibly the most crucial component in the success of an iterative method! A preconditioner which is expensive to compute may become viable if it is to be used many times, as may be the case, e.g., when dealing with time-dependent or nonlinear problems. It is also dependent on the architecture of the computing system. Preconditioners that are efficient in a scalar computing environment may show poor performance on vector and parallel machines.

10.7.1 The Preconditioned CG Method

The conjugate gradient algorithm 10.4.1 can be applied to linear systems $Ax = b$, where A is symmetric positive definite. In this case it is natural to use a split preconditioner and consider the system (10.7.3).

In implementing the preconditioned conjugate gradient method we need to form matrix-vector products of the form $t = \tilde{A}p = L^{-1}(A(L^{-T}p))$. These can be calculated by solving two linear systems and performing one matrix multiplication with A as

$$L^T q = p, \qquad s = Aq, \qquad Lt = s.$$

Thus, the extra work per step in using the preconditioner essentially is to solve two linear systems with matrix L^T and L respectively.

The preconditioned algorithm will have recursions for the transformed variables and residuals vectors $\tilde{x} = L^T x$ and $\tilde{r} = L^{-1}(b - Ax)$. It can be simplified by reformulating it in terms of the original variables x and residual $r = b - Ax$. It is left as an exercise to show that if we let $p_k = L^{-T}\tilde{p}_k$, $z_k = L^{-T}\tilde{r}_k$, and

$$M = LL^T, \tag{10.7.4}$$

we can obtain the following implementation of the **preconditioned conjugate gradient method**:

Algorithm 10.9.
Preconditioned Conjugate Gradient Method

$$r_0 = b - Ax_0; \quad p_0 = z_0 = M^{-1}r_0;$$

for $k = 0, 1, 2, \ldots,$ **while** $\|r_k\|_2 > \epsilon$ **do**

$w = Ap_k;$

$\beta_k = (z_k, r_k)/(p_k, Ap_k);$

$x_{k+1} = x_k + \beta_k p_k;$

$r_{k+1} = r_k - \beta_k Ap_k;$

$z_{k+1} = M^{-1} r_{k+1};$

$\beta_k = (z_{k+1}, r_{k+1})/(z_k, r_k);$

$p_{k+1} = z_{k+1} + \beta_k p_k;$

end

A surprising and important feature of this version is that it depends only on the symmetric positive definite matrix $M = LL^T$.

The rate of convergence in the $\tilde{A}$-norm depends on $\kappa(\tilde{A})$, see (10.4.21). Note, however, that

$$\|\tilde{x} - \tilde{x}_k\|^2_{\tilde{A}} = (\tilde{x} - \tilde{x}_k)^T L^{-1} A L^{-T} (\tilde{x} - \tilde{x}_k) = \|x - x_k\|^2_A,$$

so the rate of convergence in A-norm of the error in x also depends on $\kappa(\tilde{A})$. The preconditioned conjugate gradient method will have rapid convergence if one or both of the following conditions are satisfied:

i. $M^{-1}A$ to have small condition number, or

ii. $M^{-1}A$ to have only few distinct eigenvalues.

For symmetric indefinite systems SYMMLQ can be combined with a positive definite preconditioner M. To solve the symmetric indefinite system $Ax = b$ the preconditioner is regarded to have the form $M = LL^T$ and SYMMLQ *implicitly* applied to the system

$$L^{-1} A L^{-T} w = L^{-1} b.$$

The algorithm accumulates approximations to the solution $x = L^{-T} w$, without approximating w. A MATLAB implementation of this algorithm, which only requires solves with M, is given by Gill et al. [164].

To precondition CGLS Algorithm (10.5.4) it is natural to use a right preconditioner $S \in \mathbf{R}^{n\times n}$, i.e., perform the transformation of variables

$$\min_y \|(AS^{-1})y - b\|_2, \quad Sx = y.$$

(Note that for a nonconsistent system $Ax = b$ a left preconditioner would change the problem.) If we apply CGLS to this problem and formulate the algorithm in terms of the original variables x, we obtain the following algorithm:

Algorithm 10.10.
Preconditioned CGLS.

$$
\begin{aligned}
&r_0 = b - Ax_0; \quad p_0 = s_0 = S^{-T}(A^T r_0);\\
&\textbf{for } k = 0, 1, \ldots \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do}\\
&\qquad t_k = S^{-1} p_k;\\
&\qquad q_k = A t_k;\\
&\qquad \alpha_k = \|s_k\|_2^2 / \|q_k\|_2^2;\\
&\qquad x_{k+1} = x_k + \alpha_k t_k;\\
&\qquad r_{k+1} = r_k - \alpha_k q_k;\\
&\qquad s_{k+1} = S^{-T}(A^T r_{k+1});\\
&\qquad \beta_k = \|s_{k+1}\|_2^2 / \|s_k\|_2^2;\\
&\qquad p_{k+1} = s_{k+1} + \beta_k p_k;\\
&\textbf{end}
\end{aligned}
$$

For solving a consistent underdetermined systems we can derive a preconditioned version of CGNE. Here it is natural to use a left preconditioner S, and apply CGNE to the problem

$$\min \|x\|_2, \qquad S^{-1}Ax = S^{-1}b,$$

i.e., the residual vectors are transformed. If the algorithm is formulated in terms of the original residuals, the following algorithm results:

Algorithm 10.11.
Preconditioned CGNE

$$
\begin{aligned}
&r_0 = b - Ax_0; \quad z_0 = S^{-1} r_0; \quad p_0 = A^T(S^{-T} z_0);\\
&\textbf{for } k = 0, 1, \ldots \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do}\\
&\qquad \alpha_k = \|z_k\|_2^2 / \|p_k\|_2^2;\\
&\qquad x_{k+1} = x_k + \alpha_k p_k;\\
&\qquad r_{k+1} = r_k - \alpha_k A p_k;\\
&\qquad z_{k+1} = S^{-1} r_{k+1};\\
&\qquad \beta_k = \|z_{k+1}\|_2^2 / \|z_k\|_2^2;\\
&\qquad p_{k+1} = A^T(S^{-T} z_{k+1}) + \beta_k p_k;\\
&\textbf{end}
\end{aligned}
$$

Algorithm PCCGLS still minimizes the error functional $\|\hat{r} - r^{(k)}\|_2$, where $r = b - Ax$, but over a different Krylov subspace

$$x_k = x_0 + \mathcal{K}_k, \qquad \mathcal{K}_k = (S^{-1}S^{-T}A^TA, S^{-1}S^{-T}A^Tr_0).$$

Algorithm PCCGNE minimizes the error functional $\|\hat{x} - x^{(k)}\|_2$, over the Krylov subspace

$$x_k = x_0 + \mathcal{K}_k, \qquad \mathcal{K}_k = (A^TS^{-T}S^{-1}A, A^TS^{-T}S^{-1}r_0).$$

The rate of convergence for PCGTLS depends on $\kappa(AS^{-1})$, and for PCCGNE on $\kappa(S^{-1}A) = \kappa(A^TS^{-T})$.

10.7.2 Preconditioned GMRES

For nonsymmetric linear systems there are two options for applying the preconditioner. We can use the left preconditioned system (10.7.1) or the right preconditioned system (10.7.2). (If A is almost symmetric positive definite, then a split preconditioner might also be considered.) The changes to the GMRES algorithm are small.

In the case of using a left preconditioner M only the following changes in the Arnoldi algorithm are needed. We start the recursion with

$$r_0 = M^{-1}(b - Ax_0), \quad \beta_1 = \|r_0\|_2; \quad v_1 = r_0/\beta_1,$$

and define

$$z_j = M^{-1}Av_j, \quad j = 1 : k.$$

All computed residual vectors will be preconditioned residuals $M^{-1}(b - Ax_m)$. This is a disadvantage since most stopping criteria use the actual residuals $r_m = b - Ax_m$. In this left preconditioned version the transformed residual norm $\|M^{-1}(b - Ax)\|_2$ will be minimized among all vectors of the form

$$x_0 + \mathcal{K}_m(M^{-1}r_0, M^{-1}A). \tag{10.7.5}$$

In the right preconditioned version of GMRES the actual residual vectors are used, but the variables are transformed according to $u = Mx$ ($x = M^{-1}u$). The right preconditioned algorithm can easily be modified to give the untransformed solution. We have

$$z_j = AM^{-1}v_j, \quad j = 1 : k.$$

The kth approximation is $x_k = x_0 + M^{-1}V_ky_k$, where y_k solves

$$\min_{y_k} \|\beta_1e_1 - \bar{H}_ky_k\|_2.$$

As before this can be written as

$$x_k = x_{k-1} + \beta_1\tau_kM_k^{-1}w_k, \quad w_k = R_ky_k,$$

see (10.5.27).

In the right preconditioned version the residual norm $\|b - AM^{-1}u\|_2$ will be minimized among all vectors of the form $u_0 + \mathcal{K}_m(r_0, AM^{-1})$. However, this is equivalent to minimizing $\|b - AM^{-1}u\|_2$ among all vectors of the form

$$x_0 + M^{-1}\mathcal{K}_m(r_0, AM^{-1}). \tag{10.7.6}$$

Somewhat surprisingly the two affine subspaces (10.7.5) and (10.7.6) are the same! The jth vector in the two Krylov subspaces are $w_j = (M^{-1}A)^j M^{-1} r_0$ and $\tilde{w}_j = M^{-1}(AM^{-1})^j r_0$. By a simple induction proof it can be shown that $M^{-1}(AM^{-1})^j = (M^{-1}A)^j M^{-1}$ and so $\tilde{w}_j = w_j$, $j \geq 0$. Hence the left and right preconditioned versions generate approximations in the same Krylov subspaces, and they differ only with respect to which error norm is minimized.

For the case when A is diagonalizable, $A = X\Lambda X^{-1}$, where $\Lambda = \operatorname{diag}(\lambda_i)$ we proved the error estimate

$$\frac{||r_k||_2}{||r_0||_2} \leq \kappa_2(X) \min_{q_k} \max_{i=1,2,\dots,n} |q_k(\lambda_i)|, \tag{10.7.7}$$

where q_k is a polynomial of degree $\leq k$ and $q_k(0) = 1$. Because of the factor $\kappa_2(X)$ in (10.7.7) the rate of convergence can no longer be deduced from the spectrum $\{\lambda_i\}$ of the matrix A alone. Since the spectrum of $M^{-1}A$ and AM^{-1} are the same we can expect the convergence behavior to be similar if A is close to normal.

10.7.3 Preconditioners from Matrix Splittings

The stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \qquad k = 0, 1, \dots, \tag{10.7.8}$$

corresponds to a matrix splitting $A = M - N$, and the iteration matrix

$$B = M^{-1}N = I - M^{-1}A.$$

The iteration (10.7.8) can be considered as a fixed point iteration applied to the preconditioned system $M^{-1}Ax = M^{-1}b$. Hence, the basic iterative methods considered in Sections 10.1.4 give simple examples of preconditioners.

The Jacobi and Gauss–Seidel methods are both special cases of one-step stationary iterative methods. Using the standard splitting $A = D_A - L_A - U_A$, where D_A is diagonal, L_A and U_A are strictly lower and upper triangular, these methods correspond to the matrix splittings

$$M_J = D_A, \quad \text{and} \quad M_{GS} = D_A - L_A.$$

If A is symmetric positive definite then $M_J = D_A > 0$ and symmetric. However, M_{GS} is lower triangular and unsymmetric.

The simplest choice related to this splitting is to take $M = D_A$. This corresponds to a diagonal scaling of the rows of A, such that the scaled matrix $M^{-1}A = D_A^{-1}A$ has a unit diagonal. For s.p.d. matrices symmetry can be preserved by using a split preconditioner with $L = L^T = D_A^{1/2}$. In this case it can be shown that this is close to the optimal diagonal preconditioning.

Lemma 10.7.1. Van der Sluis [365]

Let $A \in \mathbf{R}^{n\times n}$ be a symmetric positive definite matrix with at most $q \le n$ nonzero elements in any row. Then if $A = D_A - L_A - L_A^T$, it holds that

$$\kappa(D_A^{-1/2} A D_A^{-1/2}) = q \min_{D>0} \kappa(DAD).$$

Although diagonal scaling may give only a modest improvement in the rate of convergence it is trivial to implement and therefore recommended even if no other preconditioning is carried out.

In Section 10.2.2 it was shown that for a symmetric matrix A the SSOR iteration method corresponds to a splitting with the matrix

$$M_{SSOR} = \frac{1}{\omega(2-\omega)}\big(D_A - \omega L_A\big) D_A^{-1}\big(D_A - \omega U_A\big).$$

Since M_{SSOR} is given in the form of an LDL^T factorization it is easy to solve linear systems involving this preconditioner. It also has the same sparsity as the original matrix A. For $0 < \omega < 2$, if A is s.p.d. so is M_{SSOR}.

The performance of the SSOR splitting turns out to be fairly insensitive to the choice of ω. For systems arising from second order boundary values problems, e.g., the model problem studied previously, the original condition number $\kappa(A) = \mathcal{O}(h^{-2})$ can be reduced to $\kappa(M^{-1}A) = \mathcal{O}(h^{-1})$. Taking $\omega = 1$ is often close to optimal. This corresponds to the symmetric Gauss–Seidel (SGS) preconditioner

$$M_{SGS} = \big(D_A - L_A\big) D_A^{-1}\big(D_A - U_A\big).$$

All the above preconditioners satisfy conditions (ii) and (iii). The application of the preconditioners involves only triangular solves and multiplication with a diagonal matrix. They are all defined in terms of elements of the original matrix A, and hence do not require extra storage. However, they may not be very effective with respect to the condition (i).

10.7.4 Incomplete LU Factorizations

The SGS preconditioner has the form $M_{SGS} = LU$ where $L = \big(I - L_A^T D_A^{-1}\big)$ is lower triangular and $U = \big(D_A - L_A^T\big)$ upper triangular. To find out how well M_{SGS} approximates A we form the **defect matrix** ixdefect matrix

$$A - LU = D_A - L_A - U_A - \big(I - L_A D_A^{-1}\big)\big(D_A - U_A\big) = -L_A D_A^{-1} U_A.$$

An interesting question is whether we can find matrices L and U with the same nonzero structure as above, but with a smaller defect matrix $R = LU - A$.

We now develop an important class of preconditioners obtained from so called **incomplete LU-factorizations** of A. The idea is to compute a lower triangular matrix L and an upper triangular matrix U with a *prescribed* sparsity structure such that

$$A = LU - R,$$

with R small. Such incomplete LU-factorizations can be realized by performing a modified Gaussian elimination on A, in which elements are allowed only in specified places in the L and U matrices. We assume that these places (i,j) are given by the index set

$$\mathcal{P} \subset \mathcal{P}_n \equiv \{(i,j) \,|\, 1 \le i,j \le n\},$$

where the diagonal positions always are included in $\mathcal{P}$. For example, we could take $\mathcal{P} = \mathcal{P}_A$, the set of nonzero elements in A.

Algorithm 10.12.
Incomplete LU Factorization

> **for** $k = 1, \dots, n-1$
>   **for** $i = k+1, \dots, n$
>   **if** $(i,k) \in \mathcal{P}$ $\;l_{ik} = a_{ik}/a_{kk}$;
>   **for** $j = k+1, \dots, n$
>   **if** $(k,j) \in \mathcal{P}$ $\;a_{ij} = a_{ij} - l_{ik}a_{kj}$;
>   **end**
>   **end**
> **end**

The elimination consists of $n-1$ steps. In the kth step we first subtract from the current matrix elements with indices (i,k) and $(k,i) \notin \mathcal{P}$ and place in a defect matrix R_k. We then carry out the $k$the step of Gaussian elimination on the so modified matrix. This process can be expressed as follows. Let $A_0 = A$ and

$$\widetilde{A}_k = A_{k-1} + R_k, \qquad A_k = L_k\widetilde{A_k}, \quad k = 1, \dots, n-1.$$

Applying this relation recursively we obtain

$$\begin{aligned} A_{n-1} &= L_{n-1}\widetilde{A}_{n-1} = L_{n-1}A_{n-2} + L_{n-1}R_{n-1} \\ &= L_{n-1}L_{n-2}A_{n-3} + L_{n-1}L_{n-2}R_{n-2} + L_{n-1}R_{n-1} \\ &= L_{n-1}L_{n-2}\cdots L_1 A + L_{n-1}L_{n-2}\cdots L_1 R_1 \\ &\quad + \cdots + L_{n-1}L_{n-2}R_{n-2} + L_{n-1}R_{n-1}. \end{aligned}$$

We further notice that since the first $m-1$ rows of R_m are zero it follows that $L_kR_m = R_m$, if $k < m$. Then by combining the above equations we find $LU = A+R$, where

$$U = A_{n-1}, \qquad L = (L_{n-1}L_{n-2}\cdots L_1)^{-1}, \qquad R = R_1 + R_2 + \cdots R_{n-1}.$$

Algorithm 10.8.1 can be improved by noting that any elements in the resulting $(n-k) \times (n-k)$ lower part of the reduced matrix not in $\mathcal{P}$ need not be carried along and can also be included in the defect matrix R_k. This is achieved simply by changing line five in the algorithm to

$$\textbf{if } (k,j) \in \mathcal{P} \textbf{ and } (i,j) \in \mathcal{P} \quad a_{ij} = a_{ij} - l_{ik}a_{kj};$$

In practice the matrix A is sparse and the algorithm should be specialized to take this into account. In particular, a version where A is processed a row at a time is more convenient for general sparse matrices. Such an algorithm can be derived by interchanging the k and i loops in Algorithm 10.8.1.

Example 10.7.1.

For the model problem using a five-point approximation the non-zero structure of the resulting matrix is given by

$$\mathcal{P}_{\mathcal{A}} = \{(i,j) \,|\, |i-j| = -n, -1, 0, 1, n\}.$$

Let us write $A = LU + R$, where

$$L = L_{-n} + L_{-1} + L_0, \qquad U = U_0 + U_1 + U_n,$$

where L_{-k} (and U_k) denote matrices with nonzero elements only in the k-th lower (upper) diagonal. By the rule for multiplication by diagonals (see Problem 6.1.6),

$$A_k B_l = C_{k+l}, \text{ if } k + l \leq n - 1,$$

we can form the product

$$\begin{aligned} LU = (L_{-n} + L_{-1} + L_0)(U_0 + U_1 + U_n) &= (L_{-n}U_n + L_{-1}U_1 + L_0U_0) \\ + L_{-n}U_0 + L_{-1}U_0 + L_0U_n + L_0U_1 + R, \end{aligned}$$

where $R = L_{-n}U_1 + L_{-1}U_n$. Hence the defect matrix R has nonzero elements only in two extra diagonals.

A family of preconditioners can be derived by different choices of the set $\mathcal{P}$. The simplest choice is to take $\mathcal{P}$ equal to the sparsity pattern of A. This is called a level 0 incomplete factorization. A level 1 incomplete factorization is obtained by using the union of $\mathcal{P}$ and the pattern of the defect matrix $R = LL^T - A$. Higher level incomplete factorizations are defined in a similar way, and so on.

An incomplete LU factorization may not exist even if A is nonsingular and has an LU factorization. However, for some more restricted classes of matrices the existence of incomplete factorizations can be guaranteed. The following result was proved by Meijerink and van der Vorst [276, 1977].

Theorem 10.7.2.

If A is an M-matrix, there exists for every set $\mathcal{P}$ such that $(i,j) \in \mathcal{P}$ for $i = j$, uniquely defined lower and upper triangular matrices L and U with $l_{ij} = 0$ or $u_{ij} = 0$ if $(i,j) \notin \mathcal{P}$, such that the splitting $A = LU - R$ is regular.

In case A is s.p.d., we define similarly an **incomplete Cholesky factorization**. Here the nonzero set $\mathcal{P}$ is assumed to be symmetric, i.e., if $(i,j) \in \mathcal{P}$ then also $(j,i) \in \mathcal{P}$. Positive definiteness of A alone is not sufficient to guarantee the existence of an incomplete Cholesky factorization. This is because zero elements may occur on the diagonal during the factorization.

For the case when A is a symmetric M-matrix, a variant of the above theorem guarantees the existence for each symmetric set $\mathcal{P}$ such that $(i,j) \in \mathcal{P}$ for $i = j$, a uniquely defined lower triangular matrix L, with $l_{ij} = 0$ if $(i,j) \notin \mathcal{P}$ such that the splitting $A = LL^T - R$ is regular.

A description of the incomplete Cholesky factorization in the general case is given below.

for $j = 1 : n$

$$l_{jj} = \Big(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\Big)^{1/2};$$

for $i = j+1, \ldots, n$

if $(i,j) \notin \mathcal{P}$ **then** $l_{ij} = 0$ **else**

$$l_{ij} = \Big(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\Big)$$

end

end

10.7.5 Block Incomplete Factorizations

Many matrices arising from the discretization of multidimensional problems have a block structure. For such matrices one can generalize the above idea and develop **block incomplete factorizations**. In particular, we consider here a symmetric positive definite block tridiagonal matrices of the form

$$A = \begin{pmatrix} D_1 & A_2^T & & & \\ A_2 & D_2 & A_3^T & & \\ & A_3 & \ddots & \ddots & \\ & & \ddots & \ddots & A_N^T \\ & & & A_N & D_N \end{pmatrix} = D_A - L_A - L_A^T, \qquad (10.7.9)$$

with square diagonal blocks D_i. For the model problem with the natural ordering of mesh points we obtain this form with $A_i = -I$, $D_i = \mathrm{tridiag}(-1 \;\; 4 \;\; -1)$. If systems with D_i can be solved efficiently a simple choice of preconditioner is the block diagonal preconditioner

$$M = \mathrm{diag}\,(D_1, D_2, \ldots, D_N).$$

The case $N = 2$ is of special interest. For the system

$$\begin{pmatrix} D_1 & A_2^T \\ A_2 & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \tag{10.7.10}$$

the **block diagonal preconditioner** gives a preconditioned matrix of the form

$$M^{-1}A = \begin{pmatrix} I & D_1^{-1}A_2^T \\ D_2^{-1}A_2 & I \end{pmatrix}.$$

Note that this matrix is of the form (10.2.5) and therefore has property A. Suppose the conjugate gradient method is used with this preconditioner and a starting approximation $x_1^{(0)}$. If we set

$$x_2^{(0)} = D_2^{-1}(b_2 - A_2 x_1^{(0)}),$$

then the corresponding residual $r_2^{(0)} = b_2 - D_2 x_1^{(0)} A_2 x_2^{(0)} = 0$. It can be shown that in the following steps of the conjugate gradient method we will alternately have

$$r_2^{(2k)} = 0, \qquad r_1^{(2k+1)} = 0, \qquad k = 0, 1, 2, \ldots.$$

This can be used to save about half the work.

If we eliminate x_1 in the system (10.7.10) then we obtain

$$Sx_2 = b_2 - A_2 D_1^{-1} b_1, \qquad S = D_2 - A_2 D_1^{-1} A_2^T, \tag{10.7.11}$$

where S is the Schur complement of D_1 in A. If A is s.p.d., then S is also s.p.d., and hence the conjugate gradient can be used to solve the system (10.7.11). This process is called **Schur complement preconditioning**. Here it is not necessary to form the Schur complement S, since we only need the effect of S on vectors. We can save some computations by writing the residual of the system (10.7.11) as

$$r_2 = (b_2 - D_2 x_2) - A_2 D_1^{-1}(b_1 - A_2^T x_2).$$

Note here that $x_1 = D_1^{-1}(b_1 - A_2^T x_2)$ is available as an intermediate result. The solution of the system $D_1 x_1 = b_1 - A_2^T x_2$ is cheap when, e.g., D_1 is tridiagonal. In other cases this system may be solved in each step by an iterative method in an **inner iteration**.

We now describe a **block incomplete Cholesky factorization** due to Concus, Golub and Meurant [73], which has proved to be very useful. We assume in the following that in (10.7.9) D_i is tridiagonal and A_i is diagonal, as in the model problem. First recall from Section 6.4.6 that the exact block Cholesky factorization of a symmetric positive definite block-tridiagonal matrix can be written as

$$A = (\Sigma + L_A)\Sigma^{-1}(\Sigma + L_A^T),$$

where L_A is the lower block triangular part of A, and $\Sigma = \operatorname{diag}(\Sigma_1, \ldots, \Sigma_n)$, is obtained from the recursion

$$\Sigma_1 = D_1, \qquad \Sigma_i = D_i - A_i \Sigma_{i-1}^{-1} A_i^T, \quad i = 2, \ldots, N.$$

For the model problem, although D_1 is tridiagonal, Σ^{-1} and hence Σ_i, $i \geq 2$, are dense. Because of this the exact block Cholesky factorization is not useful.

Instead we consider computing an incomplete block factorization from

$$\Delta_1 = D_1, \qquad \Delta_i = D_i - A_i \Lambda_{i-1}^{-1} A_i^T, \quad i = 2, \ldots, N. \tag{10.7.12}$$

Here, for each i, Λ_{i-1} is a sparse approximation to Δ_{i-1}. The incomplete block Cholesky factorization is then

$$M = (\Delta + L_A)\Delta^{-1}(\Delta + L_A^T), \qquad \Delta = \operatorname{diag}(\Delta_1, \ldots, \Delta_n).$$

The corresponding defect matrix is $R = M - A = \operatorname{diag}(R_1, \ldots, R_n)$, where $R_1 = \Delta_1 - D_1 = 0$,

$$R_i = \Delta_i - D_i - A_i \Delta_{i-1}^{-1} A_i^T, \qquad i = 2, \ldots, n.$$

We have assumed that the diagonal blocks D_i are diagonally dominant symmetric tridiagonal matrices. We now discuss the construction of an approximate inverse of such a matrix

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n, \end{pmatrix}$$

where $\alpha_i > 0$, $i = 1, \ldots, n$ and $\beta_i < 0$, $i = 1, \ldots, m-1$. A sparse approximation of D_i^{-1} can be obtained as follows. First compute the Cholesky factorization $T = LL^T$, where

$$L = \begin{pmatrix} \gamma_1 & & & & \\ \delta_1 & \gamma_2 & & & \\ & \delta_2 & \ddots & & \\ & & \ddots & \gamma_{n-1} & \\ & & & \delta_{n-1} & \gamma_n \end{pmatrix}.$$

It can be shown that the elements of the inverse $T^{-1} = L^{-T}L^{-1}$ decrease strictly away from the diagonal. This suggests that the matrix L^{-1}, which is lower triangular and dense, is approximated by a banded lower triangular matrix $L^{-1}(p)$, taking only the first $p + 1$ lower diagonals of the exact L^{-1}. Note that elements of the matrix L^{-1}

$$L^{-1} = \begin{pmatrix} 1/\gamma_1 & & & & \\ \zeta_1 & 1/\gamma_2 & & & \\ \eta_1 & \zeta_2 & \ddots & & \\ \vdots & \ddots & \ddots & 1/\gamma_{n-1} & \\ & \cdots & \eta_{n-2} & \zeta_{n-1} & 1/\gamma_n \end{pmatrix},$$

can be computed diagonal by diagonal. For example, we have

$$\zeta_i = \frac{\delta_i}{\gamma_i \gamma_{i+1}}, \qquad i = 2, \ldots, n-1.$$

For $p = 0$ we get a diagonal approximate inverse. For $p = 1$ the approximate Cholesky factor $L^{-1}(1)$ is lower bidiagonal, and the approximate inverse is a tridiagonal matrix. Since we have assumed that A_i are diagonal matrices, the approximations Δ_i generated by (10.7.12) will in this case be tridiagonal.

10.7.6 Fast Direct Methods

For the solution of discretizations of some elliptic problems on a rectangular domain fast direct methods can be developed. For this approach to be valid we needed to make strong assumptions about the regularity of the system. It applies only to discretizations of problems with constant coefficients on a rectangular domain. However, if we have variable coefficients the fast solver may be used as a preconditioner in the conjugate gradient method. Similarly, problems on an irregular domain may be embedded in a rectangular domain, and again we may use a preconditioner based on a fast solver.

Consider a linear system $Ax = b$ where A has the block-tridiagonal form

$$A = \begin{pmatrix} B & T & & & \\ T & B & T & & \\ & T & B & \ddots & \\ & & \ddots & \ddots & T \\ & & & T & B \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{pmatrix}, \tag{10.7.13}$$

where $B, T \in \mathbf{R}^{p\times p}$. We assume that the matrices B and T commute, i.e., $BT = TB$. Then it follows that B and T share a common system of eigenvectors, and we let

$$Q^T BQ = \Lambda = \operatorname{diag}(\lambda_i), \qquad Q^T TQ = \Omega = \operatorname{diag}(\omega_i).$$

The system $Ax = b$ can then be written $Cz = y$, where

$$C = \begin{pmatrix} \Lambda & \Omega & & & \\ \Omega & \Lambda & \Omega & & \\ & \Omega & \Lambda & \ddots & \\ & & \ddots & \ddots & \Omega \\ & & & \Omega & \Lambda \end{pmatrix},$$

and $x_j = Qz_j$, $y_j = Q^T b_j$, $j = 1, \dots, q$.

For the model problem with Dirichlet boundary conditions the eigenvalues and eigenvectors are known. Furthermore, the multiplication of a vector by Q and Q^T is efficiently obtained by the Fast Fourier Transform algorithm (see Section 9.6.3). One fast algorithm then is as follows:

1. Compute

$$y_j = Q^T b_j, \qquad j = 1, \dots, q.$$

2. Rearrange taking one element from each vectors y_j,

$$\hat{y}_i = (y_{i1}, y_{i2}, \dots, y_{iq})^T, \quad i = 1, \dots, p,$$

and solve by elimination the p systems

$$\Gamma_i \hat{z}_i = \hat{y}_i, \qquad i = 1, \ldots, p,$$

where

$$\Gamma_i = \begin{pmatrix} \lambda_i & \omega_i & & & \\ \omega_i & \lambda_i & \omega_i & & \\ & \omega_i & \lambda_i & \ddots & \\ & & \ddots & \ddots & \omega_i \\ & & & \omega_i & \lambda_i \end{pmatrix}, \quad i = 1, \ldots, p.$$

3. Rearrange (inversely to step 2) taking one element from each $\hat{z}_i$,

$$z_j = (\hat{z}_{j1}, \hat{z}_{j2}, \ldots, \hat{z}_{jq})^T, \quad j = 1, \ldots, q,$$

and compute

$$x_j = Q z_j, \qquad j = 1, \ldots, q.$$

The fast Fourier transforms in steps 1 and 3 takes $\mathcal{O}(n^2 \log n)$ operations. Solving the tridiagonal systems in step 2 only takes $\mathcal{O}(n^2)$ operations, and hence for this step Gaussian elimination is superior to FFT. In all this algorithm uses only $\mathcal{O}(n^2 \log n)$ operations to compute the n^2 unknown components of x.

Review Questions

1. What is meant by preconditioning of a linear system $Ax = b$. What are the properties that a good preconditioner should have?

2. What is meant by an incomplete factorization of a matrix? Describe how incomplete factorizations can be used as preconditioners.

3. Describe two different transformations of a general nonsymmetric linear system $Ax = b$, that allows the transformed system to be solved by the standard conjugate gradient method.

Problems and Computer Exercises

1. Consider square matrices of order n, with nonzero elements only in the k-th upper diagonal, i.e., of the form

$$T_k = \begin{pmatrix} & t_1 & & \\ & & \ddots & \\ & & & t_{n-k} \\ & & & \end{pmatrix}, \qquad k \geq 0.$$

Show the following rule for *multiplication by diagonals*:

$$A_k B_l = \begin{cases} C_{k+l}, & \text{if } k+l \le n-1; \\ 0, & \text{otherwise,} \end{cases}$$

where the elements in C_{k+l} are $a_1 b_{k+1}, \ldots, a_{n-k-l} b_{n-l}$.

2. Let B be a symmetric positive definite M-matrix of the form

$$B = \begin{pmatrix} B_1 & -C^T \\ -C & B_2 \end{pmatrix}$$

with B_1 and B_2 square. Show that the Schur complement $S = B_2 - CB_1^{-1}C^T$ of B_1 in B is a symmetric positive definite M-matrix.

3. The penta-diagonal matrix of the model problem has nonzero elements in positions $\mathcal{P}_{\mathcal{A}} = \{(i,j) \,|\, |i-j| = 0, 1, n\}$, which defines a level 0 incomplete factorization. Show that the level 1 incomplete factorization has two extra diagonals corresponding to $|i-j| = n-1$.

4. The triangular solves needed when using an incomplete Cholesky factorizations as a preconditioner are inherently sequential and difficult to vectorize. If the factors are normalized to be unit triangular, then the solution can be computed making use of one of the following expansions

$$(I-L)^{-1} = \begin{cases} I + L + L^2 + L^3 + \cdots & \text{(Neumann expansion)} \\ (I+L)(I+L^2)(I+L^4)\cdots & \text{(Euler expansion)} \end{cases}$$

Verify these expansions and prove that they are finite.

5. Let A be a symmetric positive definite matrix. An incomplete Cholesky preconditioner for A is obtained by neglecting elements in places (i,j) prescribed by a symmetric set

$$\mathcal{P} \subset \mathcal{P}_n \equiv \{(i,j) \,|\, 1 \le i, j \le n\},$$

where $(i,j) \in \mathcal{P}$, if $i = j$.
The simplest choice is to take $\mathcal{P}$ equal to the sparsity pattern of A, which for the model problem is $\mathcal{P}_A = \{(i,j) \,|\, |i-j| = 0, 1, n\}$. This is called a level 0 incomplete factorization. A level 1 incomplete factorization is obtained by using the union of $\mathcal{P}_0$ and the pattern of the defect matrix $R = LL^T - A$. Higher level incomplete factorizations are defined in a similar way.
(a) Consider the model problem, where A is block tridiagonal

$$A = \text{tridiag}(-I, T + 2I, -I) \in \mathbf{R}^{n^2 \times n^2}, \qquad T = \text{tridiag}(-1, 2-1) \in \mathbf{R}^{n \times n}.$$

Show that A is an M-matrix and hence that an incomplete Cholesky factorizations of A exists?
(b) Write a MATLAB function, which computes the level 0 incomplete Cholesky factor L_0 of A. (You should *NOT* write a general routine like that in the textbook, but an efficient routine using the special five diagonal structure of A!)

Implement also the preconditioned conjugate gradient method in MATLAB, and a function which solves $L_0 L_0^T z = r$ by forward and backward substitution. Solve the model problem for $n = 10$, and 20 with and without preconditioning, plot the error norm $\|x - x_k\|_2$, and compare the rate of convergence. Stop the iterations when the recursive residual is of the level of machine precision. Discuss your results!

(c) Take the exact solution to be $x = (1, 1, \ldots, 1, 1)^T$. To investigate the influence of the preconditioner $M = LL^T$ on the spectrum of $M^{-1}A$ do the following. For $n = 10$ plot the eigenvalues of A and of $M^{-1}A$ for level 0 and level 1 preconditioner. You may use, e.g., the built-in MATLAB functions to compute the eigenvalues, and efficiency is not a premium here. (To handle the level 1 preconditioner you need to generalize your incomplete Cholesky routine.)

Notes and References

Two classical texts on iterative methods as they were used in the 1960s and 1970s are Varga [375, 1962] and Young [396, 1971]. An excellent and authoritative survey of the development of iterative methods for solving linear systems during the last century is given by Saad and van der Vorst in [329, 2000]. This paper also contains references to many classical papers.

More recent monographs discussing preconditioners and Krylov space methods include those of Axelsson [12, 1994], Greenbaum [185, 1997], Saad [328, 2003] and H. van der Vorst [369, 2003]. Barret et al. [23, 1994] contain templates for implementation of many iterative methods. A pdf version of the second edition of this book can be downloaded from `http://www.csm.ornl.gov/ rbarret`. Implementation in MATLAB code is also available here.

[23, 1994] gives a compact survey of iterative methods and their implementation. (See also available software listed in Chapter 15.) The early history of the conjugate gradient and Lanczos algorithms are are detailed in [178, 1989] and Krylov solvers are studied by Broyden and Vespucci [52]. Domain decomposition methods are surveyed in Toselli and Widlund [357].

Section 10.2

The use of overrelaxation techniques was initiated by the work of David Young 1950, who introduced the notion of "consistent orderings"and "property A". These methods were used successively in the early 1960s to solve linear systems of order 20,000 with a Laplace-type matrix.

We remark that Kaczmarz's method has been rediscovered and used successfully in image reconstruction. In this context the method is known as the unconstrained ART algorithm (algebraic reconstruction technique). A survey of "row action methods" is given by Censor [57].

Section 10.4

Krylov subspace methods, which originated with the conjugate gradient method has been named one of the Top 10 Algorithms of the 20th century. The conjugate gradient method was developed independently by E. Stiefel and M. R. Hestenes.

Further work was done at the Institute for Numerical Analysis, on the campus of the University of California, in Los Angeles (UCLA). This work was published in 1952 in the seminal paper [206], which has had a tremendous impact in scientific computing. In this paper the author acknowledge cooperation with J. B. Rosser, G. E. Forsythe, and L. Paige, who were working at the institute during this period. They also mention that C. Lanczos had developed a closely related method (see Chapter 9).

Chapter 11

Nonlinear Systems and Optimization

11.1 Systems of Nonlinear Equations

11.1.1 Introduction

Many problems can be written in the generic form

$$f_i(x_1, x_2, \dots, x_n) = 0, \quad i = 1 : n. \tag{11.1.1}$$

where f_i are given functions of n variables. In this section we consider the numerical solution of such systems, where at least one function depends nonlinearly on at least one of the variables. Such a system is called a **nonlinear system** of equations, and can be written more compactly as

$$f(x) = 0, \qquad f : \ \mathbf{R}^n \to \mathbf{R}^n. \tag{11.1.2}$$

Even more generally, if f is an operator acting on some function space (11.1.1) is **functional equation**. Applications where nonlinear systems arise include initial and boundary value problems for nonlinear differential equations, and nonlinear integral equations.

The problem of finding *all* solutions of equation (11.1.1) in some subregion $\mathcal{B} \subset \mathbf{R}^n$ can be a very difficult problem. Note that in $\mathbf{R}^n$ there is no efficient method like the bisection method (see Chapter 5) that can be used as a global method to get initial approximations. In general we must therefore be content with finding local solutions, to which reasonable good initial approximations are known.

A nonlinear **optimization problem** is a problem of the form

$$\min_x \phi(x), \qquad x \in \mathbf{R}^n, \tag{11.1.3}$$

where the **objective function** ϕ is a nonlinear mapping $\mathbf{R}^n \to \mathbf{R}$. Most numerical methods try to find a **local minimum** of $\phi(x)$, i.e., a point x^* such that $\phi(x^*) \leq \phi(y)$ for all y in a neighborhood of x^*. If the objective function ϕ is continuously differentiable at a point x then any local minimum point x of ϕ must satisfy

$$g(x) = \nabla\phi(x) = 0, \tag{11.1.4}$$

where $g(x)$ is the gradient vector. This shows the close relationship between solving optimization problems and nonlinear systems of equations.

Optimization problems are encountered in many applications such as operations research, control theory, chemical engineering, and all kinds of curve fitting or more general mathematical model fitting. The optimization problem (11.1.3) is said to be **unconstrained**. In this chapter we consider mainly methods for unconstrained optimization. Methods for linear programming problems will be discussed in Section 11.4.

If in (11.1.1) there are $m > n$ equation we have an overdetermined nonlinear system. A least squares solution can then be defined to be a solution to

$$\min_{x\in\mathbf{R}^n} \phi(x), \qquad \phi(x) = \tfrac{1}{2}\|f(x)\|_2^2, \tag{11.1.5}$$

which is a **nonlinear least squares problem**. Note that this is an (unconstrained) optimization problem, where the objective function ϕ has a special form. Methods for this problem are described in Section 11.2.

Frequently the solution to the optimization problem (11.1.3) is restricted to lie in a region $\mathcal{B} \subset \mathbf{R^n}$. This region is often defined by inequality and equality constraints of the form

$$c_i(x) = 0, \quad i = 1 : m_1, \quad c_i(x) \geq 0, \quad i = m_1 + 1 : m. \tag{11.1.6}$$

There may also be constraints of the form $l_i \leq c_i(x) \leq u_i$. In the simplest case the constraint functions $c_i(x)$ are linear. Any point x, which satisfies the constraints, is said to be a **feasible point** and the set $\mathcal{B}$ is called the **feasible region**. An important special case is **linear programming** problems, where both $\phi(x)$ and the constraints (11.1.6) are linear. This problem has been extensively studied and very efficient methods exist for their solution; see Section 11.4

11.1.2 Fixed Point Iteration

In this section we generalize the theory of fixed-point iteration developed in Section 5.2 for a single nonlinear equation. Rewriting the system (11.1.1) in the form

$$x_i = g_i(x_1, x_2, \dots, x_n), \qquad i = 1 : n,$$

suggests an iterative method where, for $k = 0, 1, 2, \dots$, we compute

$$x_i^{(k+1)} = g_i\left(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}\right), \qquad i = 1 : n, \tag{11.1.7}$$

Using vector notations this can be written

$$x^{(k+1)} = g\bigl(x^{(k)}\bigr), \qquad k = 0, 1, 2, \dots, \tag{11.1.8}$$

which is known as a **fixed point iteration**. Clearly, if g is continuous and $\lim_{k\to\infty} x^{(k)} = x^*$, then $x^* = g(x^*)$ and x^* solves the system $x = g(x)$. (Recall that a vector sequence is said to converge to a limit x^* if $\lim_{k\to\infty} \|x^{(k)} - x^*\| = 0$ for some norm $\|\cdot\|$, see Section 6.2.5).

Example 11.1.1.

The nonlinear system

$$\begin{aligned} x^2 - 2x - y + 1 &= 0 \\ x^2 + y^2 - 1 &= 0 \end{aligned}$$

defines the intersection between a circle and a parabola. The two real roots are $(1,0)$ and $(0,1)$. Taking $x_0 = 0.9$ and $y_0 = 0.2$ and using the following fixed point iteration

$$x_{k+1} = (y_k - 1)/(x_k - 2), \qquad y_{k+1} = 1 - x_k^2/(y_k + 1),$$

we obtain the results

k	x_k	y_k
1	0.72727273	0.32500000
2	0.53035714	0.60081085
3	0.27162323	0.82428986
4	0.10166194	0.95955731
5	0.02130426	0.99472577
6	0.00266550	0.99977246
7	0.00011392	0.99999645
8	0.00000178	0.99999999

Note that although we started close to the root $(1,0)$ the sequence converges to the other real root $(0,1)$. (See also Problem 1.)

We will now derive sufficient conditions for the convergence of the fixed point iteration (11.1.8). We first need a definition.

Definition 11.1.1.

A function $f(x)$: $\mathbf{R}^n \to \mathbf{R}^n$, is said to **Lipschitz continuous** *in an open set $D \in \mathbf{R}^n$ if there exists a constant L such that*

$$\|f(x) - f(y)\| \le L\|x - y\|, \quad \forall x, y \in D.$$

The constant L is called a **Lipschitz constant***. If $L < 1$ then f is called a* **contraction***.*

The following important theorem generalizes Theorem 5.2.2. It not only provides a solid basis for iterative numerical techniques, but also is an important tool in theoretical analysis. Note that, *the existence of a fixed point is not assumed a priori.*

Theorem 11.1.2. The Contraction Mapping Theorem.

Let $T :\ E \to F$, where $E = F = \mathbf{R}^n$, be an iteration function, and $S_r = \{u \mid \|u - u_0\| < r\}$ be a ball of radius r around a given starting point $u_0 \in \mathbf{R^n}$. Assume that T is a contraction mapping in S_r, i.e.,

$$u, v \in S_r \Rightarrow \|T(u) - T(v)\| \le L\|u - v\|, \tag{11.1.9}$$

where $L < 1$. Then if

$$\|u_0 - T(u_0)\| \le (1-L)r \tag{11.1.10}$$

the equation $u = T(u)$ has a unique solution u^* in the closure $\overline{S_r} = \{u \mid \|u - u_0\| \le r\}$. This solution can be obtained by the convergent iteration process $u_{k+1} = T(u_k)$, $k = 0, 1, \ldots$, and we have the error estimate

$$\|u_k - u^*\| \le \|u_k - u_{k-1}\| \frac{L}{1-L} \le \|u_1 - u_0\| \frac{L^k}{1-L}. \tag{11.1.11}$$

Proof. We first prove the uniqueness. If there were two solutions u' and u'', we would get $u' - u'' = T(u') - T(u'')$ so that

$$\|u' - u''\| = \|T(u') - T(u'')\| \le L\|u' - u''\|.$$

Since $L < 1$, it follows that $\|u' - u''\| = 0$, i.e., $u' = u''$.

By (11.1.10) we have $\|u_1 - u_0\| = \|T(u_0) - u_0\| \le (1-L)r$, and hence $u_1 \in S_r$. We now use induction to prove that $u_n \in S_r$ for $n < j$, and that

$$\|u_j - u_{j-1}\| \le L^{j-1}(1-L)r, \qquad \|u_j - u_0\| \le (1 - L^j)r.$$

We already know that these estimates are true for $j = 1$. Using the triangle inequality and (11.1.9) we get

$$\begin{aligned}
\|u_{j+1} - u_j\| &= \|T(u_j) - T(u_{j-1})\| \le L\|u_j - u_{j-1}\| \le L^j(1-L)r, \\
\|u_{j+1} - u_0\| &\le \|u_{j+1} - u_j\| + \|u_j - u_0\| \le L^j(1-L)r + (1-L^j)r \\
&= (1 - L^{j+1})r.
\end{aligned}$$

This proves the induction step, and it follows that the sequence $\{u_k\}_{k=0}^{\infty}$ stays in S_r. We also have for $p > 0$

$$\begin{aligned}
\|u_{j+p} - u_j\| &\le \|u_{j+p} - u_{j+p-1}\| + \cdots + \|u_{j+1} - u_j\| \\
&\le (L^{j+p-1} + \ldots + L^j)(1-L)r \le L^j(1-L^p)r \le L^j r,
\end{aligned}$$

and hence $\lim_{j\to\infty} \|u_{j+p} - u_j\| = 0$. The sequence $\{u_k\}_{k=0}^{\infty}$ therefore is a Cauchy sequence, and since $\mathbf{R^n}$ is complete has a limit u^*. Since $u_j \in S_r$ for all j it follows that $u^* \in \overline{S_r}$.

Finally, by (11.1.9) T is continuous, and it follows that $\lim_{k\to\infty} T(u_k) = T(u^*) = u^*$. The demonstration of the error estimates (11.1.11) is left as exercises to the reader. □

Theorem 11.1.2 holds also in a more general setting, where $T: S_r \to \mathcal{B}$, and $\mathcal{B}$ is a Banach space[53] The proof goes through with obvious modifications. In this form the theorem can be used, e.g., to prove existence and uniqueness for initial value problems for ordinary differential equations, see Section 13.2.1.

[53] A Banach space is a normed vector space which is complete, i.e., every Cauchy sequence converges to a point in $\mathcal{B}$, see Dieudeonné [106, 1961].

The Lipschitz constant L is a measure of the rate of convergence; at every iteration the upper bound for the norm of the error is multiplied by a factor equal to L. The existence of a Lipschitz condition is somewhat more general than a differentiability condition, which we now consider.

Definition 11.1.3.

The function $f_i(x)$, $\mathbf{R}^n \to \mathbf{R}$, is said to be continuously differentiable at a point x if the **gradient** *vector*

$$\nabla\phi(x) = \Big(\frac{\partial f_i}{\partial x_1}, \dots, \frac{\partial f_i}{\partial x_n}\Big)^T \in \mathbf{R}^n \tag{11.1.12}$$

exists and is continuous. The vector valued function $f(x)$, $\mathbf{R}^n \to \mathbf{R}^n$, is said to be differentiable at the point x if each component $f_i(x)$ is differentiable at x. The matrix

$$J(x) = \begin{pmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_n(x)^T \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \in \mathbf{R}^{n\times n}, \tag{11.1.13}$$

is called the **Jacobian** *of f.*

The following theorem shows how a Lipschitz constant for $f(x)$ can be expressed in terms of the derivative $f'(x)$.

Lemma 11.1.4.

Let function $f(x)$, $\mathbf{R}^n \to \mathbf{R}^n$, be differentiable in a convex set $\mathcal{D} \subset \mathbf{R}^n$. Then $L = \max_{y\in\mathcal{D}} \|f'(y)\|$ is a Lipschitz constant for f.

Proof. Let $0 \le t \le 1$ and consider the function $g(t) = f(a + t(x - a))$, $a, x \in \mathcal{D}$. By the chain rule $g'(t) = f'\big(a + t(x - a)\big)(x - a)$ and

$$f(x) - f(a) = g(1) - g(0) = \int_0^1 g'(t)dt = \int_0^1 f'\big(a + t(x - a)\big)(x - a)dt.$$

Since $\mathcal{D}$ is convex the whole line segment between the points a and x belongs to $\mathcal{D}$. Applying the triangle inequality (remember that an integral is the limit of a sum) we obtain

$$\|f(x) - f(a)\| < \int_0^1 \|f'(a + t(x - a))\| \; \|x - a\|dt \le \max_{y\in\mathcal{D}} \|f'(y)\| \; \|x - a\|.$$

□

11.1.3 Newton-Type Methods

Newton's method for solving a single nonlinear equation $f(x) = 0$ can be derived by using Taylor's formula to get a linear approximation for f at a point. To get a quadratically convergent method for a system of nonlinear equations we must similarly use derivative information of $f(x)$.

Let x_k be the current approximation[54] and assume that $f_i(x)$ is twice differentiable at x_k. Then by Taylor's formula

$$f_i(x) = f_i(x_k) + (\nabla f_i(x_k))^T(x - x_k) + O(\|x - x_k\|^2), \qquad i = 1 : n.$$

Using the Jacobian matrix (11.1.13) the nonlinear system $f(x) = 0$ can be written

$$f(x) = f(x_k) + J(x_k)(x - x_k) + O(\|x - x_k\|^2) = 0.$$

Neglecting higher order terms we get the linear system

$$J(x_k)(x - x_k) = -f(x_k). \tag{11.1.14}$$

If $J(x_k)$ is nonsingular then (11.1.14) has a unique solution x_{k+1}, which can be expected to be a better approximation. The resulting iterative algorithm can be written

$$x_{k+1} = x_k - \big(J(x_k)\big)^{-1} f(x_k). \tag{11.1.15}$$

which is **Newton's method**. Note that, in general the inverse Jacobian matrix should not be computed. Instead the linear system (11.1.14) is solved, e.g., by Gaussian elimination. If n is very large and $J(x_k)$ sparse it may be preferable to use one of the iterative methods given in Chapter 11. Note that in this case x_k can be used as an initial approximation.

The following example illustrates the quadratic convergence of Newton's method for simple roots.

Example 11.1.2.

The nonlinear system

$$\begin{aligned} x^2 + y^2 - 4x &= 0 \\ y^2 + 2x - 2 &= 0 \end{aligned}$$

has a solution close to $x_0 = 0.5$, $y_0 = 1$. The Jacobian matrix is

$$J(x, y) = \begin{pmatrix} 2x - 4 & 2y \\ 2 & 2y \end{pmatrix},$$

and Newton's method becomes

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - J(x_k, y_k)^{-1} \begin{pmatrix} x_k^2 + y_k^2 - 4x_n \\ y_k^2 + 2x_k - 2 \end{pmatrix}.$$

We get the results:

k	x_k	y_k
1	0.35	1.15
2	0.35424528301887	1.13652584085316
3	0.35424868893322	1.13644297217273
4	0.35424868893541	1.13644296914943

[54] In the following we use vector notations so that x_k will denote the kth approximation and not the kth component of x.

All digits are correct in the last iteration. The quadratic convergence is obvious; the number of correct digits approximately doubles in each iteration.

It is useful to have a precise measure of the asymptotic rate of convergence for a vector sequence converging to a limit point.

Definition 11.1.5.
A convergent sequence $\{x_k\}$ *with* $\lim_{k\to\infty}\{x_k\} = x^*$*, and* $x_k \neq x^*$*, is said to have* **order of convergence** *equal to* p $(p \geq 1)$*, if*

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} = C, \tag{11.1.16}$$

where $|C| < 1$ *for* $p = 1$ *and* $|C| < \infty$*,* $p > 1$*.* C *is called the* **asymptotic error constant***. The sequence has exact convergence order* p *if (11.1.16) holds with* $C \neq 0$*. We say the convergence is* **superlinear** *if* $C = 0$ *for some* $p \geq 1$*.*

Note that for finite dimensional vector sequences, the order of convergence p does not depend on the choice of norm, and that the definitions agree with those introduced for scalar sequences, see Def. 5.2.1. (More detailed discussions of convergence rates is found in Dennis and Schnabel [101, pp. 19–21], and Chapter 9 of Ortega and Rheinboldt [293].)

In order to analyze the convergence of Newton's method we need to study how well the linear model (11.1.14) approximates the equation $f(x) = 0$. The result we need is given in the lemma below.

Lemma 11.1.6.
Assume that the Jacobian matrix satisfies the Lipschitz condition

$$\|J(x) - J(y)\| \leq \gamma\|x - y\|, \qquad \forall x, y \in \mathcal{D},$$

where $\mathcal{D} \subset \mathbf{R}^n$ *is a convex set. Then for all* $x, y \in \mathcal{D}$ *it holds that*

$$\|f(x) - f(y) - J(y)(x - y)\| \leq \frac{\gamma}{2}\|x - y\|^2.$$

Proof. The function $g(t) = f(y + t(x - y))$, $x, y \in \mathcal{D}$ is differentiable for all $0 \leq t \leq 1$, and by the chain rule $g'(t) = J(y + t(x - y))(x - y)$. It follows that

$$\|g'(t) - g'(0)\| = \|\big(J(y + t(x - y)) - J(y)\big)(x - y)\| \leq \gamma t\|x - y\|^2. \tag{11.1.17}$$

Since the line segment between x and y belongs to $\mathcal{D}$

$$f(x) - f(y) - J(y)(x - y) = g(1) - g(0) - g'(0) = \int_0^1 \big(g'(t) - g'(0)\big)dt.$$

Taking norms and using (11.1.17) it follows that

$$\|f(x) - f(y) - J(y)(x - y)\| \leq \int_0^1 \|g'(t) - g'(0)\|dt \leq \gamma\|x - y\|^2 \int_0^1 t dt.$$

□

The following famous theorem gives rigorous conditions for the quadratic convergence of Newton's method. It also shows that Newton's method in general converges provided that x_0 is chosen sufficiently close to a solution x^*.

Theorem 11.1.7. (Newton–Kantorovich Theorem)

Let $f : \mathbf{R}^n \to \mathbf{R}^n$ be continuously differentiable in an open convex set $C \in \mathbf{R}^n$, and Let the Jacobean matrix of $f(x)$ be $J(x)$. Assume that $f(x^*) = 0$, for $x^* \in \mathbf{R}^n$. Let positive constants r, $\beta > 0$ be given such that $S_r(x^*) = \{x| \, \|x - x^*\| < r\} \subseteq C$, and

(a) $\quad \|J(x) - J(y)\| \le \gamma\|x - y\|, \qquad \forall \; x, y \in S_r(x^*)$,

(b) $\quad J(x^*)^{-1}$ *exists and satisfies* $\|J(x^*)^{-1}\| \le \beta$.

Then there exists an $\epsilon > 0$ such that for all $x_0 \in S_\epsilon(x^*)$ the sequence generated by

$$x_{k+1} = x_k - J\big(x_k\big)^{-1} f(x_k), \quad k = 0, 1, \dots$$

is well defined, $\lim_{n\to\infty} = x^*$, *and satisfies*

$$\|x_{k+1} - x^*\| \le \beta\gamma\|x_k - x^*\|^2.$$

Proof. We choose $\epsilon = \min\{r, 1/(2\beta\gamma)\}$. Then by (a) and (b) it follows that

$$\|J(x^*)^{-1}(J(x_0) - J(x^*))\| \le \beta\gamma\|x_0 - x^*\| \le \beta\gamma\epsilon \le 1/2.$$

By Corollary 6.6.1 and (b) we have $\|J(x_0)^{-1}\| \le \|J(x^*)^{-1}\|/(1 - 1/2) = 2\beta$. It follows that x_1 is well defined and

$$\begin{aligned} x_1 - x^* &= x_0 - x^* - J(x_0)^{-1}\big(f(x_0) - f(x^*)\big) \\ &= J(x_0)^{-1}\left(f(x^*) - f(x_0) - J(x_0)(x^* - x_0)\right). \end{aligned}$$

Taking norms we get

$$\begin{aligned} \|x_1 - x^*\| &\le \|J\big(x_0\big)^{-1}\| \, \|f(x^*) - f(x_0) - J(x_0)(x^* - x_0)\| \\ &\le 2\beta\gamma/2\|x_0 - x^*\|^2, \end{aligned}$$

which proves quadratic convergence. □

We remark that a result by Kantorovich shows quadratic convergence under weaker conditions. In particular, it is not necessary to assume the existence of a solution , or the nonsingularity of $J(x)$ at the solution. For a discussion and proof of these results we refer to Ortega and Rheinboldt [293, 1970, Ch. 12.6].

Each step of Newton's method requires the evaluation of the n^2 entries of the Jacobian matrix $J(x_k)$, and to solve the resulting linear system $n^3/3$ arithmetic operations are needed. This may be a time consuming task if n is large. In many situations it might be preferable to reevaluate $J(x_k)$ only occasionally using the same Jacobian in $m > 1$ steps,

$$J(x_p)(x_{k+1} - x_k) = -f(x_k), \quad k = p : p + m - 1. \tag{11.1.18}$$

Once we have computed the LU factorization of $J(x_p)$ the linear system can be solved in n^2 arithmetic operations. The motivation for this approach is that if either the iterates or the Jacobian matrix are not changing too rapidly $J(x_p)$ is a good approximation to $J(x_k)$. (These assumptions do not usually hold far away from the solution, and may cause divergence in cases where the unmodified algorithm converges.)

The modified Newton method can be written as a fixed point iteration with

$$g(x) = x - J(x_p)^{-1} f(x), \quad g'(x) = I - J(x_p)^{-1} J(x).$$

We have, using assumptions from Theorem 11.1.7

$$\|g'(x)\| \le \|J(x_p)^{-1}\| \, \|J(x_p) - J(x)\| \le \beta\gamma \|x_p - x\|.$$

Since $g' \neq 0$ the modified Newton method will only have *linear* rate of convergence. Also, far away from the solution the modified method may diverge in cases where Newton's method converges.

Example 11.1.3.

Consider the nonlinear system in Example 11.1.2. Using the modified Newton method with fixed Jacobian matrix evaluated at $x_1 = 0.35$ and $y_1 = 1.15$

$$J(x_1, y_1) = \begin{pmatrix} 2x_1 - 4 & 2y_1 \\ 2 & 2y_1 \end{pmatrix} = \begin{pmatrix} -3.3 & 2.3 \\ 2.0 & 2.3 \end{pmatrix}.$$

we obtain the result

k	x_k	y_k
1	0.35	1.15
2	0.35424528301887	1.13652584085316
3	0.35424868347696	1.13644394786146
4	0.35424868892666	1.13644298069439
5	0.35424868893540	1.13644296928555
6	0.35424868893541	1.13644296915104

11.1.4 Numerical Differentiation

It has been stated at several places in this book that numerical differentiation should be avoided, when the function values are subject to irregular errors, like errors of measurement or rounding errors. Nowadays, when a typical value of the machine constant $\mathbf{u}$ is $2^{-53} \approx 10^{-16}$, the harmful effect of *rounding errors* in the context of numerical differentiation, however, should not be exaggerated. We shall see that the accuracy of the first and second derivatives is satisfactory for most purposes, *if the step size is chosen appropriately.*

With the multilinear mapping formalism, the general case of vector valued dependent and independent variables becomes almost as simple as the scalar case. Let η be a small positive number. By Taylor's formula,

$$g'(x_0)v = \frac{g(x_0+\eta v) - g(x_0-\eta v)}{2\eta} + R_T, \quad R_T \approx -\frac{\eta^2 g'''(x_0)v^3}{6}, \tag{11.1.19}$$

where, as above, we use v^3 as an abbreviation for the list (v, v, v) of vector arguments. The Jacobian $g'(x_0)$ is obtained by the application of this to $v = e_j$, $j = 1 : k$. If the Jacobian has a band structure, then it can be computed by means of fewer vectors v; see Problem 3. First note that, *if g is quadratic, there is no truncation error, and η can be chosen rather large, so the rounding error causes no trouble either.*

Suppose that the rounding error of $g(x)$ is (approximately) bounded by $\epsilon\|g\|/\eta$. (The norms here are defined on a neighborhood of x_0.) The total error is therefore (approximately) bounded by

$$B(\eta) = \|g\|\frac{\epsilon}{\eta} + \|g'''\|\|v\|^3\frac{\eta^2}{6}.$$

Set $\|g\|/\|g'''\| = \xi^3$, and note that ξ measures a local length scale of the variation of the function g, (if we interpret x as a length). A good choice of η is found by straightforward optimization:

$$\min_{\eta} B(\eta) = (3\epsilon)^{2/3}\|g\|\|v\|/(2\xi), \quad \eta\|v\| = (3\epsilon)^{1/3}\xi. \tag{11.1.20}$$

For $\epsilon = 10^{-16}$, we should choose $\eta\|v\| = 7\cdot 10^{-6}\xi$. The error estimate becomes $2.5\cdot 10^{-11}\|g\|\|v\|/\xi$. In many applications this accuracy is higher than necessary. If uncentered differences are used instead of centered differences, the error becomes $O(\epsilon^{1/2})$ with optimal choice of η, while the amount of computation may be reduced by almost 50%; see Problem 1.

It may be a little cumbersome to estimate ξ by its definition, but since we need a very rough estimate only, we can replace it by some simpler measure of the length scale of $g(x)$, e.g. a rough estimate of (say) $\frac{1}{2}\|g\|/\|g'\|$.[55] Then the error estimate simplifies to $(3\epsilon)^{2/3}\|g'\|\|v\| \approx 5\cdot 10^{-11}\|g'\|\|v\|$ for $\epsilon = 10^{-16}$. This is usually an overestimate, though not always. Recall that if g is quadratic, there is no truncation error.

The result of a similar study of the directional *second derivative* reads

$$f''(x_0)v^2 = \frac{f(x_0+\eta v) - 2f(x_0) + f(x_0-\eta v)}{\eta^2} + R_T, \tag{11.1.21}$$

[55] The factor $\frac{1}{2}$ is a safety factor. So is the factor $\frac{1}{3}$ in the equation for ξ in the group (11.1.21).

$$
\begin{aligned}
R_T &\approx -\frac{\eta^2 f^{iv}(x_0)v^4}{12}, \\
B(\eta) &= \|f\|\frac{4\epsilon}{\eta^2} + \frac{\|f^{iv}\|\,\|v\|^4\eta^2}{12}, \\
\xi &= (\|f\|/\|f^{iv}|)^{1/4} \approx (\frac{1}{3}\|f\|/\|f''\|)^{1/2}, \\
\min_\eta B(\eta) &= 2(\epsilon/3)^{1/2}\|f\|\,\|v\|^2/\xi^2 \approx \epsilon^{1/2}\|f''\|\|v\|^2, \quad \eta\|v\| = (48\epsilon)^{1/4}\xi.
\end{aligned}
$$

Note that:

- if g is a cubic function, there is no truncation error, and $\eta\|v\|$ can be chosen independent of ϵ. Otherwise, for $\epsilon = 10^{-16}$, we should choose $\eta\|v\| \approx 3{\cdot}10^{-4}\xi$. The simplified error estimate becomes $2\cdot 10^{-8}\|f''\|\|v\|^2$;
- if $f'(x)$ is available, we can obtain $f''(x_0)v^2$ more accurately by setting $g(x) = f'(x)v$ into (11.1.19), since the value of η can then usually be chosen smaller;
- if $f(x)$ is a quadratic form, then $f''(x)$ is a constant bilinear operator and $f''(x)v^2 = f(v)$. If f is a non-homogeneous quadratic function, its affine part must be subtracted from the right hand side;
- in order to compute $f''(x_0)(u,v)$, it is sufficient to have a subroutine for $f''(x_0)v^2$, since the following formula can be used. It is easily derived by the bilinearity and symmetry of $f''(x_0)$.

$$
f''(x_0)(u,v) = \frac{1}{4}\big(f''(x_0)(u+v)^2 - f''(x_0)(u-v)^2\big) \tag{11.1.22}
$$

11.1.5 Derivative Free Methods

In many applications the Jacobian matrix is not available or too expensive to evaluate. Then we can use the **discretized Newton method**, where each of the derivative elements in $J(x_k)$ is discretized separately by a difference quotient. There are many different variations depending on the choice of discretization. A frequently used approximation for the jth column of $J(x)$ is the forward difference quotient

$$
\frac{\partial f(x)}{\partial x_j} \approx \Delta_j f(x) \equiv \frac{f(x + h_j e_j) - f(x)}{h_j}, \quad j = 1:n,
$$

where e_j denotes the jth unit vector and $h_j > 0$ is a suitable scalar. If the resulting approximation is denoted by $J(x, D)$, then we can write

$$
J(x,D) = \big(f(x + h_1 e_1) - f(x), \ldots, f(x + h_n e_n) - f(x)\big)D^{-1},
$$

where $D = \mathrm{diag}(h_1, h_2, \ldots, h_n)$ is a nonsingular diagonal matrix. This shows that $J(x_k, d)$ is nonsingular if and only if the vectors

$$
f(x_k + h_j e_j) - f(x_k), \quad j = 1:n,
$$

are linearly independent.

It is important that the step sizes h_j are chosen carefully. If h_j is chosen too large then the derivative approximation will have a large truncation error; if it is chosen too small then roundoff errors may be dominate (cf. numerical differentiation). As a rule of thumb one should choose h_j so that $f(x)$ and $f(x + h_j e_j)$ have roughly the first half digits in common, i.e.,

$$|h_j| \, \|\Delta_j f(x)\| \approx u^{1/2} \|f(x)\|.$$

In the discretized Newton method the vector function $f(x)$ needs to be evaluated at $n + 1$ points, including the point x_k. Hence it requires $n^2 + n$ component function evaluations per iteration. Methods which only only require $(n^2 + 3n)/2$ component function evaluations have been proposed by Brown (1966) and Brent (1973). Brent's method requires the computation of difference quotients

$$\frac{f(x + h_j q_k) - f(x)}{h_j}, \quad j = 1 : n, \quad k = j : n,$$

where $Q = (q_1, \dots, q_n)$ is a certain orthogonal matrix determined by the method. Note that because of common subexpressions, in some applications a component function evaluation may be almost as expensive as a vector function evaluation. In such cases the original Newton method is still to be preferred. For a discussion of these methods see Moré and Cosnard [284, 1979].

If in equation () we take $h_j = f_j(x_k)$, then we get a generalization of **Steffensen's method**; see Volume I, Sec. 6.2.3. In this method we have to evaluate $f(x)$ at the $(n + 1)$ points x_k and

$$x_k + f_j(x_k) e_j, \quad j = 1 : n.$$

Thus when $n > 1$ the number of function evaluations for Steffensen's method is the same as for the discretized Newton's method, but the order of convergence equals 2. A generalization of Steffensen's method to nonlinear operator in Banach space is given in [227]. There conditions are given for convergence, uniqueness and the existence of a fixed point.

11.1.6 Quasi-Newton Methods

See Jerry Eriksson [124].

If the function $f(x)$ is complicated to evaluate even the above method may be too expensive. In the methods above we obtain the next approximation x_{k+1} by a step along the direction h_k, computed by solving the linear system

$$B_k h_k = -f(x_k), \tag{11.1.23}$$

where B_k is an approximation to the Jacobian $J(x_k)$. The class of **quasi-Newton methods** can be viewed as a generalization of the secant method to functions of more than one variable. The approximate Jacobian B_{k+1} is required to satisfy the **secant equation**

$$B_{k+1} s_k = y_k \tag{11.1.24}$$

where s_k and y_k are the vectors

$$s_k = x_{k+1} - x_k, \qquad y_k = f(x_{k+1}) - f(x_k).$$

This means that B_{k+1} correctly imitates the Jacobian along the direction of change s_k. Of course many matrices satisfy this condition.

In the very successful **Broyden's method** it is further required that the difference $B_{k+1} - B_k$ has minimal Frobenius norm. It is left as an exercise to verify that these conditions lead to

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T}{s_k^T s_k}. \tag{11.1.25}$$

This is generally referred to as Broyden's "good" updating formula. Note that $B_{k+1} - B_k$ is *a matrix of rank one*, and that $B_{k+1}p = B_k p$ for all vectors p such that $p^T(x_{k+1} - x_k) = 0$. (To generate an initial approximation B_1 we can use finite differences along the coordinate directions.)

It can be shown that Broyden's modification of Newton's method has superlinear convergence.

Theorem 11.1.8.

Let $f(x) = 0$, $f:\ \mathbf{R}^n \to \mathbf{R}^n$, be sufficiently smooth, and let x^* be a regular zero point of f. Let

$$x_{k+1} = x_k - B_k^{-1} f(x_k)$$

be the Newton type method where B_k is updated according to Broyden's formula (11.1.25). If x_0 is sufficiently close to x^*, and B_0 sufficiently close to $f'(x_0)$, then the sequence $\{x_k\}$ is defined and converges superlinearly to x^*, i.e.,

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \to 0, \quad n \to \infty.$$

Proof. See Dennis and Moré [100]. □

We can compute B_{k+1} from (11.1.25) in only $2n^2$ operations and no extra function evaluations. To solve (11.1.23) for the Newton direction still seems to require $O(n^3)$ operations. However, assume that a QR decomposition $B_k = Q_k R_k$ was computed in the previous step. Then we can write

$$B_{k+1} = Q_k(R_k + u_k v_k^T), \quad u_k = Q_k^T(y_k - B_k s_k), \quad v_k^T = s_k^T / s_k^T s_k.$$

We will show below that the QR decomposition of $R_k + u_k v_k^T = \bar{Q}_k \bar{R}_{k+1}$ can be computed in $O(n^2)$ operation. Then we have

$$B_{k+1} = Q_{k+1} R_{k+1}, \qquad Q_{k+1} = Q_k \bar{Q}_k.$$

We start by determining a sequence of Givens rotations $G_{j,j+1}$, $j = n-1, \dots, 1$ such that

$$G_{1,2}^T \dots G_{n-1,n}^T u_k = \alpha e_1, \qquad \alpha = \pm \|u_k\|_2.$$

Note that these transformations zero the last $n-1$ components of u_k from bottom up. (For details on how to compute $G_{j,j+1}$ see Section 7.4.2.) The same transformations are now applied to the R_k, and we form

$$\bar{H} = G_{1,2}^T \dots G_{n-1,n}^T (R_k + u_k v_k^T) = H + \alpha e_1 v_k^T.$$

It is easily verified that in the product $H = G_{1,2}^T \dots G_{n-1,n}^T R_k$ the Givens rotations will introduce extra nonzero elements only in positions $(j, j+1)$, $j = 1 : n$, so that H becomes an upper Hessenberg matrix of the form

$$H = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}, \quad n = 4.$$

The addition of $\alpha e_1 v_k^T$ only modifies the first row of H, and hence also $\bar{H}$ is an upper Hessenberg matrix. We now determine a sequence of Givens rotations $\bar{G}_{j,j+1}$ so that $\bar{G}_{j,j+1}$ zeros the element $\bar{h}_{j+1,j}$, $j = 1 : n-1$. Then

$$\bar{G}_{n-1,n}^T \dots \bar{G}_{1,2}^T \bar{H} = \bar{R}_{k+1}$$

is the updated triangular factor. The orthogonal factor equals the product

$$\bar{Q}_k = G_{n-1,n} \dots G_{1,2} \bar{G}_{1,2} \dots \bar{G}_{n-1,n}.$$

The work needed for this update is as follows: Computing u_k takes n^2 flops. Computing $\bar{H}$ and R_k takes $4n^2$ flops and accumulating the product of $G_{j,j+1}$ and $\bar{G}_{j,j+1}$ takes $8n^2$ flops, for a total of $13n^2$ flops. It is possible to do a similar cheap update of the LU decomposition, but this may lead to stability problems.

If the Jacobian $f'(x)$ is sparse the advantages of Broyden's method is lost, since the update in general is not sparse. One possibility is then to keep the LU factors of the most recently computed sparse Jacobian and save several Broyden updates as pairs of vectors $y_k - B_k s_k$ and s_k.

11.1.7 Modifications for Global Convergence

We showed above that under certain regularity assumptions Newton's method is convergent from a sufficiently good initial approximation, i.e., under these assumptions Newton's method is **locally convergent**. However, Newton's method is not in general **globally convergent**, i.e., it does not converge from an arbitrary starting point. Far away from the root Newton's method may not behave well, e.g., it is not uncommon that the Jacobian matrix is illconditioned or even singular. This is a serious drawback since it is *much more difficult to find a good starting point in* $\mathbf{R}^n$ *than in* $\mathbf{R}$!

We now discuss techniques to modify Newton's method, which attempt to ensure **global convergence**, i.e., convergence from a large set of starting approximations. As mentioned in the introduction the solution of the nonlinear system $f(x)$ also solves the minimization problem

$$\min_x \phi(x), \qquad \phi(x) = \frac{1}{2}\|f(x)\|_2^2 = \frac{1}{2} f(x)^T f(x). \tag{11.1.26}$$

We seek modifications which will make $\|f(x)\|_2^2$ decrease at each step. We call d a **descent direction** for $\phi(x)$ if $\phi(x + \alpha d) < \phi(x)$, for all sufficiently small $\alpha > 0$. This will be the case if if the directional derivative is negative, i.e.

$$\nabla\phi(x)^T d = f(x)^T J(x)d < 0.$$

The steepest-descent direction

$$-g = -\nabla\phi(x) = -J(x)^T f(x)$$

is the direction in which $\phi(x)$ decreases most rapidly, see Section 11.3.2.

Assuming that $J(x)$ is nonsingular, the Newton direction $h = -J(x)^{-1}f(x)$ is also a descent direction if $f(x) \neq 0$ since

$$\nabla\phi(x)^T h = -f(x)^T J(x)J(x)^{-1}f(x) = -\|f(x)\|_2^2 < 0.$$

In the **damped Newton** method we take

$$x_{k+1} = x_k + \alpha_k h_k, \qquad J(x_k)h_k = -f(x_k). \tag{11.1.27}$$

where the step length α_k is computed by a **line search**. Ideally α_k should be chosen to be minimize the scalar function

$$\psi(\alpha) = \|f(x_k + \alpha h_k)\|_2^2.$$

Algorithms for solving such an one-dimensional minimization are discussed in Section 6.7. In practice this problem need not be solved accurately. It is only necessary to ensure that the reduction $\|f(x_k)\|_2^2 - \|f(x_{k+1})\|_2^2$ is sufficiently large.

In the **Armijo-Goldstein criterion** α_k is taken to be the largest number in the sequence $1, \frac{1}{2}, \frac{1}{4}, \dots$ for which

$$\psi(0) - \psi(\alpha_k) \geq \frac{1}{2}\alpha_k\psi(0)$$

is satisfied. Close to a simple zero x^* this criterion will automatically chose $\alpha_k = 1$. It then becomes identical to Newton's method and convergence becomes quadratic. Another common choice is to require that α_k satisfies the two conditions

$$\psi(\alpha_k) \leq \psi(0) + \mu\alpha_k\psi'(0), \quad |\psi'(\alpha_k)| \leq \eta|\psi'(0)|$$

where typically $\mu = 0.001$ and $\eta = 0.9$. The first condition ensures a sufficient decrease in $\|f(x)\|_2^2$ and the second that the gradient is decreased by a significant amount.

The addition of line searches to the Newton iteration greatly increases the range of nonlinear equations that can successfully be solved. However, if the Jacobian $J(x_k)$ is nearly singular, then h_k determined by (11.1.27) will be large and the linear model

$$f(x_k + \alpha_k h_k) \approx f(x_k) + \alpha_k J(x_k)h_k$$

inadequate. In this case the Newton direction tends to be very inefficient.

The idea in **trust region methods** is to avoid using a linear model outside its range of validity, see also Section 11.2.3. Here one takes $x_{k+1} = x_k + d_k$, where d_k solves the constrained linear least squares problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2^2, \text{ subject to } \|d_k\|_2 \le \Delta_k,$$

where Δ_k is a parameter, which is updated recursively. If the constraint is binding this problem can be solved by introducing a Lagrange parameter λ and minimizing

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2^2 + \lambda\|d_k\|_2^2. \tag{11.1.28}$$

Here λ is determined by the **secular equation** $\|d_k(\lambda)\|_2 = \Delta_k$. Note that the problem (11.1.28) is equivalent to the linear least squares problem

$$\min_{d_k} \left\| \begin{pmatrix} f(x_k) \\ 0 \end{pmatrix} + \begin{pmatrix} J(x_k) \\ \lambda^{1/2} I \end{pmatrix} d_k \right\|_2^2,$$

where the matrix always has full column rank for $\lambda > 0$.

A typical rule for updating Δ_{k+1} is to first calculate the ratio ρ_k of $\|f(x_k)\|_2^2 - \|f(x_k+d_k)\|_2^2$ to the reduction $\|f(x_k)\|_2^2 - \|f(x_k)+J(x_k)d_k\|_2^2$ predicted by the linear model. Then we take

$$\Delta_{k+1} = \begin{cases} \frac{1}{2}\|d_k\|, & \text{if } \rho_k \le 0.1; \\ \Delta_k, & \text{if } 0.1 < \rho_k \le 0.7; \\ \max\{2\|d_k\|, \Delta_k\}, & \text{if } \rho_k > 0.7. \end{cases}$$

The trust region is made smaller if the model is unsuccessful and is increased if a substantial reduction in the objective function is found. A difference to line search methods is that if $\Delta_{k+1} < \Delta_k$ we set $x_{k+1} = x_k$.

A related idea is used in **Powell's hybrid method**, where a linear combination of the steepest descent and the Newton (or the quasi-Newton) direction is used. Powell takes

$$x_{k+1} = x_k + \beta_k d_k + (1 - \beta_k)h_k, \quad 0 \le \beta_k \le 1,$$

where h_k is the Newton direction in (11.1.27), and

$$d_k = -\mu_k g_k, \qquad g_k = J(x_k)^T f(x_k), \qquad \mu_k = \|g_k\|_2^2 / \|J(x_k)g_k\|_2^2.$$

The choice of β_k is monitored by a parameter Δ_k, which equals the maximum allowed step size. The algorithm also includes a prescription for updating Δ_k. Powell chooses x_{k+1} as follows:

i. If $\|h_k\|_2 \le \Delta_k$ then $x_{k+1} = x_k + h_k$.

ii. If $\|g_k\|_2 \le \Delta_k \le \|h_k\|_2$, choose $\beta_k \in (0, 1]$ so that $\|x_{k+1} - x_k\|_2 = \Delta_k$.

iii. Otherwise set $x_{k+1} = x_k + \Delta_k g_k / \|g_k\|_2$.

The convergence is monitored by $\|f(x_k)\|_2$. When convergence is slow, Δ_k can be decreased, giving a bias towards steepest descent. When convergence is fast, Δ_k is increased, giving a bias towards the Newton direction.

Global methods for nonlinear systems may introduce other problems not inherent in the basic Newton method. The modification introduced may lead to slower convergence and even lead to convergence to a point where the equations are not satisfied.

11.1.8 Numerical Continuation Methods

When it is hard to solve the system $f(x) = 0$, or to find an initial approximation, continuation, embedding or homotopy methods are useful tools. Their use to solve nonlinear systems of equations goes back at least as far as Lahaye [1934]. Briefly, the idea is to find a simpler system $g(x) = 0$, for which the solution $x = x_0$ can be obtained without difficulty, and define a convex embedding (or **homotopy**)

$$H(x,t) = tf(x) + (1-t)g(x), \tag{11.1.29}$$

so that

$$H(x,0) = g(x), \qquad H(x,1) = f(x).$$

If the functions $f(x)$ and $g(x)$ are sufficiently smooth then a solution curve $x(t)$ exists, which satisfies the conditions $x(0) = x_0$, and $x(1) = x^*$. One now attempts to trace the solution curve $x(t)$ of (11.1.29) by computing $x(t_j)$ for an increasing sequence of values of t, $0 = t_0 < t_1 < \ldots < t_p = 1$ by solving the nonlinear systems

$$H(x, t_{j+1}) = 0, \qquad j = 0 : p-1, \tag{11.1.30}$$

by some appropriate method. The starting approximations can be obtained from previous results, e.g.,

$$x_0(t_{j+1}) = x(t_j),$$

or, if $j \geq 1$, by linear interpolation

$$x_0(t_{j+1}) = x(t_j) + \frac{t_{j+1} - t_j}{t_j - t_{j-1}}\big(x(t_j) - x(t_{j-1})\big).$$

This technique can be used in connection with any of the methods previously mentioned. For example, Newton's method can be used to solve (11.1.30)

$$x_{k+1} = x_k - \left(\frac{\partial H(x_k, t_{j+1})}{\partial x}\right)^{-1} H(x_k, t_{j+1}).$$

The step size should be adjusted automatically to approximately minimize the total number of iterations. A simpler strategy is to choose the number of increments M and take a constant step $\Delta t = 1/M$. If m is sufficiently large, then the iterative process will generally converge. However, the method may fail when turning points of the curve with respect of parameter t are encountered. In this case the embedding family has to be changed, or some other special measure must be taken. Poor performance can also occur because t is not well suited for parametrization. Often the arclength s of the curve provides a better parametrization

Embedding has important applications to the nonlinear systems encountered when finite-difference or finite-element methods are applied to nonlinear boundary-value problems; see Chapter 14. It is also an important tool in nonlinear optimization, e.g., in interior point methods. Often a better choice than (11.1.29) can be made for the embedding, where the systems for $t_j \neq 1$ also contribute to the insight into the questions which originally lead to the system. In elasticity, a technique called **incremental loading** is used, because $t = 1$ may correspond to an unloaded structure for which the solution is known, while $t = 0$ correspond to the actual loading. The technique is also called the **continuation** method.

If the equation $H(x,t) = 0$ is differentiated we obtain

$$\frac{\partial H}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial H}{\partial t} = 0.$$

This gives the differential equation

$$\frac{dx}{dt} = F(x,t), \qquad F(x,t) = -\Big(\frac{\partial H}{\partial x}\Big)^{-1} \frac{\partial H}{\partial t}.$$

Sometimes it is recommended to use a numerical method to integrate this differential equation with initial value $x(1) = x_1$ to obtain the solution curve $x(s)$, and in particular $x(1) = x^*$. However, to use a general purpose method for solving the differential equation is an unnaturally complicated approach. One should instead numerically integrate (11.1.30) very coarsely and then locally use a Newton-type iterative method for solving (11.1.29) as a corrector. This has the advantage that one takes advantage of the fact that the solution curve consists of solutions of (11.1.30), and uses the resulting strong contractive properties of Newton's method. Such predictor corrector continuation methods have been very successful, see Allgower and Georg [3, 1990]. The following algorithm uses Euler's method for integration as a predictor step and Newton's method as a corrector:

Algorithm 11.1.
Euler-Newton Method
Assume that $g(x_0) = 0$. Let $t_0 = 0$, and $h_0 > 0$ be an initial step length.

$x := x_0$; $\quad t_1 = t_0 + h_0$;
for $j = 1, 2, \ldots,$
  $x_j := x_{j-1} + h_{j-1} F(x_{j-1}, t_{j-1})$; Euler step
  repeat
    $x_j := x_j - (H'(x_j, t_j))^{-1} H(x_j, t_j)$; Newton step
  until convergence
  if $t_j \equiv 1$ **then** *stop*
  else $t_{j+1} = t_j + h_j$; $\quad h_j > 0$; new steplength
end

Note the possibility of using the same Jacobian in several successive steps. The convergence properties of the Euler-Newton Method and other predictor-corrector methods are discussed in Allgower and Georg [3, 1990].

Review Questions

1. Describe the nonlinear Gauss–Seidel method.

2. Describe Newton's method for solving a nonlinear system of equations.

3. In order to get global convergence Newton's method has to be modified. Two different approaches are much used. Describe the main features of these modifications.

4. For large n the main cost of an iteration step in Newton's method is the evaluation and factorizing of the matrix of first derivatives. Describe some ways to reduce this cost.

5. Define what is meant by the completeness of a space, a Banach space, a Lipschitz constant and a contraction. Formulate the Contraction Mapping Theorem. You don't need to work out the full proof, but tell where in the proof the completeness is needed.

6. Give the essential features of the assumptions needed in the theorem in the text which is concerned with the convergence of Newton's method for a nonlinear system. What is the order of convergence for simple roots?

7. Describe the essential features of numerical continuation methods for solving a nonlinear system $f(x) = 0$. How is a suitable convex embedding constructed?

Problems

1. The fixed point iteration in Example 11.1.1 can be written $u_{k+1} = \phi(u_k)$, where $u = (x, y)^T$. Compute $\|\phi(u^*)\|_\infty$ for the two roots $u^* = (1, 0)^T$ and $(0, 1)^T$, and use the result to explain the observed convergence behavior.

2. Consider the system of equations

$$\begin{aligned} x_1^2 - x_2 + \alpha &= 0, \\ -x_1 + x_2^2 + \alpha &= 0. \end{aligned}$$

Show that for $\alpha = 1, 1/4$, and 0 there is no solution, one solution, and two solutions, respectively.

3. (a) Describe graphically in the (x, y)-plane nonlinear Gauss–Seidel applied to the system $f(x, y) = 0$, $g(x, y) = 0$. Consider all four combinations of orderings for the variables and the equations.

(b) Do the same thing for nonlinear Jacobi. Consider both orderings of the equations.

4. The system of equations

$$\begin{aligned} x &= 1 + h^2(e^{y\sqrt{x}} + 3x^2) \\ y &= 0.5 + h^2 \tan(e^x + y^2), \end{aligned}$$

can, for small values of h, be solved by fixed point iteration. Write a program which uses this method to solve the system for $h = 0, 0.01, \dots, 0.10$. For $h = 0$ take $x_0 = 1$, $y_0 = 0.5$, else use the solution for the previous value of h. The iterations should be broken off when the changes in x and y are less than $0.1h^4$.

5. For each of the roots of the system in Example 11.1.1,

$$\begin{aligned} x^2 - 2x - y + 1 &= 0 \\ x^2 + y^2 - 1 &= 0 \end{aligned}$$

determine whether or not the following iterations are locally convergent:
(a) $x^{k+1} = (1 - y_k^2)^{1/2}, \quad y_{k+1} = (x_k - 1)^2$.
(b) $x_{k+1} = y_k^{1/2} + 1, \quad y_{k+1} = (1 - x_k^2)$.

6. Apply two iterations of Newton's method to the equations of Problem 5, using the initial approximations $x_0 = 0.1$, and $y_0 = 1.1$.

7. If some of the equations in the system $f(x) = 0$ are linear, Newton's method will take this into account. Show that if (say) $f_i(x)$ is linear, then the Newton iterates x_k, $k \geq 1$, will satisfy $f_i(x_k) = 0$.

Figure 11.1.1. *A rotating double pendulum.*

8. A double pendulum rotates with angular velocity ω around a vertical axis (like a centrifugal regulator). At equilibrium the two pendulums make the angles x_1 and x_2 to the vertical axis, see Figure 11.1.1. It can be shown that the angles are determined by the equations

$$\begin{aligned} &\tan x_1 - k(2\sin x_1 + \sin x_2) = 0, \\ &\tan x_2 - 2k(\sin x_1 + \sin x_2) = 0. \end{aligned}$$

where $k = l\omega^2/(2g)$.
(a) Solve by Newton's method the system for $k = 0.3$, with initial guesses $x_1 = 0.18$, $x_2 = 0.25$. How many iterations are needed to obtain four correct decimals?
(b) Determine the solutions with four correct decimals and plot the results for

$$k = 0.30, 0.31, \dots, 0.35, 0.4, 0.5, \dots, 0.9, 1, 2, 3, 4, 5, 10, 15, 20, \infty.$$

Use the result obtained for the previous k as initial guess for the new k. Record also how many iterations are needed for each value of k.
(c) Verify that the Jacobian is singular for $x_1 = x_2 = 0$, when $k = 1 - 1/\sqrt{2} \approx 0.2929$.

A somewhat sloppy theory suggests that

$$x_1 \approx x_2 \approx \sqrt{k - (1 - 1/\sqrt{2})}, \qquad 0 \le k - (1 - 1/\sqrt{2}) \ll 1.$$

Do your results support this theory?

9. Describe how to apply the Newton idea to the solution of the steady state of a Matrix Riccati equation, i.e., to the solution of a matrix equation of the form,

$$A + BX + XC + XDX = 0,$$

where A, B, C, D are rectangular matrices of appropriate size. Assume that an algorithm for equations of the form $PX + XQ = R$ is given. Under what condition does such a linear equation have a unique solution? You don't need to discuss how to find the first approximation.

10. (a) Derive the formula for $\min B(\eta)$ and the optimal choice of η for the *uncentered* difference approximation to $g'(x)v$, also the simplified error estimate (for $\xi = \frac{1}{2}\|g\|/\|g'\|$).

(b) Work out the details of the study of the directional second derivative.

11. Investigate, for various functions f, g, the ratio of values of $B(\eta)$, obtained with the optimal η and with the value of η derived from he simplified estimate of ξ. Take, for example, $g(x) = e^{\alpha x}$, $g(x) = x^{-k}$.

12. Suppose that $x \in \mathbf{R}^n$, where n is divisible by 3, and that the Jacobian is a square *tridiagonal* matrix.

(a) Design an algorithm, where all the elements of the Jacobian are found by four evaluations of $g(x)$, when the uncentered difference approximation is used.

(b) You may obtain the elements packed in three vectors. How do you unpack them into an $n \times n$ matrix? How many function evaluations do you need with the centered difference approximation?

(c) Generalize to the case of an arbitrary *banded Jacobian*.

Comment: This idea was first published by Curtis, Powell, and Reid [82]

11.2 Nonlinear Least Squares Problems

11.2.1 Introduction

In this section we discuss the numerical solution of nonlinear least squares problem. Let $f:\ \mathbf{R}^n \to \mathbf{R}^m$, $m \ge n$, and consider the problem

$$\min_{x \in \mathbf{R}^n} \phi(x), \qquad \phi(x) = \tfrac{1}{2}\|f(x)\|_2^2 = \tfrac{1}{2} f(x)^T f(x). \tag{11.2.1}$$

This is a special case of the general optimization problem in $\mathbf{R}^n$ studied in Section 11.3. We will in the following mainly emphasize those aspects of the problem (11.2.1), which derive from the special form of $\phi(x)$. (Note that the nonlinear system $f(x) = 0$ is equivalent to (11.2.1) with $m = n$.)

Fitting data to a mathematical model is an important source of nonlinear least squares problems. Here one attempts to fit given data (y_i, t_i), $i = 1 : m$ to a model function $y = h(x, t)$. If we let $r_i(x)$ represent the error in the model prediction for the i:th observation,

$$r_i(x) = y_i - h(x, t_i), \qquad i = 1, \dots, m,$$

we want to minimize some norm of the vector $r(x)$. The choice of the least squares measure is justified here, as for the linear case, by statistical considerations. If the observations have equal weight, this leads to the minimization problem in (11.2.1) with $f(x) = r(x)$.

Example 11.2.1.

Exponential fitting problems occur frequently–e.g., the parameter vector x in the expression

$$y(t, x) = x_1 + x_2 e^{x_4 t} + x_3 e^{x_5 t}$$

is to be determined to give the best fit to m observed points (t_i, y_i), $i = 1 : m$, where $m > 5$. Here $y(t, x)$ is linear the parameters x_4, x_5. but nonlinear in x_4, x_5. Hence this problem cannot be handled by the methods in Chapter 7. Special methods for problems which are nonlinear only in some of the parameters are given in Section 11.2.5.

The standard methods for the nonlinear least squares problem require derivative information about the component functions of $f(x)$. We assume here that $f(x)$ is twice continuously differentiable. It is easily shown that the gradient of $\phi(x) = \frac{1}{2} f^T(x) f(x)$ is

$$g(x) = \nabla\phi(x) = J(x)^T f(x), \tag{11.2.2}$$

where

$$J(x)_{ij} = \frac{\partial f_i(x)}{\partial x_j} \in \mathbf{R}^{m\times n}, \quad i = 1 : m, \quad j = 1 : n.$$

is the Jacobian matrix of $f(x)$. The Hessian matrix is

$$H(x) = \nabla^2\phi(x) = J(x)^T J(x) + Q(x), \quad Q(x) = \sum_{i=1}^{m} f_i(x) G_i(x), \tag{11.2.3}$$

where $G_i(x) \in \mathbf{R}^{n\times n}$, is the Hessian matrix of $f_i(x)$ with elements

$$G_i(x)_{jk} = \frac{\partial^2 f_i(x)}{\partial x_j \partial x_k}, \quad i = 1 : m, \quad j, k = 1 : n. \tag{11.2.4}$$

The special forms of the gradient $g(x)$ and Hessian $H(x)$ can be exploited by methods for the nonlinear least squares problem.

A necessary condition for x^* to be a local minimum of $\phi(x)$ is that x^* is a stationary point, i.e., satisfies

$$g(x^*) = J(x^*)^T f(x^*) = 0. \tag{11.2.5}$$

A necessary condition for a stationary point x^* to be a local *minimum* of $\phi(x)$ is that the Hessian matrix $H(x)$ is positive definite at x^*.

There are basically two different ways to view problem (11.2.1). One could think of this problem as arising from an overdetermined system of nonlinear equations $f(x) = 0$. It is then natural to approximate $f(x)$ by a linear model around a given point x_k

$$\tilde{f}(x) = f(x_k) + J(x_k)(x - x_k), \tag{11.2.6}$$

and use the solution p_k to the linear least squares problem

$$\min_p \|f(x_k) + J(x_k)p\|_2. \tag{11.2.7}$$

to derive an new (hopefully improved) improved approximate solution $x_{k+1} = x_k + p_k$. This approach, which only uses first order derivative information about $f(x)$, leads to a class of methods called **Gauss–Newton** type methods. These methods, which in general only have linear rate of convergence, will be discussed in Section 11.2.2.

In the second approach (11.2.1) is viewed as a special case of unconstrained optimization in $\mathbf{R^n}$. A quadratic model at a point x_k is used,

$$\tilde{\phi}_c(x) = \phi(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k), \tag{11.2.8}$$

where the gradient and Hessian of $\phi(x) = \frac{1}{2}f^T(x)f(x)$ are given by (11.2.2) and (11.2.3). The minimizer of $\tilde{\phi}_c(x)$ is given by $x_{k+1} = x_k + p_k$, where

$$p_k = -H(x_k)^{-1}J(x_k)^T f(x_k). \tag{11.2.9}$$

This method is equivalent to Newton's method applied to (11.2.1), which usually is locally quadratically convergent.

The Gauss–Newton method can be thought of as arising from neglecting the second derivative term

$$Q(x) = \sum_{i=1}^m f_i(x)G_i(x),$$

in the Hessian $H(x_k)$. Note that $Q(x_k)$ will be small close to the solution x^* if either the residual norm $\|f(x^*)\|$ is small or if $f(x)$ is only mildly nonlinear. The behavior of the Gauss–Newton method can then be expected to be similar to that of Newton's method. In particular for a consistent problem where $f(x^*) = 0$ the local convergence will be the same for both methods. However, for moderate to large residual problems the local convergence rate for the Gauss–Newton method can be much inferior to that of Newton's method.

The cost of computing and storing the mn^2 second derivatives (11.2.4) in $Q(x)$ can be prohibitively high. Note, however, that for curve fitting problems the function values $f_i(x) = y_i - h(x, t_i)$ and the derivatives $\partial^2 f_i(x)/\partial x_j \partial x_k$, can be obtained from the single function $h(x, t)$. If $h(x, t)$ is composed of, e.g., simple exponential and trigonometric functions then the Hessian can sometimes be computed cheaply. Another case when it may be feasible to store approximations to all $G_i(x)$, $i = 1 : m$, is when every function $f_i(x)$ only depends on a small subset of the n variables. Then both the Jacobian $J(x)$ and the Hessian matrices $G_i(x)$ will be *sparse* and special methods, such as those discussed in Section 6.5 may be applied.

11.2.2 Gauss–Newton-Type Methods

The Gauss–Newton method for problem (11.2.1) is based on a sequence of linear approximations of $f(x)$ of the form (11.2.6). If x_k denotes the current approximation then the Gauss–Newton step d_k is a solution to the linear least squares problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \qquad d_k \in \mathbf{R}^n. \tag{11.2.10}$$

and the new approximation is $x_{k+1} = x_k + d_k$. The solution d_k is unique if $\operatorname{rank}(J(x_k)) = n$. Since $J(x_k)$ may be ill-conditioned or singular, d_k should be computed by a stable method using, e.g., the QR- or SVD-decomposition of $J(x_k)$.

The Gauss–Newton step $d_k = -J(x_k)^\dagger f(x_k)$ has the following important properties:

(i) d_k is invariant under linear transformations of the independent variable x, i.e., if $\tilde{x} = Sx$, S nonsingular, then $\tilde{d_k} = Sd_k$.

(ii) if $J(x_k)^T f(x_k) \neq 0$ then d_k is a descent direction for $\phi(x) = \frac{1}{2}f^T(x)f(x)$,

The first property is easily verified. To prove the second property we note that

$$d_k^T g(x_k) = -f(x_k)^T J^\dagger(x_k)^T J(x_k)^T f(x_k) = -\|P_{J_k} f(x_k)\|_2^2, \tag{11.2.11}$$

where $P_{J_k} = J(x_k)J^\dagger(x_k) = P_{J_k}^2$ is the orthogonal projection onto the range space of $J(x_k)$. Further if $J(x_k)^T f(x_k) \neq 0$ then $f(x_k)$ is not in the nullspace of $J(x_k)^T$ and it follows that $P_{J_k} f(x_k) \neq 0$. This proves (ii).

The Gauss–Newton method can fail at an intermediate point where the Jacobian is rank deficient or illconditioned. Formally we can take d_k to be the minimum norm solution

$$d_k = -J(x_k)^\dagger f(x_k).$$

In practice it is necessary to include some strategy to estimate the numerical rank of $J(x_k)$, cf. Section 7.3.2 and 7.6.2. That the assigned rank can have a decisive influence is illustrated by the following example:

Example 11.2.2. (Gill, Murray and Wright [165, p. 136])

Let $J = J(x_k)$ and $f(x_k)$ be defined by

$$J = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix}, \qquad f = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix},$$

where $\epsilon \ll 1$ and f_1 and f_2 are of order unity. If J is considered to be of rank two then the search direction $d_k = s_1$, whereas if the assigned rank is one $d_k = s_2$, where

$$s_1 = -\begin{pmatrix} f_1 \\ f_2/\epsilon \end{pmatrix}, \qquad s_2 = -\begin{pmatrix} f_1 \\ 0 \end{pmatrix}.$$

Clearly the two directions s_1 and s_2 are almost orthogonal and s_1 is almost orthogonal to the gradient vector $J^T f$.

Usually it is preferable to *underestimate* the rank except when $\phi(x)$ is actually close to an ill-conditioned quadratic function. One could also switch to a search direction along the negative gradient $-g_k = -J(x_k)^T f(x_k)$, or use a linear combination

$$d_k - \mu_k g_k, \qquad \mu_k = \|g_k\|_2^2 / \|J(x_k) g_k\|_2^2.$$

as in Powell's method.

The Gauss–Newton method as described above has several advantages. It solves linear problems in just one iteration and has fast convergence on small residual and mildly nonlinear problems. However, it may not be locally convergent on problems that are very nonlinear or have large residuals.

To analyze the rate of convergence of Gauss–Newton type methods let $J^\dagger(x)$ denote the pseudoinverse of $J(x)$, and assume that $\text{rank}(J(x)) = n$. Then $I = J^\dagger(x)J(x)$, and (11.2.3) can be written in the form

$$H(x) = J(x)^T (I - \gamma K(x)) J(x), \qquad K(x) = J^\dagger(x)^T G_w(x) J^\dagger(x). \tag{11.2.12}$$

where $\gamma = \|f(x)\|_2 \neq 0$, and

$$G_w(x) = \sum_{i=1}^m w_i G_i(x), \qquad w(x) = -\frac{1}{\gamma} f(x). \tag{11.2.13}$$

The matrix $K(x)$ is symmetric, and has a geometric interpretation. It is called the **normal curvature matrix** of the n-dimensional surface $z = f(x)$ in $\mathbf{R}^m$, with respect to the unit normal vector $w(x)$. The quantities $\rho_i = 1/\kappa_i$, where

$$\kappa_1 \geq \kappa_2 \geq \ldots \geq \kappa_n.$$

are the eigenvalues of $K(x)$, are called the **principal radii of curvature** of the surface.

The Hessian matrix $H(x^*)$ is positive definite and x^* a local minimum if and only if $u^T H(x^*) u > 0$, for all $u \in \mathbf{R}^n \neq 0$. If $\text{rank}(J(x^*)) = n$, it follows that $u \neq 0 \Rightarrow J(x^*)u \neq 0$, and hence $H(x^*)$ is positive definite when $I - \gamma K(x^*)$ is positive definite, i.e., when

$$1 - \gamma\kappa_1 > 0. \tag{11.2.14}$$

If $1 - \gamma\kappa_1 \leq 0$ then the least squares problem has a saddle point at x^* or if also $1 - \gamma\kappa_n < 0$ even a local maximum at x^*.

Example 11.2.3.

The geometrical interpretation of the nonlinear least squares problem (11.2.1) is to find a point on the surface $\{f(x) \mid x \in \mathbf{R}^n\}$ in $\mathbf{R}^m$ closest to the origin. In case of data fitting $f_i(x) = y_i - h(x, t_i)$, and it is more illustrative to consider the surface

$$z(x) = (h(x, t_1), \ldots, h(x, t_m))^T \in \mathbf{R}^m.$$

The problem is then to find the point $z(x^*)$ on this surface closest to the observation vector $y \in \mathbf{R}^m$. This is illustrated in Figure 11.2.1 for the simple case of $m = 2$

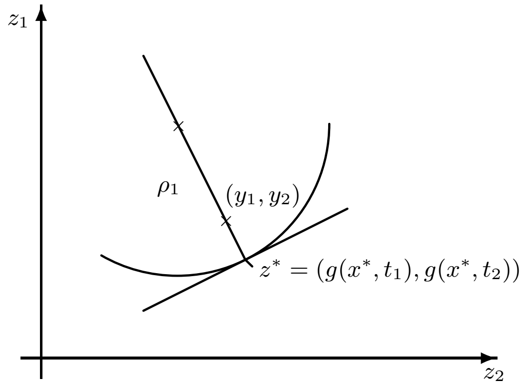

Figure 11.2.1. *Geometry of the data fitting problem for $m = 2$, $n = 1$.*

observations and a scalar parameter x. Since in the figure we have $\gamma = \|y - z(x^*)\|_2 < \rho$, it follows that $1 - \gamma\kappa_1 > 0$, which is consistent with the fact that x^* is a local minimum. In general the solution (if it exists) is given by an orthogonal projection of y onto the surface $z(x)$. Compare the geometrical interpretation in Figure 8.1.1 for the linear case $z(x) = Ax$.

It can be shown that the asymptotic rate of convergence of the Gauss–Newton method in the neighborhood of a critical point x^* is equal to

$$\rho = \gamma \max(\kappa_1, -\kappa_n),$$

where κ_i are the eigenvalues of the of the normal curvature matrix $K(x)$ in (11.2.12) evaluated at x^* and $\gamma = \|f(x^*)\|_2 = 0$. In general convergence is linear, but if $\gamma = 0$ then convergence becomes superlinear. Hence the asymptotic rate of convergence of the undamped Gauss–Newton method is fast when either

(i) the residual norm $\gamma = \|r(x^*)\|_2$ is small, or

(ii) $f(x)$ is mildly nonlinear, i.e. $|\kappa_i|$, $i = 1 : n$ are small.

If x^* is a saddle point then $\gamma\kappa_1 \geq 1$, i.e., using undamped Gauss–Newton one is repelled from a saddle point. This is an excellent property since saddle points are not at all uncommon for nonlinear least squares problems.

The Gauss–Newton method can be modified for global convergence. in a similar way as described in Section 11.1.7 Newton's method. If the Gauss–Newton direction d_k is used as a search direction we consider the one-dimensional minimization problem

$$\min_{\lambda} \|f(x_k + \lambda d_k)\|_2^2.$$

As remarked above it is in general not worthwhile to solve this minimization accurately. Instead we can take λ_k to be the largest number in the sequence $1, \frac{1}{2}, \frac{1}{4}, \ldots$

for which

$$\|f(x_k)\|_2^2 - \|f(x_k + \lambda_k d_k)\|_2^2 \geq \frac{1}{2}\lambda_k \|P_{J_k} f(x_k)\|_2^2.$$

Here $\lambda = 1$ corresponds to the full Gauss–Newton step. Since d_k is a descent direction, this damped Gauss–Newton method is locally convergent on almost all nonlinear least squares problems. In fact is is usually even globally convergent. For large residual or very nonlinear problems convergence may still be slow.

The rate of convergence for the Gauss–Newton method with *exact* line search can be shown to be

$$\tilde{\rho} = \gamma(\kappa_1 - \kappa_n)/(2 - \gamma(\kappa_1 + \kappa_n)).$$

We have $\tilde{\rho} = \rho$ if $\kappa_n = -\kappa_1$ and $\tilde{\rho} < \rho$ otherwise. Since $\gamma\kappa_1 < 1$ implies $\tilde{\rho} < 1$, we always get convergence close to a local minimum. This is in contrast to the undamped Gauss–Newton method, which may fail to converge to a local minimum.

The rate of convergence for the undamped Gauss–Newton method can be estimated during the iterations from

$$\rho_{\text{est}} = \|P_J(x_{k+1}) r_{k+1}\|_2 / \|P_J(x_k) r_k\|_2 = \rho + O(\|x_k - x^*\|_2^2). \tag{11.2.15}$$

Since $P_J(x_k) r_k = J(x_k) J(x_k)^\dagger r_k = -J(x_k) p_k$ the cost of computing this estimate is only one matrix-vector multiplication. When $\rho_{\text{est}} > 0.5$ (say) then one should consider switching to a method using second derivative information, or perhaps evaluate the quality of the underlying model.

11.2.3 Trust Region Methods

(See Conn, Gould and Toint [74].)

Even the damped Gauss–Newton method can have difficulties to get around an intermediate point where the Jacobian matrix rank deficient. This can be avoided either by taking second derivatives into account (see Section 11.2.4) or by further stabilizing the damped Gauss–Newton method to overcome this possibility of failure. Methods using the latter approach were first suggested by Levenberg [272, 1944] and Marquardt [262, 1963]. Here a search direction d_k is computed by solving the problem

$$\min_{d_k} \{\|f(x_k) + J(x_k) d_k\|_2^2 + \mu_k \|d_k\|_2^2\}, \tag{11.2.16}$$

where the parameter $\mu_k \geq 0$ controls the iterations and limits the size of d_k. Note that if $\mu_k > 0$ then d_k is well defined even when $J(x_k)$ is rank deficient. As $\mu_k \to \infty$, $\|d_k\|_2 \to 0$ and d_k becomes parallel to the steepest descent direction. It can be shown that d_k is the solution to the least squares problem with quadratic constraint

$$\min_{d_k} \|f(x_k) + J(x_k) d_k\|_2, \quad \text{subject to} \quad \|d_k\|_2 \leq \delta_k, \tag{11.2.17}$$

where $\mu_k = 0$ if the constraint in (11.2.17) is not binding and $\mu_k > 0$ otherwise. The set of feasible vectors d_k, $\|d_k\|_2 \leq \delta_k$ can be thought of as a region of trust for the linear model $f(x) \approx f(x_k) + J(x_k)(x - x_k)$.

The following trust region strategy has proved very successful in practice:

Let x_0, D_0 and δ_0 be given and choose $\beta \in (0, 1)$. For $k = 0, 1, 2, \ldots$ do

(a) Compute $f(x_k)$, $J(x_k)$, and determine d_k as a solution to the subproblem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \quad \text{subject to} \quad \|D_k d_k\|_2 \le \delta_k,$$

where D_k is a diagonal scaling matrix.

(b) Compute the ratio $\rho_k = \big(\|f(x_k)\|_2^2 - \|f(x_k + d_k)\|_2^2\big)/\psi_k(d_k)$, where

$$\psi_k(d_k) = \|f(x_k)\|_2^2 - \|f(x_k) + J(x_k)d_k\|_2^2$$

is the model prediction of the decrease in $\|f(x_k)\|_2^2$.

(c) If $\rho_k > \beta$ the step is successful and we set $x_{k+1} = x_k + d_k$, and $\delta_{k+1} = \delta_k$; otherwise set $x_{k+1} = x_k$ and $\delta_{k+1} = \beta\delta_k$. Update the scaling matrix D_k.

The ratio ρ_k measures the agreement between the linear model and the nonlinear function. After an unsuccessful iteration δ_k is reduced. The scaling D_k can be chosen such that the algorithm is scale invariant, i.e., the algorithm generates the same iterations if applied to $r(Dx)$ for any nonsingular diagonal matrix D. It can be proved that if $f(x)$ is continuously differentiable, $f'(x)$ uniformly continuous and $J(x_k)$ bounded then this algorithm will converge to a stationary point.

A trust region implementation of the Levenberg-Marquard method will give a Gauss–Newton step close to the solution of a regular problem. Its convergence will therefore often be slow for large residual or very nonlinear problems. Methods using second derivative information , see Section 11.2.4 are somewhat more efficient but also more complex than the Levenberg-Marquardt methods.

11.2.4 Newton-Type Methods

The analysis in the Section 11.2.2 showed that for large residual problems and strongly nonlinear problems, methods of Gauss–Newton type may converge slowly. Also, these methods can have problems at points where the Jacobian is rank deficient. When second derivatives of $f(x)$ are available Newton's method, which uses the quadratic model (11.2.8), can be used to overcome these problems. The optimal point d_k of this quadratic model, satisfies the linear system

$$H(x_k)d_k = -J(x_k)^T f(x_k), \tag{11.2.18}$$

where $H(x_k)$ is the Hessian matrix at x_k, and $x_k + d_k$ is chosen as the next approximation.

It can be shown, see Dennis and Schnabel [101, 1983, p. 229], that Newton's method is quadratically convergent to a local minimum x^* as long as $H(x)$ is Lipschitz continuous around x_k and $H(x^*)$ is positive definite. To get global convergence a line search algorithm is used, where the search direction d_k is taken as the Newton direction. Note that the Hessian matrix $H(x_k)$ must be positive definite in order for the Newton direction d_k to be a descent direction.

Newton's method is not often used since the second derivative term $Q(x_k)$ in the Hessian is rarely available at a reasonable cost. However, a number of methods have been suggested that partly takes the second derivatives into account, either explicitly or implicitly. An implicit way to obtain second derivative information is to use a general quasi-Newton optimization routine, which successively builds up approximations B_k to the Hessian matrices $H(x_k)$. The search directions are computed from

$$B_k d_k = -J(x_k)^T f(x_k),$$

where B_k satisfies the quasi-Newton conditions

$$B_k s_k = y_k, \qquad s_k = x_k - x_{k-1}, \qquad y_k = g(x_k) - g(x_{k-1}), \tag{11.2.19}$$

where $g(x_k) = J(x_k)^T f(x_k)$. As starting value $B_0 = J(x_0)^T J(x_0)$ is recommended.

The direct application of quasi-Newton methods to the nonlinear least squares problem outlined above has not been so efficient in practice. One reason is that these methods disregard the information in $J(x_k)$, and often $J(x_k)^T J(x_k)$ is the dominant part of $H(x_k)$. A more successful approach is to approximate $H(x_k)$ by $J(x_k)^T J(x_k) + S_k$, where S_k is a quasi-Newton approximation of the term $Q(x_k)$. Initially one takes $S_0 = 0$. The quasi-Newton relations (11.2.19) can now be written

$$S_k s_k = z_k, \qquad z_k = \big(J(x_k) - J(x_{k-1})\big)^T f(x_k), \tag{11.2.20}$$

where S_k is required to be symmetric. It can be shown that a solution to (11.2.20) which minimizes the change from S_{k-1} in a certain weighted Frobenius norm is given by the update formula

$$B_k = B_{k-1} + \frac{w_k y_k^T + y_k w_k^T}{y_k^T s_k} - \frac{w_k^T s_k y_k y_k^T}{y_k^T s_k^2}, \tag{11.2.21}$$

where $s_k = x_k - x_{k-1}$, and $w_k = z_k - B_{k-1} s_k$.

In some cases the updating (11.2.21) gives inadequate results. This motivates the inclusion of "sizing" in which the matrix B_k is replaced by $\tau_k B_k$, where

$$\tau_k = \min\{|s_k^T z_k|/|s_k^T B_k s_k|, 1\}.$$

This heuristic choice ensures that S_k converges to zero for zero residual problems, which improves the convergence behavior.

In another approach, due to Gill and Murray [163], $J(x_k)^T J(x_k)$ is regarded as a good estimate of the Hessian in the right invariant subspace corresponding to the large singular values of $J(x_k)$. In the complementary subspace the second derivative term $Q(x_k)$ is taken into account. Let the singular value decomposition of $J(x_k)$ be

$$J(x_k) = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T, \qquad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n),$$

where the singular values are ordered so that $\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_n$. Then putting $Q_k = Q(x_k)$ the equations for the Newton direction $d_k = Vq$ can be written

$$(\Sigma^2 + V^T Q_k V) q = -\Sigma r_1, \qquad r_1 = (\, I_n \quad 0 \,)\, U^T f(x_k). \tag{11.2.22}$$

We now split the singular values into two groups, $\Sigma = \mathrm{diag}(\Sigma_1, \Sigma_2)$, where $\Sigma_1 = \mathrm{diag}(\sigma_1, \dots, \sigma_r)$ are the "large" singular values. If we partition V, q and $\bar{r}$ conformally, then the first r equations in (11.2.22) can be written.

$$(\Sigma_1^2 + V_1^T Q_k V_2) q_1 + V_1^T Q_k V_2 q_2 = -\Sigma_1 \bar{r}_1.$$

If the terms involving Q_k are neglected compared to $\Sigma_1^2 q_1$ we get $q_1 = -\Sigma_1^{-1} \bar{r}_1$. If this is substituted into the last $(n-r)$ equations we can solve for q_2 from

$$(\Sigma_2^2 + V_2^T Q_k V_2) q_2 = -\Sigma_2 \bar{r}_2 - V_2^T Q_k V_1 q_1.$$

The approximate Newton direction is then given by $d_k = Vq = V_1 q_1 + V_2 q_2$. The splitting of the singular values is updated at each iteration, the idea being to maintain r close to n as long as adequate progress is made.

There are several alternative ways to implement the method by Gill and Murray [163, 1978]. If Q_k is not available explicitly, then a finite difference approximation to $V_2^T Q_k V_2$ can be obtained as follows. Let v_j be a column of V_2 and h a small positive scalar. Then

$$(\nabla r_i(x_k + h v_j) - \nabla r_i(x_k))/h = v_j^T G_i(x_k) + O(h).$$

The vector on the left hand side is the ith row of $(J(x_k + h v_j) - J(x_k))/h$. Multplying with $r_i(x_k)$ and adding we obtain

$$\begin{aligned} r(x_k)^T (J(x_k + h v_j) - J(x_k))/h &= v_j^T \sum_{i=1}^m r_i(x_k) G_i(x_k) + O(h) \\ &= v_j^T Q_k + O(h). \end{aligned}$$

Repeating this for all columns in V_2 we obtain an approximation for $V_2^T Q_k$ and we finally form $(V_2^T Q_k) V_2$.

11.2.5 Separable Problems

In some structured nonlinear least squares problems it is advantageous to separate the parameters into two sets. For example, suppose that we want to minimize the nonlinear functional

$$\|r(y,z)\|^2, \quad r_i(y,z) = s_i - \sum_{j=1}^p y_j \phi_j(z; t_i), \quad i = 1:m. \tag{11.2.23}$$

where (s_i, t_i), $i = 1:m$, is the data to be fitted. Here $y \in \mathbf{R}^p$ are linear parameters and $z \in \mathbf{R}^q$ are nonlinear parameters. Simple examples of nonlinear functions $\phi_j(z; t_i)$ are exponential or rational functions $\phi_j = e^{z_j t}$, $\phi_j = 1/(t - z_j)$. The full least squares problem can be written in the form

$$\min_{y,z} \|g(z) - \Phi(z) y\|^2, \tag{11.2.24}$$

where $\Phi(z)$ is a matrix whose jth column equals $\phi_j(z;t_i)$. For any fixed values of z the problem of finding the corresponding optimal values of y is a *linear* least squares problem that can be easily solved. The solution can be expressed as

$$y(z) = \Phi^{\dagger}(z)g(z), \tag{11.2.25}$$

where $\Phi^{\dagger}(z)$ is the pseudoinverse of $\Phi(z)$. The expression (11.2.25) allows the linear parameters to be eliminated in (11.2.24) and the original minimization problem to be cast in the form

$$\min_z \|(I - P_{\Phi(z)})g\|^2, \quad P_{\Phi(z)} = \Phi(z)\Phi(z)^{+}, \tag{11.2.26}$$

where $P_{\Phi(z)}$ is the orthogonal projector onto the the column space of $\Phi(z)$. This is a pure nonlinear problem of reduced dimension. The **variable projection method** consists of solving (11.2.26), for example by a Gauss–Newton–Marquardt method, obtaining the optimal vector z. The linear parameters are then computed from $y = \Phi(z)^{+}g$.

Many practical nonlinear least squares problems are separable in this way. A particularly simple case is when $r(y,z)$ is linear in both y and z so that we also have

$$r(y,z) = h(y) - \Psi(y)z, \qquad \Psi(y) \in \mathbf{R}^{m\times q}.$$

To use the Gauss–Newton method we need a formula for the derivative of an orthogonal projection $P_{\Phi(z)} = \Phi(z)\Phi(z)^{+}$. The following formula was shown by Golub and Pereyra [179] for any symmetric generalized inverse.

Lemma 11.2.1.

Let $A = A(\alpha) \in \mathbf{R}^{m\times n}$ be a matrix of local constant rank and A^- be any symmetric generalized inverse, i.e. $AA^-A = A$ and $(AA^-)^T = AA^-$. Then

$$\frac{d}{d\alpha}(AA^-) = P_{\mathcal{N}(A^T)}\frac{dA}{d\alpha}A^- + (A^-)^T\frac{dA^T}{d\alpha}P_{\mathcal{N}(A^T)}, \tag{11.2.27}$$

where $P_{\mathcal{R}(A)} = I - AA^-$.

Proof. Since $P_{\mathcal{R}(A)}A = A$,

$$\frac{d}{d\alpha}(P_{\mathcal{R}(A)}A) = \frac{d}{d\alpha}(P_{\mathcal{R}(A)})A + P_{\mathcal{R}(A)}\frac{dA}{d\alpha} = \frac{dA}{d\alpha},$$

and hence

$$\frac{d}{d\alpha}(P_{\mathcal{R}(A)})A = \frac{dA}{d\alpha} - P_{\mathcal{R}(A)}\frac{dA}{d\alpha} = (P_{\mathcal{N}(A^T)})\frac{dA}{d\alpha}.$$

Thus, since $P_{\mathcal{R}(A)}) = AA^-$,

$$\frac{d}{d\alpha}(P_{\mathcal{R}(A)})P_{\mathcal{R}(A)} = (P_{\mathcal{N}(A^T)})\frac{dA}{d\alpha}A^-. \tag{11.2.28}$$

Since an orthogonal projector is symmetric we have

$$\left(\frac{d}{d\alpha}(P_{\mathcal{R}(A)})P_{\mathcal{R}(A)}\right)^T = P_{\mathcal{R}(A)}\frac{d}{d\alpha}(P_{\mathcal{R}(A)}) \tag{11.2.29}$$

we finally obtain from (11.2.28) and (11.2.29)

$$\begin{aligned}\frac{d}{d\alpha}(P_{\mathcal{R}(A)}) = \frac{d}{d\alpha}(P^2_{\mathcal{R}(A)}) &= \frac{d}{d\alpha}(P_{\mathcal{R}(A)})P_{\mathcal{R}(A)} + P_{\mathcal{R}(A)}\frac{d}{d\alpha}(P_{\mathcal{R}(A)}) \\ &= (P_{\mathcal{N}(A^T)})\frac{dA}{d\alpha}A^- + (A^-)^T\frac{dA^T}{d\alpha}P_{\mathcal{N}(A^T)},\end{aligned}$$

which completes the proof. □

Example 11.2.4.

The exponential fitting problem, for example,

$$\min_{y,z}\sum_{i=1}^{m}(y_1e^{z_1t_i} + y_2e^{z_2t_i} - g_i)^2.$$

arises in many applications, e.g., where we have reactions with different time constant. This problem is often ill-conditioned because the same data can be well approximated by different exponential sums. Here the model is nonlinear only in the parameters z_1 and z_2. Given values of z_1 and z_2 the subproblem (11.2.22) is easily solved.

An important improvement of the algorithm was introduced by Kaufman in [234]. The jth column of the Jacobian of the reduced problem can be written

$$J = -\left[P_{\mathcal{N}(\Phi^T)}\frac{d\Phi}{d\alpha_j}\Phi^- + (\Phi^-)^T\frac{d\Phi^T}{d\alpha_j}P_{\mathcal{N}(\Phi^T)}\right]y.$$

Kaufman's simplification consists of using an approximate Jacobian obtained by dropping the second term in this formula. The effect is to reduce the work per iteration at the cost of marginally increasing the number of iterations. Savings up to 25% are achieved by this simplifications. This simplification was generalized to separable problems with constraints in [235].

The program VARPRO, was later modified by John Bolstad, who improved the documentation, included the modification of Kaufman and added the calculation of the covariance matrix. LeVeque later wrote a version called VARP2 which handles multiple right hand sides. Both VARPRO and VARP2 are available in the public domain in the Port Library, by David Gay[56] see also A. J. Miller[57]

Golub and LeVeque [175] extended the VARPRO algorithm to the case when several data sets are to be fitted to the model with the same nonlinear parameter vector; see also Kaufman and Sylvester [236]

[56] `http://netlib.bell.labs/netlib/master/readme.html;`
[57] `http://users.igpond.net.au/amiller.`

This variable projection approach not only reduces the dimension of the parameter space but also leads to a better conditioned problem. F. Krogh [245] notes that the variable projection algorithm solved several problems at JPL which could not be solved using the old nonlinear least squares approach.

Among later theoretical developments of variable projection methods is the paper by Ruhe and Wedin [322] They analyze several different algorithms for a more general class of separable problems. They conclude that the Gauss–Newton algorithm applied to the variable projection functional has the same asymptotic rate of convergence as when it is applied to the full problem.

We describe a standard method for solving this problem is **Prony's method**. Assume that that the function $y = f(x)$ is given in equidistant points with the coordinates (x_i, y_i), $i = 1 : m$, where $x_i = x_1 + ih$. We want to approximate these data with a function

$$q(x) = \sum_{j=1}^{n} a_j e^{\lambda_j x}. \tag{11.2.30}$$

Putting $c_j = a_j e^{\lambda_j x_1}$ and $v_j = e^{h\lambda_j}$, we obtain the linear system of equations $Mc = y$, where

$$M = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_1 & v_2 & \cdots & v_n \\ \vdots & \vdots & \cdots & \vdots \\ v_1^m & v_2^m & \cdots & v_n^m \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}. \tag{11.2.31}$$

Now assume that the unknown $v_1, \dots, v_n$ are roots to the polynomial

$$\phi(v) = (v - v_1)(v - v_2)\cdots(v - v_n) = v^n + s_1 v^{m-1} + \cdots s_m.$$

Multiplying the equations in (11.2.31) in turn by $s_n, s_{n-1}, \dots, s_1, s_0 = 1$, and adding, we obtain

$$\sum_{j=1}^{n} \phi(v_j)c_j = \sum_{j=0}^{n} s_{n-j} y_j = 0$$

since $\phi(v_j) = 0$, $j = 1 : n$. Normally n is substantially smaller than m. By shifting the origin with h we get a new equation. Repeating this we get a (usually overdetermined) system Thus we have $m - n + 1$ equations for determining the unknowns $s_n, s_{n-1}, \dots, s_1$. This can be solved by the method of least squares. Determining the roots of the polynomial $\phi(v)$ we obtain v_j and $\lambda_j = \ln v_j/h$. Finally we get c_j from the linear system (11.2.31) and $a_j = c_j e^{-\lambda_j x_1}$.

We here describe a variable projection algorithm due to Kaufman [234, 1975] which uses a Gauss–Newton method applied to the problem (11.2.26). The algorithm contains two steps merged into one. Let $x_k = (y_k, z_k)^T$ be the current approximation. The next approximation is determined as follows:

(i) Compute the solution δy_k to the linear subproblem

$$\min_{\delta y_k} \big\| f(z_k)\delta y_k - \big(g(z_k) - f(z_k)y_k\big)\big\|_2, \tag{11.2.32}$$

and put $y_{k+1/2} = y_k + \delta_k$, and $x_{k+1/2} = (y_{k+1/2}, z_k)^T$.

(ii) Compute d_k as the Gauss–Newton step at $x_{k+1/2}$, i.e., d_k is the solution to

$$\min_{d_k} \left\| C(x_{k+1/2})d_k + r(y_{k+1/2}, z_k) \right\|_2, \tag{11.2.33}$$

where the Jacobian is $C(x_{k+1/2}) = \big(f(z_k), r_z(y_{k+1/2}, z_k)\big)$. Take $x_{k+1} = x_k + \lambda_k d_k$ and go to (i).

In (11.2.33) we have used that by (11.2.24) the first derivative of r with respect to y is given by $r_y(y_{k+1/2}, z_k) = f(z_k)$. The derivatives with respect to z are given by

$$r_z(y_{k+1/2}, z_k) = B(z_k)y_{k+1/2} - g'(z_k), \qquad B(z)y = \left(\frac{\partial F}{\partial z_1}y, \dots, \frac{\partial F}{\partial z_q}y\right),$$

where $B(z)y \in \mathbf{R}^{m\times q}$. Note that in case $r(y,z)$ is linear also in y it follows from (11.2.4) that $C(x_{k+1/2}) = (f(z_k), H(y_{k+1/2}))$. To be robust the algorithms for separable problems must employ a line search or trust region approach for the Gauss–Newton steps as described in Section 11.2.3 and 11.2.4.

It can be shown that the Gauss–Newton algorithm applied to (11.2.26) has the same asymptotic convergence rate as the ordinary Gauss–Newton's method. In particular both converge quadratically for zero residual problem. This is in contrast to the naive algorithm for separable problems of alternatively minimizing $\|r(y,z)\|_2$ over y and z, which *always* converges linearly. One advantage of the Kaufman algorithm is that *no starting values for the linear parameters have to be provided.* We can, e.g., take $y_0 = 0$ and determine $y_1 = \delta y_1$, in the first step of (11.2.32). This seems to make a difference in the first steps of the iterations, and sometimes the variable projection algorithm can solve problems for which methods not using separability fail.

A review of developments and applications of the variable projection approach for separable nonlinear least squares problems is given by Golub and Pereyra [180].

Constrained nonlinear least squares; see Gulliksson, Söderkvist, and Wedin [191].

11.2.6 Orthogonal Distance Regression

Consider the problem of fitting observations (y_i, t_i), $i = 1 : m$ to a mathematical model

$$y = f(p, t). \tag{11.2.34}$$

where y and t are scalar variables and $p \in \mathbf{R}^n$ are parameters to be determined. In the classical regression model the values t_i of the independent variable are assumed to be exact and only y_i are subject to random errors. Then it is natural to minimize the sum of squares of the deviations $y_i - g(p, t_i)$. In this section we consider the more general situation, when also the values t_i contain errors.

Assume that y_i and t_i are subject to errors $\bar\epsilon_i$ and $\bar\delta_i$ respectively, so that

$$y_i + \bar\epsilon_i = f(p, t_i + \bar\delta_i), \qquad i = 1 : m,$$

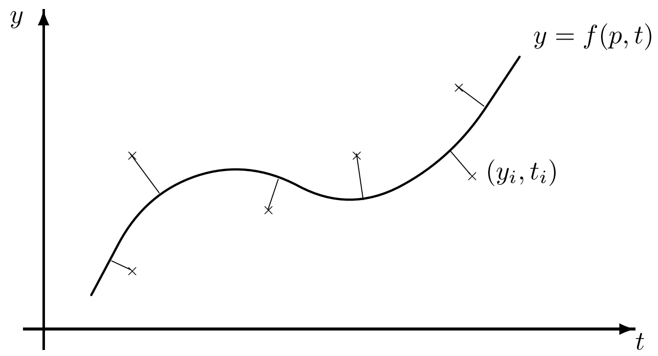

Figure 11.2.2. *Orthogonal distance regression.*

where $\bar{\epsilon}_i$ and $\bar{\delta}_i$ are independent random variables with zero mean and variance σ^2. Then the parameters p should be chosen so that the sum of squares of the **orthogonal distances** from the observations (y_i, t_i) to the curve in (11.2.34) is minimized, cf. Figure 11.2.2. Hence the parameters p should be chosen as the solution to

$$\min_{p,\epsilon,\delta} \sum_{i=1}^{m} (\epsilon_i^2 + \delta_i^2), \quad \text{subject to} \quad y_i + \epsilon_i = f(p, t_i + \delta_i), \quad i = 1 : m.$$

Eliminating ϵ_i using the constraints we arrive at the **orthogonal distance problem**

$$\min_{p,\delta} \sum_{i=1}^{m} \big(f(p, t_i + \delta_i) - y_i\big)^2 + \delta_i^2. \tag{11.2.35}$$

Note that (11.2.35) is a nonlinear least squares problem even if $f(p,t)$ is linear in p.

The problem (11.2.35) has $(m+n)$ unknowns p and δ. In applications usually $m \gg n$ and accounting for the errors in t_i will considerably increase the size of the problem. Therefore the use of standard methods will not be efficient unless the special structure is taken into account to reduce the work. If we define the residual vector $r(\delta, p) = (r_1^T(\delta, p), r_2^T(\delta))$ by

$$r_1^T(\delta, p)_i = f(p, t_i + \delta_i) - y_i, \qquad r_2^T(\delta) = \delta_i, \quad i = 1 : m,$$

the Jacobian matrix for problem (11.2.35) can be written in block form as

$$\tilde{J} = \begin{pmatrix} D_1 & J \\ \underbrace{I_m}_{m} & \underbrace{0}_{n} \end{pmatrix} \begin{matrix} \}m \\ \}m \end{matrix} \in \mathbf{R}^{2m \times (m+n)}, \tag{11.2.36}$$

where

$$D_1 = \operatorname{diag}(d_1, \ldots, d_m), \quad d_i = \left(\frac{\partial f}{\partial t}\right)_{t = t_i + \delta_i},$$

$$J_{ij} = \left(\frac{\partial f}{\partial p_j}\right)_{t=t_i+\delta_i}, \quad i = 1 : m, \quad j = 1 : n.$$

Note that $\tilde{J}$ is sparse and highly structured. In the Gauss–Newton method we compute corrections $\Delta\delta_k$ and Δp_k to the current approximations which solve the linear least squares problem

$$\min_{\Delta\delta,\Delta p} \left\| \tilde{J} \begin{pmatrix} \Delta\delta \\ \Delta p \end{pmatrix} - \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \right\|_2, \tag{11.2.37}$$

where $\tilde{J}, r_1$, and r_2 are evaluated at the current estimates of δ and p. To solve this problem we need the QR decomposition of $\tilde{J}$. This can be computed in two steps. First we apply a sequence of Givens rotations $Q_1 = G_m \cdots G_2 G_1$, where $G_i = R_{i,i+m}$, $i = 1 : m$, to zero the (2,1) block of $\tilde{J}$:

$$Q_1 \tilde{J} = \begin{pmatrix} D_2 & K \\ 0 & L \end{pmatrix}, \qquad Q_2 \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix},$$

where D_2 is again a diagonal matrix. The problem (11.2.37) now decouples, and Δp_k is determined as the solution to

$$\min_{\Delta p} \|L\Delta p - s_2\|_2.$$

Here $L \in \mathbf{R}^{m\times n}$, so this is a problem of the same size as that which defines the Gauss–Newton correction in the classical nonlinear least squares problem. We then have

$$\Delta\delta_k = D_2^{-1}(s_2 - K\Delta p_k).$$

So far we have assumed that y and t are scalar variables. More generally if $y \in \mathbf{R}^{n_y}$ and $t \in \mathbf{R}^{n_t}$ the problem becomes

$$\min_{p,\delta} \sum_{i=1}^{m} \Big(\|f(p, t_i + \delta_i) - y_i\|_2^2 + \|\delta_i\|_2^2 \Big).$$

The structure in this more general problem can also be taken advantage of in a similar manner.

Schwetlik and Tiller [333, 1985] use a partial Marquardt type regularization where only the Δx part of $\tilde{J}$ is regularized. The algorithm by Boggs, Byrd and Schnabel [1985] incorporates a full trust region strategy. Algorithms for the nonlinear case, based on stabilized Gauss–Newton methods, have been given by Schwetlik and Tiller [1986] and Boggs, Byrd and Schnabel [1986].

11.2.7 Fitting of Circles and Ellipses.

A special nonlinear least squares problem that arises in many areas of applications is to fit given data points to a geometrical element, which may be defined in implicit form. We have already discussed fitting data to an affine linear manifold such as a line or a plane. The problem of fitting circles, ellipses, spheres, and cylinders arises in applications such as computer graphics, coordinate meteorology, and statistics.

Least squares algorithms to fit an by $f(x, y, p)$ implicitly defined curve in the x-y plane can be divided into two classes. In the first, called **algebraic fitting**, a least squares functional is used, which directly involves the function $f(x, y, p) = 0$ to be fitted, If (x_i, y_i), $i = 1 : n$ are given data points we minimize the functional

$$\Phi(p) = \sum_{i=1}^{m} f^2(x_i, y_i, p).$$

The second method, geometric fitting, minimizes a least squares functional involving the geometric distances from the data points to the curve; cf. orthogonal distance regression. Often algebraic fitting leads to a simpler problem, in particular when f is linear in the parameters p.

We first discuss algebraic fitting of circles. A circle has three degrees of freedom and can be represented algebraically by

$$f(x, y, p) = a\ (\,x\ y\,) \begin{pmatrix} x \\ y \end{pmatrix} + (b_1\ b_2) \begin{pmatrix} x \\ y \end{pmatrix} + c = 0.$$

We define a parameter vector p and an $m \times 4$ matrix S with rows s_i^T by

$$p = (a, b_1, b_2, c)^T, \qquad s_i^T = (x_i^2 + y_i^2, x_i, y_i, 1). \tag{11.2.38}$$

The problem can now be formulated as

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|p\|_2 = 1.$$

Note that the p is defined only up to a constant multiple, which is why the constraint is required. The solution equals the right singular vector corresponding to the smallest singular value of S. When p is known the center z and radius ρ of the circle can be obtained from

$$z = -\frac{1}{2a} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \qquad \rho = \frac{1}{2a}\sqrt{\|b\|_2^2 - 4ac}. \tag{11.2.39}$$

We now discuss the algebraic fitting of ellipses. An ellipse in the x-y plane can be represented algebraically by

$$f(x, y, p) = (x\ y) \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + (b_1\ b_2) \begin{pmatrix} x \\ y \end{pmatrix} + c = 0. \tag{11.2.40}$$

It we define

$$p = (a_{11}, a_{12}, a_{22}, b_1, b_2, c)^T, \qquad s_i^T = (x_i^2, 2x_iy_i, y_i^2, x_i, y_i, 1), \tag{11.2.41}$$

then we have $\Phi(p) = \|Sp\|_2^2$, where S is an $m \times 6$ matrix with rows s_i^T. Obviously the parameter vector is only determined up to a constant factor. Hence, we must complete the problem formulation by including some constraint on p. Three such constraints have been considered for fitting ellipses.

(a) **SVD constraint**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|p\|_2 = 1. \tag{11.2.42}$$

The solution of this constrained problem equals the right singular vector corresponding to the smallest singular value of S.

(b) **Linear constraint**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad p^T b = 1, \tag{11.2.43}$$

where b is a fixed vector. Assuming $\|b\|_2 = 1$, which is no restriction, and let H be an orthogonal matrix such that $Hb = e_1$. Then the constraint becomes $(Hp)^T e_1 = 1$ so we can write $Sp = (SH^T)(Hp)$, where $Hp = (1 q^T)^T$. Now if we partition $SH^T = [sS_2]$ we arrive at the unconstrained problem

$$\min_q \|S_2 q + s\|_2^2, \tag{11.2.44}$$

which is a standard linear least squares problem.

(c) **Quadratic constraint**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|Bp\|_2 = 1. \tag{11.2.45}$$

Of particular interest is the choice $B = (0\ I)$. In this case, if we let $p^T = (p_1, p_2)$ the constraint can be written $\|p_2\|_2^2 = 1$, and is equivalent to a generalized total least squares problem. The solution can then be obtained as follows. First form the QR decomposition of S,

$$S = QR = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}.$$

We can now determine p_2 from the SVD of S and then p_1 from back-substitution in $R_{11} p_1 = -R_{12} p_2$.

It should be stressed that the different constraints above can lead to very different solutions, unless the errors in the fit are small. One desirable property of the fitting algorithm is that when the data is translated and rotated the fitted ellipse should be transformed in the same way. It can be seen that to lead to this kind of invariance the constraint must involve only symmetric functions of the eigenvalues of the matrix A.

The disadvantage of the SVD constraint is its non-invariance under translation and rotations. In case of a linear constraint the choice $b^T = (1\ 0\ 1\ 0\ 0\ 0)$, which corresponds to

$$\mathrm{trace}(A) = a_{11} + a_{22} = \lambda_1 + \lambda_2 = 1. \tag{11.2.46}$$

gives the desired invariance. This constraint, attributed to Bookstein,

$$\|A\|_F^2 = a_{11}^2 + 2a_{12}^2 + a_{22}^2 = \lambda_1^2 + \lambda_2^2 = 1. \tag{11.2.47}$$

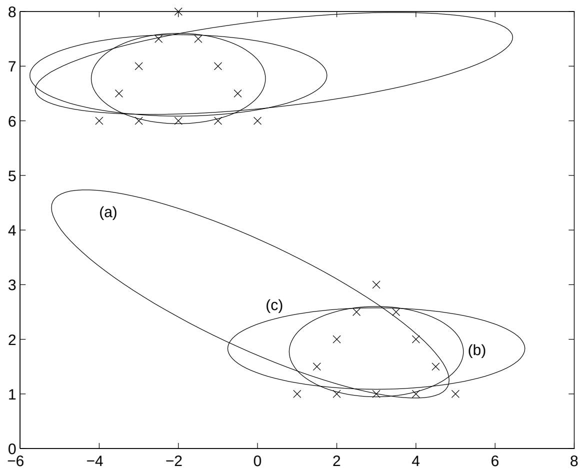

Figure 11.2.3. *Ellipse fits for triangle and shifted triangle data: (a) SVD constraint; (b) Linear constraint $\lambda_1 + \lambda_2 = 1$; (c) Bookstein constraint $\lambda_1^2 + \lambda_2^2 = 1$.*

also leads to this kind of invariance. Note that the Bookstein constraint can be put in the form $(0\ I)$ by permuting the variables and scaling by $\sqrt{2}$.

To construct and plot the ellipse it is convenient to convert the algebraic form (11.2.40) to the parametric form

$$\begin{pmatrix} x(\theta) \\ y(\theta) \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + Q(\alpha) \begin{pmatrix} a\,\cos(\theta) \\ b\,\sin(\theta) \end{pmatrix}, \quad Q(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}. \quad (11.2.48)$$

The new parameters (x_c, y_c, a, b, α) can be obtained from the algebraic parameters p. The eigendecomposition $A = Q\Lambda Q^T$, where A is the 2×2 matrix in (11.2.40) can be obtained by a Jacobi rotation, see Section 10.4.1. We assume that $a_{12} = 0$ since otherwise $Q = I$ and $\Lambda = A$ is the solution. To determine Λ and Q we first compute

$$\tau = (a_{22} - a_{11})/(2a_{12}), \qquad \tan\alpha = t = \operatorname{sign}(\tau)/\big(|\tau| + \sqrt{1+\tau^2}\big).$$

The elements in Q and Λ are then given by

$$\cos\alpha = 1/\sqrt{1+t^2}, \qquad \sin\alpha = t\,\cos\alpha,$$
$$\lambda_1 = a_{11} - t\,a_{12}, \qquad \lambda_2 = a_{22} + t\,a_{12}.$$

If we introduce the new coordinates $z = Q\tilde{z} + s$ in the algebraic form (11.2.40) this equation becomes

$$\tilde{z}^T \Lambda \tilde{z} + (2As + b)^T Q\tilde{z} + (As + b)^T s + c = 0.$$

Here s can be chosen so that this equation reduces to

$$\lambda_1 \tilde{x}^2 + \lambda_2 \tilde{y}^2 + \tilde{c} = 0.$$

Hence the center s equals

$$s = \begin{pmatrix} x_c \\ y_c \end{pmatrix} = -\frac{1}{2}A^{-1}b = -\frac{1}{2}A^{-1}Q\Lambda^{-1}(Q^T b), \tag{11.2.49}$$

and the axis (a, b) of the ellipse are given by

$$\begin{pmatrix} a \\ b \end{pmatrix} = \sqrt{-\tilde{c}}\,\operatorname{diag}\Lambda^{-1/2}, \qquad \tilde{c} = c + \frac{1}{2}b^T s = -\frac{1}{2}\tilde{b}^T\Lambda^{-1}\tilde{b}. \tag{11.2.50}$$

In geometric fitting of data (x_i, y_i), $i = 1 : m$ to a curve of the form $f(x, y, p) = 0$, where the orthogonal distance $d_i(p)$ is first measured from each data point to the curve, where

$$d_i^2(p) = \min_{f(x,y,p)=0} \big((x - x_i)^2 + (y - y_i)^2\big).$$

Then the problem

$$\min_p \sum_{i=1}^m d_i^2(p)$$

is solved. This is similar to orthogonal distance regression described for an explicitly defined function $y = f(x, \beta)$ in Section 11.2.6. Algorithms for **geometric fitting** are described in Gander, Golub, and Strebel [145, 1994].

For implicitly defined functions the calculation of the distance function $d_i(p)$ is more complicated than for explicit functions. When the curve admits a parametrization as in the case of the ellipse the minimization problem for each point is only one-dimensional.

We consider first the orthogonal distance fitting of a circle written in parametric form

$$f(x, y, p) = \begin{pmatrix} x - x_c - r\cos\phi \\ y - y_c - r\sin\phi \end{pmatrix} = 0, \tag{11.2.51}$$

where $p = (x_c, y_c, r)^T$. The problem can be written as a nonlinear least squares problem

$$\min_{p,\phi_i} \|r(p, \phi)\|_2^2, \quad \phi = (\phi_1, \dots, \phi_m), \tag{11.2.52}$$

where

$$r = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} \in \mathbf{R}^{2m}, \quad r_i = \begin{pmatrix} x_i - x_c - r\cos\phi_i \\ y_i - y_c - r\sin\phi_i \end{pmatrix}.$$

We have $2m$ nonlinear equations for $m+3$ unknowns $\phi_1, \dots, \phi_m$ and x_c, y_c, r. (Note that at least 3 points are needed to define a circle.)

We now show how to construct the Jacobian matrix, which should be evaluated at the current approximations to the $m + 3$. parameters. We need the partial derivatives

$$\frac{\partial r_i}{\partial \phi_i} = r\begin{pmatrix} \sin\phi_i \\ -\cos\phi_i \end{pmatrix}, \qquad \frac{\partial r_i}{\partial r} = -\begin{pmatrix} \cos\phi_i \\ \sin\phi_i \end{pmatrix},$$

and

$$\frac{\partial r_i}{\partial x_c} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \qquad \frac{\partial r_i}{\partial y_c} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

After reordering the rows the Jacobian associated with this problem has the form

$$J = \begin{pmatrix} rS & A \\ \underbrace{-rC}_{m} & \underbrace{B}_{3} \end{pmatrix} \begin{matrix} \}m \\ \}m \end{matrix},$$

where

$$S = \mathrm{diag}\,(\sin\phi_i), \qquad C = \mathrm{diag}\,(\cos\phi_i), \tag{11.2.53}$$

are two $m \times m$ diagonal matrices. Here the first block column, which corresponds to the m parameters ϕ_i, is orthogonal. Multiplying from the left with an orthogonal matrix we obtain

$$Q^T J = \begin{pmatrix} rI & SA - CB \\ 0 & CA + SB \end{pmatrix}, \qquad Q = \begin{pmatrix} S & C \\ -C & S \end{pmatrix}.$$

To obtain the QR factorization of J we only need to compute the QR factorization of the $m \times 3$ matrix $CA + SB$.

A Gauss–Newton type method with a trust region strategy can be implemented using this QR decomposition of the Jacobian. Good starting values for the parameters may often be obtained using an algebraic fit as described in the previous section. Experience shows that the amount of computation involved in a geometric fit is at least an order of magnitude more than for an algebraic fit.

For the geometric fit of an ellipse we use the parametric form

$$f(x, y, p) = \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} - Q(\alpha) \begin{pmatrix} a\cos\phi \\ b\sin\phi \end{pmatrix} = 0. \tag{11.2.54}$$

where $p = (x_c, y_c, a, b, \alpha)^T$ and

$$Q(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}.$$

The problem can be written as a nonlinear least squares of the form (11.2.52), where

$$r_i = \begin{pmatrix} x_i - x_c \\ y_i - y_c \end{pmatrix} - Q(\alpha) \begin{pmatrix} a\cos\phi_i \\ b\sin\phi_i \end{pmatrix}.$$

We thus have $2m$ nonlinear equations for $m+5$ unknowns $\phi_1, \dots, \phi_m$ and x_c, y_c, a, b, α. To construct the Jacobian we need the partial derivatives

$$\frac{\partial r_i}{\partial \phi_i} = Q(\alpha) \begin{pmatrix} -a\sin\phi_i \\ b\cos\phi_i \end{pmatrix}, \qquad \frac{\partial r_i}{\partial \alpha} = -\frac{d}{d\alpha} Q(\alpha) \begin{pmatrix} a\cos\phi_i \\ b\sin\phi_i \end{pmatrix},$$

and

$$\frac{\partial r_i}{\partial a} = -Q(\alpha) \begin{pmatrix} \cos\phi_i \\ 0 \end{pmatrix}, \qquad \frac{\partial r_i}{\partial b} = -Q(\alpha) \begin{pmatrix} 0 \\ \sin\phi_i \end{pmatrix}.$$

Note that

$$\frac{d}{d\alpha}Q(\alpha) = \begin{pmatrix} -\sin\alpha & \cos\alpha \\ -\cos\alpha & -\sin\alpha \end{pmatrix} = Q\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

After a reordering of the rows the Jacobian associated with this problem has the form

$$J = U\begin{pmatrix} -aS & A \\ \underbrace{bC}_{m} & \underbrace{B}_{3} \end{pmatrix} \begin{matrix} \}m \\ \}m \end{matrix}, \quad S = \operatorname{diag}(\sin\phi_i), \quad C = \operatorname{diag}(\cos\phi_i).$$

where $U = -\operatorname{diag}(Q, \dots, Q) \in \mathbf{R}^{2m\times 2m}$ is a block diagonal orthogonal matrix and S and C given by (11.2.53). The ith row of the matrices $A \in \mathbf{R}^{m\times 5}$ and $B \in \mathbf{R}^{m\times 5}$ are

$$\begin{aligned} a_i^T &= (\, -b\sin\phi_i \quad \cos\phi_i \quad 0 \quad \cos\alpha \quad \sin\alpha\,), \\ b_i^T &= (\ \ a\cos\phi_i \quad 0 \quad \sin\phi_i \quad -\sin\alpha \quad \cos\alpha\,). \end{aligned}$$

The first m columns of U^TJ can be diagonalized using a sequence of Givens rotations, where the ith rotation zeros the second component in the vector

$$\begin{pmatrix} -a\ \sin\phi_i \\ b\ \cos\phi_i \end{pmatrix}, \quad i = 1:m.$$

The fitting of a sphere or an ellipsoid can be treated analogously. The sphere can be represented in parametric form as

$$f(x, y, z, p) = \begin{pmatrix} x - x_c - r\cos\theta\cos\phi \\ y - y_c - r\cos\theta\sin\phi \\ z - z_c - r\sin\theta \end{pmatrix} = 0, \tag{11.2.55}$$

where $p = (x_c, y_c, z_c, r)^T$. We get $3m$ nonlinear equations for $2m + 4$ unknowns. The first $2m$ columns of the Jacobian matrix can simply be brought into upper triangular form; cf. Computer Exercise 2.

When the data covers only a small arc of the circle or a small patch of the sphere the fitting problem can be ill-conditioned. An important application involving this this type of data is the fitting of a spherical lens. Also the fitting of a sphere or an ellipsoid to near planar data gives rise to ill-conditioned problems.

Review Questions

1. Describe the damped Gauss–Newton method with a recommended step length procedure.
2. How does the Gauss–Newton method differ from the full Newton method? When can the behavior of the Gauss–Newton method be expected to be similar to that of Newton's method?
3. What is a separable nonlinear least squares problem? Describe a recommended method. Give an important example.

4. Consider fitting observations (y_i, t_i), $i = 1 : m$ to the model $y = g(p, t)$, where y and t are scalar variables and $p \in \mathbf{R}^n$ are parameters to be determined. Formulate the method of orthogonal distance regression for this problem.

Computer Exercises

1. One wants to fit a circle with radius r and center (x_0, y_0) to given data (x_i, y_i), $i = 1 : m$. The orthogonal distance from (x_i, y_i) to the circle

$$d_i(x_0, y_0, r) = r_i - r, \qquad r_i = \Big((x_i - x_0)^2 + (y_i - y_0)^2\Big)^{1/2},$$

depends nonlinearly on the parameters x_0, y_0. The problem

$$\min_{x_0,y_0,r} \sum_{i=1}^{m} d_i^2(x_0, y_0, r)$$

is thus a nonlinear least squares problem. An approximative linear model is obtained by writing the equation of the circle $(x - x_0)^2 + (y - y_0)^2 = r^2$ in the form

$$\delta(x_0, y_0, c) = 2xx_0 + 2yy_0 + c = x^2 + y^2,$$

which depends linearly on the parameters x_0, y_0 and $c = r^2 - x_0^2 - y_0^2$. If these parameters are known, then the radius of the circle can be determined by $r = (c + x_0^2 + y_0^2)^{1/2}$.

(a) Write down the overdetermined linear system $\delta_i(x_0, y_0, c) = x^2 + y^2$ corresponding to the data $(x, y) = (x_i, y_i)$, where

x_i	0.7	3.3	5.6	7.5	0.3	−1.1
y_i	4.0	4.7	4.0	1.3	−2.5	1.3

(b) Describe, preferably in the form of a MATLAB program a suitable algorithm to calculate x_0, y_0, c with the linearized model. The program should function for all possible cases, e.g., even when $m < 3$.

2. Generalize the algorithm described in Sec. 11.3.9 to fit a sphere to three-dimensional data (x_i, y_i, z_i), $i = 1 : m$.

11.3 Unconstrained Optimization

11.3.1 Optimality Conditions

Consider an unconstrained optimization problem of the form

$$\min_x \phi(x), \qquad x \in \mathbf{R}^n. \tag{11.3.1}$$

where the objective function ϕ is a mapping $\mathbf{R}^n \to \mathbf{R}$. Often one would like to find a **global minimum**, i.e., a point where $\phi(x)$ assumes its least value in some subset $\times \in \mathcal{B} \subset \mathbf{R}^n$. However, this is only possible in rather special cases and most numerical methods try to find local minima of $\phi(x)$.

Definition 11.3.1.
A point x^* is said to be a **local minimum** *of ϕ if $\phi(x^*) \le \phi(y)$ for all y in a sufficiently small neighborhood of x^*. If $\phi(x^*) < \phi(y)$ then x^* is a* **strong** *local minimum.*

Assume that the objective function ϕ is continuously differentiable at a point x with gradient vector $g(x) = \nabla\phi(x)$. The gradient vector $g(x) = \nabla\phi(x)$ is the normal to the tangent hyperplane of the multivariate function $\phi(x)$ (see Def. 11.1.3). As in the scalar case, a *necessary* condition for a point x^* to be optimal is that it satisfies the nonlinear system $g(x) = 0$.

Definition 11.3.2.
A point x^* which satisfies $g(x) = \nabla\phi(x) = 0$ is called a **stationary point**.

Definition 11.3.3.
A function $\phi :\ \mathbf{R}^n \to \mathbf{R}$ is twice continuously differentiable at x, if

$$g_{ij} = \frac{\partial}{\partial x_i}\left(\frac{\partial\phi}{\partial x_j}\right) = \frac{\partial^2\phi}{\partial x_i \partial x_j}, \qquad 1 \le i, j \le n.$$

exist and are continuous. The square matrix $H(x)$ formed by these n^2 quantities is called the **Hessian** *of $\phi(x)$,*

$$H(x) = \nabla^2\phi(x) = \begin{pmatrix} \frac{\partial^2\phi}{\partial x_1^2} & \cdots & \frac{\partial^2\phi}{\partial x_n \partial x_1} \\ \vdots & & \vdots \\ \frac{\partial^2\phi}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2\phi}{\partial x_n^2} \end{pmatrix} \in \mathbf{R}^{n\times n}. \tag{11.3.2}$$

If the gradient and Hessian exist and are continuous then the Hessian matrix is *symmetric*, i.e., $\partial^2\phi/\partial x_i\partial x_j = \partial^2\phi/\partial x_j\partial x_i$. Note that information about the Hessian is needed to determine if a stationary point corresponds to a minimum of the objective function. We have the following fundamental result.

Theorem 11.3.4.
Necessary conditions for x^* to be a local minimum of ϕ is that x^* is a stationary point, i.e., $g(x^*) = 0$, and that $H(x^*)$ is positive semi-definite. If $g(x^*) = 0$ and $H(x^*)$ positive definite then x^* is a strong local minimum.

Proof. The Taylor-series expansion of ϕ about x^* is

$$\phi(x^* + \epsilon d) = \phi(x^*) + \epsilon d^T g(x^*) + \frac{1}{2}\epsilon^2 d^T H(x^* + \epsilon\theta d)d,$$

where $0 \le \theta \le 1$, ϵ is a scalar and d a vector. Assume that $g(x^*) \ne 0$ and choose d so that $d^T g(x^*) < 0$. Then for sufficiently small $\epsilon > 0$ the last term is negligible and $\phi(x^* + \epsilon d) < \phi(x^*)$. $\square$

Note that, as in the one-dimensional case, it is possible for a stationary point to be neither a maximum or a minimum. Such a point is called a **saddle point**, and is illustrated in Figure 11.3.1.

Figure 11.3.1. *A saddle point.*

11.3.2 Steepest Descent

In many iterative methods for minimizing a function $\phi(x):\ \mathbf{R}^n \to \mathbf{R}$, a sequence of points $\{x_k\}$, $k = 0, 1, 2, \dots$ are generated from

$$x_{(k+1)} = x_k + \alpha_k d_k, \tag{11.3.3}$$

where d_k is a **search direction** and α_k a **step length**. If we put

$$f(\alpha) = \phi(x_k + \alpha_k d_k), \tag{11.3.4}$$

then $f'(0) = (d_k)^T g(x_k)$, where $g(x_k)$ is the gradient at x_k. The search direction d_k is said to be a **descent direction** if $(d_k)^T g(x_k) < 0$.

We assume in the following that d_k is normalized so that $\|d_k\|_2 = 1$. Then by the Schwarz inequality $f'(0)$ is minimized when

$$d_k = -g(x_k)/||g(x_k)||_2. \tag{11.3.5}$$

Hence the negative gradient direction is a *direction of steepest descent*, and this choice with $\lambda_k > 0$ leads to the **steepest descent method** (Cauchy, 1847). If combined with a suitable step length criteria this method is always guaranteed to converge to a stationary point.

In the steepest descent method the Hessian is not needed. Because of this the rate of convergence is only *linear*, and can be very slow. Hence this method is usually used only as a starting step, or when other search directions fail.

Example 11.3.1.

If the steepest descent method is applied to a quadratic function

$$\phi(x) = b^T x + \frac{1}{2} x^T G x,$$

where G is a symmetric positive definite matrix. Then from the analysis in Sec. 11.4.3 it follows that

$$\phi(x_{k+1}) - \phi(x^*) \approx \rho^2(\phi(x_k) - \phi(x^*)), \qquad \rho = \frac{\kappa - 1}{\kappa + 1},$$

where $\kappa = \kappa_2(G)$ is the condition number of G. For example, if $\kappa = 1000$, then $\rho^2 = (999/1001)^2 \approx 0.996$, and about 575 iterations would be needed to gain one decimal digit of accuracy!

11.3.3 Newton and Quasi-Newton Methods

Faster convergence can be achieved by making use, not only of the gradient, but also of the second derivatives of the objective function $\phi(x)$. The basic Newton method determines the new iterate x_{k+1}, by minimizing the **quadratic model** $\phi(x_k + s_k) \approx q_k(s_k)$,

$$q_k(s_k) = \phi(x_k) + g(x_k)^T s_k + \frac{1}{2} s_k^T H(x_k) s_k, \tag{11.3.6}$$

of the function $\phi(x)$ at the current iterate x_k. When the Hessian matrix $H(x_k)$ is positive definite, q_k has a unique minimizer that is obtained by taking $x_{k+1} = x_k + s_k$, where the **Newton step** s_k is the solution of the symmetric linear system

$$H(x_k)s_k = -g(x_k). \tag{11.3.7}$$

As in the case of solving a nonlinear system Newton's method needs to be modified when the initial point x_0 is not close to a minimizer. Either a line search can be included or a trust region technique used. In a line search we take the new iterate to be

$$x_{k+1} = x_k + \lambda_k d_k,$$

where d_k is a search direction and $\lambda_k > 0$ chosen so that $\phi(x_{k+1}) < \phi(x_k)$. The algorithms described in Section 11.3.2 for minimizing the univariate function $\phi(x_k + \lambda d_k)$ can be used to determine λ_k. However, it is usually not efficient to determine an accurate minimizer. Rather it is required that λ_k satisfy the two conditions

$$\phi(x_k + \lambda_k d_k) \le \phi(x_k) + \mu \lambda_k g(x_k)^T d_k, \tag{11.3.8}$$

$$\left|g(x_k + \lambda_k d_k)^T d_k\right| \le \eta \left|g(x_k)^T d_k\right|, \tag{11.3.9}$$

where μ and η are constants satisfying $0 < \mu < \eta < 1$. Typically $\mu = 0.001$ and $\eta = 0.9$ are used.

Note that the Newton step is not a descent direction if $g_k^T H(x_k)^{-1} g_k \le 0$. This situation is not likely to occur in the vicinity of a local optimum x^*, because of the positive (or at least nonnegative) definiteness of $H(x^*)$. Far away from an optimal point, however, this can happen. This is the reason for admitting the gradient as an alternative search direction—especially since there is a danger that the Newton direction will lead to a saddle point.

In the quadratic model the term $s_k^T H(x_k) s_k$ can be interpreted as the curvature of the surface $\phi(x)$ at x_k along s_k. Often $H(x_k)$ is expensive to compute, and we want to approximate this term. Expanding the gradient function in a Taylor series about x_k along a direction s_k we have

$$g(x_k + s_k) = g_k + H(x_k)s_k + \dots . \tag{11.3.10}$$

Hence the curvature can be approximated from the gradient using a forward difference approximation

$$s_k^T G_k s_k \approx \big(g(x_k + s_k) - g(x_k)\big)^T s_k.$$

In **quasi-Newton**, or variable metric methods an approximate Hessian is built up as the iterations proceed. Denote by B_k the approximate Hessian at the kth step. It is then required that B_{k+1} approximates the curvature of ϕ along $s_k = x_{k+1} - x_k$, i.e.,

$$B_{k+1}s_k = \gamma_k, \qquad \gamma_k = g(x_{k+1}) - g(x_k), \tag{11.3.11}$$

where γ_k is the change in the gradient. The first equation in (11.3.11) is the analog of the secant equation (11.3.10) and is called the **quasi-Newton condition**.

Since the Hessian matrix is symmetric, it seems natural to require also that each approximate Hessian is symmetric. The quasi-Newton condition can be satisfied by making a simple update to B_k. The Powell-Symmetric-Broyden (PSB) update is

$$B_{k+1} = B_k + \frac{r_k s_k^T + s_k r_k^T}{s_k^T s_k} - \frac{(r_k^T s_k) s_k s_k^T}{(s_k^T s_k)^2}, \tag{11.3.12}$$

where $r_k = \gamma_k - B_k s_k$. The update matrix $B_{k+1} - B_k$ is here of rank two. It can be shown that it is the unique symmetric matrix which minimizes $\|B_{k+1} - B_k\|_F$, subject to (11.3.11).

When line searches are used, practical experience has shown the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, given by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\gamma_k \gamma_k^T}{\gamma_k^T s_k},$$

to be the best update.

If the choice of step length parameter λ_k is such that $\gamma_k^T s_k > 0$, then B_{k+1} will inherit positive definiteness from B_k. Therefore it is usual to combine the BFGS formula with $B_0 = I$. The most widely used algorithms for unconstrained optimization use these techniques, when it is reasonable to store B_k as a dense matrix. Note that since the search direction will be computed from

$$B_k d_k = -g_k, \qquad k \geq 1, \tag{11.3.13}$$

this means that the first iteration of a quasi-Newton method is a steepest descent step.

If B_k is positive definite then the local quadratic model has a unique local minimum, and the search direction d_k computed from (11.3.13) is a descent direction. Therefore it is usually required that the update formula generates a positive definite approximation B_{k+1} when B_k is positive definite.

To compute a new search direction we must solve the linear system (11.3.13), which in general would require order n^3 operations. However, since the approximate Hessian B_k is a rank two modification of B_{k-1}, it is possible to solve this system more efficiently. One possibility would be to maintain an approximation to the *inverse* Hessian, using the Shermann-Morrison formula (6.2.14). Then only $O(n^2)$ operations would be needed. However, if the Cholesky factorization $B_k = L_k D_k L_k^T$ is available the system (11.3.13) can also be solved in order n^2 operations. Furthermore, the factors L_{k+1} and D_{k+1} of the updated Hessian approximation B_{k+1} can be computed in about the same number of operations that would be needed to generate B_{k+1}^{-1}. An important advantage of using the Cholesky factorization is that the positive definiteness of the approximate Hessian cannot be lost through round-off errors.

An algorithm for modifying the Cholesky factors of a symmetric positive definite matrix B was given by Gill, et al. [162, 1975]. Let $B = LDL^T$ be the Cholesky factorization of B, where $L = (l_{ij})$ is unit lower triangular and $D = \mathrm{diag}\,(d_j) > 0$ diagonal. Let $\bar{B} = B \pm vv^T$ be a rank-one modification of B. Then we can write

$$\bar{B} = LDL^T \pm vv^T = L(D \pm pp^T)L^T,$$

where p is the solution of the triangular system $Lp = v$. The Cholesky factorization $D \pm pp^T = \hat{L}\bar{D}\hat{L}^T$ can be computed by a simple recursion, and then we have $\bar{L} = L\hat{L}$. In case of a positive correction $B = B + vv^T$, the vector p and the elements of $\bar{L}$ and $\bar{D}$ can be computed in a numerical stable way using only $3n^2/2$ flops.

Review Questions

1. Consider the unconstrained optimization problem $\min_x \phi(x)$, $x \in \mathbf{R}^n$. Give necessary conditions for x^* to be a local minimum. ($\phi(x) \,:\, \mathbf{R}^n \to \mathbf{R}$ is assumed to be twice continuously differentiable.)

2. (a) In many iterative methods for minimizing a function $\phi(x)$, a sequence of points are generated from $x_{k+1} = x_k + \lambda_k d_k$, $k = 0, 1, 2, \ldots$, where d_k is a search direction. When is d_k a descent direction? Describe some strategies to choose the step length λ_k.

(b) Define the Newton direction. When is the Newton direction a descent direction?

3. In quasi-Newton, or variable metric methods an approximate Hessian is built up as the iterations proceed. Denote by B_k the approximate Hessian at the kth step. What quasi-Newton condition does B_k satisfy, and what is the geometrical significance of this condition?

4. (a) What property should the function $f(x)$ have to be unimodal in $[a, b]$?

(b) Describe an interval reduction methods for finding the minimum of a unimodal function in $[a, b]$, which can be thought of as being analogues of the bisection method.

What is its rate of convergence?

Problems

1. (a) The general form for a quadratic function is

$$\phi(x) = \frac{1}{2}x^T Gx - b^T x + c,$$

where $G \in \mathbf{R}^{n\times n}$ is a symmetric matrix and $b \in \mathbf{R}^n$ a column vector. Show that the gradient of ϕ is $g = Gx - b$ and the Hessian is G. Also show that if $g(x^*) = 0$, then

$$\phi(x) = \phi(x^*) + \frac{1}{2}(x - x^*)^T H(x - x^*).$$

(b) Suppose that G is symmetric and nonsingular. Using the result from (a) show that Newton's method will find a stationary point of ϕ in one step from an arbitrary starting point x_0. Under what condition is this a minimum point?

2. Let $\psi(x)$ be quadratic with Hessian matrix G, which need not be positive definite.
(a) Let $\psi(\lambda) = \phi(x_0 - \lambda d)$. Show using Taylor's formula that

$$\psi(\lambda) = \psi(0) - \lambda g^T d + \frac{1}{2}\lambda^2 d^T Gd.$$

Conclude that if $d^T Gd > 0$ for a certain vector d then $\psi(\lambda)$ is minimized when $\lambda = g^T d/d^T Gd$, and

$$\min_\lambda \psi(\lambda) = \psi(0) - \frac{1}{2}\frac{(d^T g)^2}{d^T Gd}.$$

(b) Using the result from (a) show that if $g^T Gg > 0$ and $g^T G^{-1} g > 0$, then the steepest descent method $d = g$ with optimal λ gives a smaller reduction of ψ than Newton's method if $g^T G^{-1} g > (g^T g)^2/g^T Gg$. (The conclusion holds also if $\phi(x_0 - \lambda d)$ can be approximated by a quadratic function of λ reasonably well in the relevant intervals.)
(c) Suppose that G is symmetric and nonsingular. Using the result from (b) show that Newton's method will find a stationary point of ϕ in one step from an arbitrary starting point x_0. Under what condition is this a minimum point?

11.4 Constrained Optimization

The nonlinear optimization problem with nonlinear inequality constrained has the following general form

$$\min_{x\in\mathbf{R}^n} c^T x \quad \text{subject to} \quad c_i(x) \ge 0, \quad i = 1:m \tag{11.4.1}$$

Equality constraints $\hat{c}_i(x) = 0$ could also be present. Nonlinear constraints are inherently difficult to treat. At a feasible point where some nonlinear constraint holds with equality its value will alter along any direction. This has a major impact on solution methods.

It is beyond the scope of this book to treat the general nonlinearly constrained problem (11.4.1). There is an important subclass of constrained problems where both the objective function and the constraints are linear. For these methods from linear algebra play an important part in the design of algorithms.

11.4.1 Optimality for Linear Inequality Constraints

Linear optimization or **linear programming** is a mathematical theory and method of calculation for determining the minimum (or maximum) of a linear objective function, where the domain of the variables are restricted by a system of linear inequalities, and possibly also by a system of linear equations. This is famous problem which has been extensively studied since the late 1940's. Problems of this type come up, e.g., in economics, strategic planning, transportation and productions problems, telecommunications, and many other applications. Important special cases arise in approximation theory, e.g., data fitting in l_1 and l_∞ norms. The number of variables in linear optimization can be very large. Today linear programs with 5 million variables are solved!

Figure 11.4.1. *Geometric illustration of a linear programming problem.*

A linear programming problem cannot be solved by setting certain partial derivatives equal to zero. As the following example shows, the deciding factor is the domain in which the variables can vary.

Example 11.4.1.

In a given factory there are three machines M_1, M_2, M_3 used in making two products P_1, P_2. One unit of P_1 occupies M_1 5 minutes, M_2 3 minutes, and M_3 4 minutes. The corresponding figures for one unit of P_2 are: M_1 1 minute, M_2 4 minutes, and M_3 3 minutes. The net profit per unit of P_1 produced is 30 dollars, and for P_2 20 dollars. What production plan gives the most profit?

Suppose that x_1 units of P_1 and x_2 units of P_2 are produced per hour. Then the problem is to maximize

$$f = 30x_1 + 20x_2$$

subject to the constraints $x_1 \geq 0$, $x_2 \geq 0$, and

$$\begin{aligned} 5x_1 + x_2 &\leq 60 \quad \text{for } M_1, \\ 3x_1 + 4x_2 &\leq 60 \quad \text{for } M_2, \\ 4x_1 + 3x_2 &\leq 60 \quad \text{for } M_3. \end{aligned} \tag{11.4.2}$$

The problem is illustrated geometrically in Figure 11.4.1. The first of the inequalities (11.4.3) can be interpreted that the solution (x_1, x_2) must lie on the left of or on the line AB whose equation is $5x_1 + x_2 = 60$. The other two can be interpreted in a similar way. Thus (x_1, x_2) must lie within or on the boundary of the pentagon $OABCD$. The value of the function f to be maximized is proportional to the orthogonal distance and the dashed line $f = 0$; it clearly takes on its largest value at the vertex B. Since every vertex is the intersection of two lines, we must have equality in (at least) two of the inequalities. At the solution x^* equality holds in the inequalities for M_1 and M_3. These two constraints are called **active** at x^*; the other are **inactive**. The active constraints give two linear equations for determining the solution, $x_1 = 120/11$, $x_2 = 60/11$. Hence the maximal profit $f = 4,800/11 = 436.36$ dollars per hour is obtained b using M_1 and M_2 continuously, while M_2 is used only $600/11 = 54.55$ minutes per hour.

A linear programming (LP) problem can more generally be stated in the following form:

$$\begin{aligned} &\min_{x \in \mathbf{R}^n} c^T x \\ &\text{subject to} \quad Ax \geq b, \quad x \geq 0. \end{aligned} \tag{11.4.3}$$

Here $x \in \mathbf{R}^n$ is the vector of unknowns, $c \in \mathbf{R}^n$ is the **cost vector**, and $A \in \mathbf{R}^{m \times n}$ the constraint matrix. The function $c^T x$ to be minimized is called the **objective** function. (Note that the problem of maximizing $c^T x$ is equivalent to minimizing $-c^T x$.)

A single linear inequality constraint has the form $a_i^T x \geq b_i$. The corresponding equality $a_i^T x = b_i$ defines a hyperplane in $\mathbf{R}^n$. The inequality restricts x to lie on the feasible side of this hyperplane. The feasible region of the LP (11.4.3) is the set

$$\mathcal{F} = \{x \in \mathbf{R}^n \mid Ax \geq b\}. \tag{11.4.4}$$

An inequality constraint is said to be **redundant** if its removal does not alter the feasible region.

Obviously, a solution to the LP (11.4.3)can exist only if $\mathcal{F}$ is not empty. When $\mathcal{F}$ is not empty, it has the important property of being a **convex set**, which is defined as follows: Let x and y be any two points in $\mathcal{F}$. Then the line segment

$$\{z \equiv (1 - \alpha)x + \alpha y \mid 0 \leq \alpha \leq 1\}$$

joining x and y is also in $\mathcal{F}$. It is simple to verify that $\mathcal{F}$ defined by (11.4.4) has this property, since

$$Az = (1 - \alpha)Ax + \alpha Ay \geq (1 - \alpha)b + \alpha b = b.$$

when $0 \le \alpha \le 1\}$.

The **active set** of the inequality constraints $Ax \ge b$ at a point x is the subset of constraints which are satisfied with equality at x. Hence the constraint $a_i^T x \ge b_i$ is active if the residual at x is zero,

$$r_i(x) = a_i^T x - b_i = 0.$$

Let x be a feasible point in $\mathcal{F}$. Then it is of interest to find directions p such that $x + \alpha p$ remains feasible for some $\alpha > 0$. If the constraint $a_i^T x \ge b_i$ is active at x, then all points $y = x + \alpha p$, $\alpha > 0$, will remain feasible with respect to this constraint if and only if $a_i^T p \ge 0$. It is not difficult to see that the feasible directions p are not affected by the inactive constraints at x. Hence p is a feasible directions at the point x if and only if $a_i^T p \ge 0$ for all active constraints at x.

Given a feasible point x the maximum step α that can be taken along a feasible direction p depends on the inactive constraints. We need to consider the set of inactive constraints i for which $a_i^T p < 0$. For these constraints, which are called decreasing constraints, $a_i^T(x+\alpha p) = a_i^T x + \alpha a_i^T p = b_i$, and thus the constraint i becomes active when

$$\alpha = \alpha_i = \frac{a_i^T x - b_i}{-a_i^T p}.$$

Hence the largest step we can take along p is $\max \alpha_i$ where we maximize over all decreasing constraints.

For an LP there are three possibilities: There may be no feasible points, in which case the LP has no solution; there may be a feasible point x^* at which the objective function is minimized; Finally, the feasible region may be unbounded and the objective function unbounded below in the feasible region. The following fundamental theorem states how these three possibilities can be distinguished:

Theorem 11.4.1.
Consider the linear program

$$\min_{x \in \mathbf{R}^n} c^T x \quad \text{subject to} \quad Ax \ge b.$$

(We assume here that the constraints $x \ge 0$ are not present.) Then the following results hold:

(a) *If no points satisfy $Ax \ge b$, the LP has no solution;*

(b) *If there exists a point x^8 satisfying the conditions*

$$Ax^* \ge b, \qquad c = A_A^T \lambda_A^*, \quad \lambda_A^* \ge 0,$$

where A_A is the matrix of active constraints at x^*, then $c^T x^*$ is the unique minimum value of $c^T x$ in the feasible region, and x^* is a minimizer.

(c) *If the constraints $Ax \ge b$ are consistent, the objective function is unbounded below in the feasible region if and only if the last two conditions in* (b) *are not satisfied at any feasible point.*

The last two conditions in (b) state that c can be written as a nonnegative linear combination of the rows in A corresponding to the active constraints. The proof of this theorem is nontrivial. It is usually proved by invoking **Farkas Ĺemma**, a classical result published in 1902. For a proof we refer to [166, Sec. 7.7].

The geometrical ideas in the introductory example are useful also in the general case. Given a set of linear constraints a **vertex** is a feasible point for which the active constraints matrix has rank n. Thus at least n constraints are active at a vertex x. A vertex is an extreme point of the feasible region $\mathcal{F}$. If exactly n constraints are active at a vertex, the vertex is said to be **nondegenerate**; if more than n constraints are active at a vertex, the vertex is said to be **degenerate**. In Example 11.4.1 there are five vertices O, A, B, C, and D, all of which are nondegenerate. The vertices form a polyhedron, or **simplex** in $\mathbf{R}^n$.

Vertices are of central importance in linear programming since many LP have the property that a minimizer lies at a vertex. The following theorem states the conditions under which this is true.

Theorem 11.4.2.

Consider the linear program of

$$\min_{x\in\mathbf{R}^n} c^T x \quad \text{subject to} \quad Ax = b, \quad x \geq 0.$$

where $A \in \mathbf{R}^{m\times n}$. *If* $\operatorname{rank}(A) = n$ *and the optimal value of* $c^T x$ *is finite, a vertex minimizer exists.*

Note that by convexity an infinity of non-vertex solutions will exist if the minimizer is not unique. For example, in a problem like Example 11.4.1, one could have an objective function $f = c^T x$ such that the line $f = 0$ were parallel to one of the sides of the pentagon. Then all points on the line segment between two optimal vertices in the polyhedron are also optimal points.

Suppose a linear program includes the constraints $x \geq 0$. Then the constraint matrix has the form

$$\begin{pmatrix} A \\ I_n \end{pmatrix} \in \mathbf{R}^{(m+n)\times n}.$$

Since the rows include the identity matrix I_n this matrix always has rank n. Hence a feasible vertex must exist if any feasible point exists.

11.4.2 Standard Form for LP

It is convenient to adopt the following **standard form** of a linear programming problem:

$$\min_{x\in\mathbf{R}^n} c^T x \quad \text{subject to} \quad Ax = b, \quad x \geq 0. \tag{11.4.5}$$

where $A \in \mathbf{R}^{m\times n}$. The constraints $x \geq 0$ are the only inequality constraints in a standard form problem. The set $\mathcal{F}$ of feasible points consists of points x that satisfy $Ax = b$ and $x \geq 0$. If $\operatorname{rank}(A) = n$ this set contains just one point if $A^{-1}b \geq 0$; otherwise it is empty. Hence in general we have $\operatorname{rank}(A) < n$.

It is simple to convert a linear programming problem to standard form. Many LP software packages apply an automatic internal conversion to standard form. The change of form involves modification of the dimensions, variables and constraints. An upper bound inequality $a^T x \le \beta$ is converted into an equality $a^T x + s = \beta$ by introducing a **slack variable** s subject to $s \ge 0$. A lower bound inequality of the form $a^T x \ge \beta$ can be changed to an upper bound inequality $(-a)^T x \le -\beta$. Thus when a linear programming problems with inequality constraints is converted to standard form, the number of variables will increase. If the original constraints are $Ax \le b$, $A \in \mathbf{R}^{m\times n}$, then the matrix in the equivalent standard form will be $(A \quad I_m)$, and the number of variables n plus m slack variables.

Example 11.4.2.

The problem in Example 11.4.1 can be brought into standard form with the help of three slack variables, x_3, x_4, x_5. We get

$$A = \begin{pmatrix} 5 & 1 & 1 & & \\ 3 & 4 & & 1 & \\ 4 & 3 & & & 1 \end{pmatrix}, \quad b = 60\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$
$$c^T = (-20 \quad -30 \quad 0 \quad 0 \quad 0) .$$

The three equations $Ax = b$ define a two-dimensional subspace (the plane in Figure 11.4.1) in the five-dimensional space of x. Each side of the pentagon $OABCD$ has an equation of the form $x_i = 0$, $i = 1 : 5$. At a vertex two of the coordinates are zero, and the rest cannot be negative.

For completeness we note that, although this is seldom used in practice, equality constraints can be converted to inequality constraints. For example, $a_i^T x = b_i$ is equivalent to the two inequality constraints $a_i^T x > b_i$ and $-a_i^T x \ge -b_i$.

The optimality conditions for an LP in standard form are as follows:

Theorem 11.4.3.

Consider the standard linear program of minimizing $c^T x$ subject to $Ax = b$ and $x \ge 0$ for which feasible points exist. Then x^* is a minimizer if and only if x^* is a feasible point and

$$c = A^T \pi^* + \eta^*, \qquad \eta^* \ge 0, \qquad \eta_i^* x_i^* = 0, \quad i = 1, \dots, n. \tag{11.4.6}$$

A vertex for a standard form problem is also called a **basic feasible point**. In case more than $n - m$ coordinates are zero at a feasible point we say that the point is a **degenerate** feasible point. A feasible vertex must exist if any feasible point exists. Since the m equality constraints are active at all feasible points, at least $n-m$ of the bound constraints must also be active at a vertex. It follows that a point x can be a vertex only if at least $n - m$ of its components are zero.

In the following we assume that there exist feasible points, and that $c^T x$ has a finite minimum. Then an eventual unboundedness of the polyhedron does not give rise to difficulties. These assumptions are as a rule satisfied in all practical problems which are properly formulated.

We have the following fundamental theorem, the validity of which the reader can easily convince himself of for $n - m \leq 3$.

Theorem 11.4.4.

For a linear programming problem in standard form some optimal feasible point is also a basic feasible point, i.e., at least $n - m$ of its coordinates are zero; equivalently at most m coordinates are strictly positive.

The standard form given above has the drawback that when variables are subject to lower and upper bounds, these bounds have to be entered as general constraints in the matrix A. Since lower and upper bounds on x can be handled much more easily, a more efficient formulation is often used where inequalities $l \leq x \leq u$ are substituted for $x \geq 0$. Then it is convenient to allow $l_i = -\infty$ and $u_i = \infty$ for some of the variables x_i. If for some j, $l_j = -\infty$ and $u_j = \infty$, x_j is said to be free, and if for some j, $l_j = u_j$, x_j is said to be fixed. For simplicity we consider in the following mainly the first standard form.

Example 11.4.3.

As a nontrivial example of the use of Theorem 11.4.2 we consider the following **transportation problem**, which is one of the most well-known problems in optimization. Suppose that a business concern has I factories which produce $a_1, a_2, \ldots, a_I$ units of a certain product. This product is sent to J consumers, who need $b_1, b_2, \ldots, b_J$ units, respectively. We assume that the total number of units produced is equal to the total need, i.e., $\sum_{i=1}^{I} a_i = \sum_{j=1}^{J} b_j$. The cost to transport one unit from producer i to consumer j equals c_{ij}. The problem is to determine the quantities x_{ij} transported so that the total cost is minimized. This problem can be formulated as a linear programming problem as follows:

$$\text{minimize} \quad f = \sum_{i=1}^{I} \sum_{j=1}^{J} c_{ij} x_{ij}$$

subject to $x_{ij} \geq 0$, and the constraints

$$\sum_{j=1}^{J} x_{ij} = a_i, \quad i = 1 : I, \qquad \sum_{i=1}^{I} x_{ij} = b_j, \quad j = 1 : J.$$

There is a linear dependence between these equations, since

$$\sum_{i=1}^{I} \sum_{j=1}^{J} x_{ij} - \sum_{j=1}^{J} \sum_{i=1}^{I} x_{ij} = 0.$$

The number of linearly independent equations is thus (at most) equal to $m = I + J - 1$. From Theorem 11.4.2 it follows that *there exist an optimal transportation* *scheme, where at most $I+J-1$ of the IJ possible routes between producer and consumer are used.* In principle the transportation problem can be solved by the simplex method described below; however, there are much more efficient methods which make use of the special structure of the equations.

Many other problems can be formulated as transportation problems. One important example is the **personnel-assignment problem**: One wants to distribute I applicants to J jobs, where the suitability of applicant i for job j is known. The problem to maximize the total suitability is clearly analogous to the transportation problem.

11.4.3 The Simplex Method

The **simplex method** was invented in 1947 by G. B. Danzig. Until the late 1980s it was the only effective algorithm for solving large linear programming problems. Later the simplex method has been rivaled by so called interior-point methods (see Section 11.4.6), but it is still competitive for many classes of problems.

The idea behind the simplex method is simple. From Theorem 11.4.2 we know that the problem is solved if we can find out which of the n coordinates x are zero at the optimal feasible point. In theory, one could consider trying all the $\binom{n}{n-m}$ possible ways of setting $n-m$ variables equal to zero, sorting out those combinations which do not give feasible points. The rest are vertices of the polyhedron, and one can look among these to find a vertex at which f is minimized. However, since the number of vertices increases exponentially with $n-m$ this is laborious even for small values of m and n.

The simplex method starts at a vertex (basic feasible point) and recursively proceeds from one vertex to an adjacent vertex with a lower value of the objective function c^Tx. The first phase in the simplex method is to determine an initial basic feasible point (vertex). In some cases an initial vertex can be trivially found (see, e.g., Example 11.4.4 below). A systematic method which can be used in more difficult situations will be described later in Section 11.4.4.

When an initial feasible point has been found, the following steps are repeated until convergence:

I. Check if the current vertex is an optimal solution. If so then stop, else continue.

II. Proceed from the current vertex to a neighboring vertex at which the value of f if possible is smaller.

Consider the standard form linear programming problem (11.4.5). At a vertex $n-m$ variables are zero. We divide the index set $I=\{1:m\}$ into two disjoint sets

$$I = B \cup N, \quad B=\{j_1,\dots,j_n\}, \quad N=\{i_1,\dots,i_{n-m}\}, \tag{11.4.7}$$

such that N corresponds to the zero variables. We call x_B **basic variables** and x_N **nonbasic variables**. If the vector x and the columns of the matrix A are split in a corresponding way, we can write the system $Ax = b$ as

$$A_B x_B = b - A_N x_N. \tag{11.4.8}$$

We start by illustrating the simplex method on the small example from the introduction.

Example 11.4.4.

In Example 11.4.2 we get an initial feasible point by taking $x_B = (x_3, x_4, x_5)^T$ and $x_N = (x_1, x_2)^T$. The corresponding splitting of A is

$$A_B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad A_N = \begin{pmatrix} 5 & 1 \\ 3 & 4 \\ 4 & 3 \end{pmatrix},$$

Putting $x_N = 0$ gives $\hat{x}_B = b = (60, 60, 60)^T$. Since $x_B \geq 0$ this corresponds to a vertex (the vertex O in Figure 11.4.1) for which $f = 0$. The optimality criterion is not fulfilled since $c_B = 0$ and $\hat{c}_N^T = c_N^T = (-30, -20) < 0$. If we choose $x_r = x_1 = \theta > 0$ then using $\hat{x}_B$ and the first column of $A_B^{-1} A_N = A_N$ we find

$$\theta_{max} = 60 \min_i \{1/5, 1/3, 1/4\} = 12.$$

Clearly a further increase in x_1 is inhibited by x_3. We now exchange these variables to get $x_B = (x_1, x_4, x_5)^T$, and $x_N = (x_3, x_2)^T$. (Geometrically this means that one goes from O to A in Figure 11.4.1.)

The new sets of basic and non-basic variables are $x_B = (x_1, x_4, x_5)^T$, and $x_N = (x_3, x_2)$. Taking $x_N = 0$ we have

$$\hat{x}_B = (12, 24, 12)^T, \qquad f = 0 + 12 \cdot 30 = 360.$$

The new splitting of A is

$$A_B = \begin{pmatrix} 5 & 0 & 0 \\ 3 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix}, \qquad A_N = \begin{pmatrix} 1 & 1 \\ 0 & 4 \\ 0 & 3 \end{pmatrix}.$$

The reduced costs $\hat{c}_N^T = (\,6 \quad -14\,)$ for non-basic variables are easily computed from (11.4.11). The optimality criterion is not satisfied. We take $x_r = x_2$ and solve $A_B b_2 = a_2$ to get $b_2 = (1/5)(1, 17, 11)^T$. We find $\theta = 5 \min(12/1, 24/17, 12/11) = 60/11$. Exchanging x_2 and x_5 we go from A to B in Figure 11.4.1. The new basic variables are

$$\hat{x}_B = (x_1, x_4, x_2) = (5/11)(24, 12, 12)^T, \qquad f = 4,800/11.$$

The non-basic variables are $x_N = (x_3, x_5)^T$, and to compute the reduced costs we must solve

$$\begin{pmatrix} 5 & 3 & 4 \\ 0 & 1 & 0 \\ 1 & 4 & 3 \end{pmatrix} d = \begin{pmatrix} -30 \\ 0 \\ -20 \end{pmatrix}.$$

We have $c_N^T = (0, 0)$ and get $\hat{c}_N^T = (d_1, d_3) = \frac{10}{11}(1 \quad 7)$. The optimality criterion is now satisfied, so we have found the optimal solution.

In older textbooks the calculations in the simplex method is usually presented in form of a **tableau**, where the whole matrix $B_N = A_B^{-1}A_N$ is updated in each step; see Problem 3. However, these formulas are costly and potentially unstable, since they do not allow for pivoting for size.

We now give a general description of the steps in the simplex method which is closer to what is used in current simplex codes. We assume that the matrix A_B in (11.4.8) is nonsingular. (This will always be the case if $\operatorname{rank}(A) = n$.) We can then express the basic variables in in terms of the nonbasic

$$x_B = \hat{x}_B - A_B^{-1}A_N x_N, \quad \hat{x}_B = A_B^{-1}b, \tag{11.4.9}$$

where $\hat{x}_B$ is obtained by solving the linear system $A_B\hat{x}_B = b$. If $\hat{x}_B \geq 0$ then $x_N = 0$ corresponds to a basic feasible point (vertex). The vector c is also split in two subvectors c_B and c_N, and using (11.4.9) we have

$$f = c^T x = c_B^T(\hat{x}_B - A_B^{-1}A_N x_N) + c_N^T x_N = c_B^T\hat{x}_B + \hat{c}_N^T x_N,$$

where

$$\hat{c}_N = c_N - A_N^T d, \qquad d = A_B^{-T}c_B. \tag{11.4.10}$$

Here d can be computed by solving the linear system

$$A_B^T d = c_B. \tag{11.4.11}$$

The components of $\hat{c}_N$ are known as the **reduced costs** for the nonbasic variables, and the process of computing them known as **pricing**. If $\hat{c}_N \geq 0$ *then* $x_N = 0$ *corresponds to an optimal point, since* f *cannot decrease when one gives one (or more) nonbasic variables positive values* (negative values are not permitted). Hence if the **optimality criterion** $\hat{c}_N \geq 0$ is satisfied, then the solution $x_B = \hat{x}_B$, $x_N = 0$ is optimal, and we can stop.

If the optimality criterion is *not* satisfied, then there is at least one non-basic variable x_r whose coefficient $\hat{c}_r$ in $\hat{c}_N$ is negative. We now determine the largest positive increment one can give x_r without making any of the basic variables negative, while holding the other non-basic variables equal to zero. Consider equation (11.4.9), and let b_r be the corresponding column of the matrix $A_B^{-1}A_N$. This column can be determined by solving the linear system

$$A_B b_r = a_r, \tag{11.4.12}$$

where a_r is the column in A_N corresponding to x_r. If we take $x_r = \theta > 0$, then $x_B = \hat{x}_B - \theta b_r$, and for any basic variable x_i we have $x_i = \hat{x}_i - \theta b_{ir}$. Hence if $b_{ir} > 0$, then x_i remains positive for $\theta = \theta_i \leq \hat{x}_i/b_{ir}$. The largest θ for which no basic variable becomes negative is given by

$$\theta = \min_i \theta_i, \qquad \theta_i = \begin{cases} \hat{x}_i/b_{ir} & \text{if } b_{ir} > 0; \\ +\infty & \text{if } b_{ir} \leq 0; \end{cases} \tag{11.4.13}$$

If $\theta = +\infty$, the object function is unbounded in the feasible region, and we stop. Otherwise there is at least one basic variable x_l that becomes zero for this value of θ. Such a variable is now interchanged with x_r, so x_r becomes a basic variable and x_l a non-basic variable. (Geometrically this corresponds to going to a neighboring vertex.) Note that the new values of the basic variables can easily be found by updating the old values using $x_i = x_i - \theta b_{ir}$, and $x_r = \theta$.

In case several components of the vector $\hat{c}_N$ are negative we have to specify which variable to choose. The so-called **textbook** strategy chooses r as the index of the most negative component in $\hat{c}_N$. This can be motivated by noting that c_r equals the reduction in the object function $f = c_B^T \hat{x}_B + \hat{c}_N^T x_N$, produced by a unit step along x_r Hence this choice leads to the largest reduction in the objective function assuming a fixed length of the step. A defect of this strategy is that it is not invariant under scalings of the matrix A. A scaling invariant strategy called the **steepest edge strategy** can lead to great gains in efficiency, see Gill, Murray, and Wright [166, 1991, Ch. 8].

It is possible that even at a vertex which is not an optimal solution one cannot increase f by exchanging a single variable without coming in conflict with the constraints. This exceptional case occurs only when one of the basic variables is zero at the same time that the non-basic variables are zero. As mentioned previously such a point is called a **degenerate vertex**. In such a case one has to exchange a non-basic variable with one of the basic variables which is zero at the vertex, and a step with $\theta = 0$ occurs. In more difficult cases, it may even be possible to make several such exchanges.

Figure 11.4.2. *Feasible points in a degenerate case.*

Example 11.4.5.

Suppose we want to maximize $f = 2x_1 + 2x_2 + 3x_3$ subject to the constraints

$$x_1 + x_3 \leq 1, \quad x_2 + x_3 \leq 1, \quad x_i \geq 0, \quad i = 1, 2, 3.$$

The feasible points form a four-sided pyramid in (x_1, x_2, x_3)-space; see Figure 11.4.2. Introduce slack variables x_4 and x_5, and take $\{x_1, x_2, x_3\}$ as non-basic variables.

This gives a feasible point since $x_1 = x_2 = x_3 = 0$ (the point O in Figure 11.4.2) satisfies the constraints. Suppose at the next step we move to point A, by exchanging x_3 and x_4. At this point the non-basic variables $\{x_1, x_2, x_4\}$ are zero but also x_5, and A is a degenerate vertex, and we have

$$\begin{aligned} x_3 &= 1 - x_1 - x_4, \\ x_5 &= x_1 - x_2 + x_4, \\ f &= 3 - x_1 + 2x_2 - 3x_4. \end{aligned}$$

The optimality condition is not satisfied, and at the next step we have to exchange x_2 and x_5, and remain at point A. In the final step we can now exchange x_1 and x_2 to get to the point B, at which

$$\begin{aligned} x_1 &= 1 - x_3 - x_4, \\ x_2 &= 1 - x_3 - x_5, \\ f &= 4 - x_3 - x_4 - 2x_5. \end{aligned}$$

The optimality criterion is fulfilled, and so B is the optimal point.

Traditionally degeneracy has been a major problem with the Simplex method. A proof that the simplex algorithm converges after a finite number of steps relies on a strict increase of the objective function in each step. When steps in which f does not increase occur in the simplex algorithm, there is a danger of **cycling**, i.e., the same sequence of vertices are repeated infinitely often, which leads to non-convergence. Techniques exist which prevent cycling by allowing slightly infeasible points, see Gill, Murray and Wright [166, 1991, Sec. 8.3.3]. By perturbing each bound by a small random amount, the possibility of a tie in choosing the variable to leave the basis is virtually eliminated.

Most of the computation in a simplex iteration is spent with the solution of the two systems of equations $A_B^T d = c_B$ and $A_B b_r = a_r$. We note that both the matrix A_B and the right hand sides c_B and a_r are often very sparse. In the original simplex method these systems were solved by recurring the inverse A_B^{-1} of the basis matrix. However, this is in general inadvisable because of lack of numerical stability.

Stable methods can be devised which store and update the LU factorization

$$P_B A_B = LU \tag{11.4.14}$$

where P_B is a permutation matrix. The initial factorization (11.4.14) is computed by Gaussian elimination and partial pivoting. The new basis matrix which results from dropping the column a_r and inserting the column a_s in the last position is a Hessenberg matrix. Special methods can therefore be used to generate the factors of the subsequent basis matrices as columns enter or leave.

From the above it is clear that the major computational effort in a simplex step is the solution of the two linear systems

$$A_B^T \hat{d} = c_B, \qquad A_B b_r = a_r, \tag{11.4.15}$$

to compute reduced costs and update the basic solution. These systems can be solved cheaply by computing a LU factorization of the matrix A_B is available. For large problems it is essential to take advantage of sparsity in A_B. In particular the initial basis should be chosen such that A_B has a structure close to diagonal or triangular. Therefore row and column permutations are used to bring A_B into such a form. Assume that a LU factorization has been computed for the initial basis. Since in each step only *one column* in A_B is changed, techniques for updating a (sparse) LU factorization play a central role in modern implementation of the simplex method.

Although the worst case behaviour of the simplex method is very poor–the number of iterations may be exponential in the number of unknowns—this is never observed in practice. Computational experience indicates that the simplex methods tends to give the exact result after about $2m$–$3m$ steps, and essentially independent of the number of variables n. Note that the number of iterations can be decreased substantially if one starts from an initial point close to an optimal feasible point. In some cases it may be possible to start from the optimal solution of a nearby problem. (This is sometimes called "a warm start".)

11.4.4 Finding an Initial Basis

It may not be trivial to decide if a feasible point exists, and if so, how to find one. Modify the problem by introducing a sufficient number of new **artificial variables** are added to the constraints in (11.4.5) to assure that an initial bases matrix A_B can be found satisfying

$$x_B = A_B^{-1} b \geq 0, \qquad x_N = 0, \qquad x = (x_B^T, x_N^T)^T.$$

By introducing large positive costs associated with the artificial variables these are driven towards zero in the initial phase of the Simplex algorithm. If a feasible point exists, then eventually all artificial variables will become non-basic variables and can be dropped. This is often called the **phase 1** in the solution of the original linear program. The following example illustrates this technique.

Example 11.4.6.

Maximize $f = x_1 - x_2$, subject to the constraints $x_i \geq 0$, $i = 1 : 5$, and

$$\begin{aligned} x_3 &= -2 + 2x_1 - x_2, \\ x_4 &= 2 - x_1 + 2x_2, \\ x_5 &= 5 - x_1 - x_2. \end{aligned}$$

Here if $x_1 = x_2 = 0$, x_3 is negative, so x_1, x_2 cannot be used as non-basic variables. It is not immediately obvious which pair of variables suffice as non-basic variables. Modify the problem by introducing a new **artificial variable** $x_6 \geq 0$, defined by the equation

$$x_6 = 2 - 2x_1 + x_2 + x_3.$$

We can now take x_4, x_5, x_6 as basic variables, and have found a feasible point for an extended problem with six variables. This problem will have the same solution as the original, if we can ensure that the artificial variable x_6 is zero at the solution. To accomplish this we modify the objective function to become

$$\bar{f} = x_1 - x_2 - Mx_6 = -2M + (1+2M)x_1 - (1+M)x_2 - Mx_3.$$

Here M is assumed to be a large positive number, much larger than other numbers in the computation. Then a positive value of x_6 will tend to make the function to be maximized quite small, which forces the artificial variable to become zero at the solution. Indeed, as soon as x_6 appears as a nonbasic variable, (this will happen if x_1 and x_6 are exchanged here) it is no longer needed in the computation, and can be deleted, since we have found an initial feasible point for the original problem.

The technique sketched above may be quite inefficient. A significant amount of time may be spent minimizing the sum of the artificial variables, and may lead to a vertex far away from optimality. We note that it is desirable to choose the initial basis so that A_B has a diagonal or triangular structure. Several such basis selection algorithms, named basis crashes, have been developed, see Bixby [35, 1992].

11.4.5 Duality

Consider the linear programming problem in standard form

$$\begin{aligned} &\min_{x\in\mathbf{R}^n} c^T x \\ &\text{subject to} \quad Ax = b, \quad x \geq 0. \end{aligned}$$

When this problem has a bounded optimal minimizer x^* The optimality conditions of Theorem 11.4.3 imply the existence of Lagrange multipliers y^* such that

$$c = A^T y^* + \eta^*, \qquad \eta^* \geq 0, \qquad \eta_i^* x_i^* = 0, \quad i = 1, \dots, n.$$

It follows that y^* satisfies the inequality constraints $y^T A \leq c^T$. This leads us to define the **dual problem** to the standard form problem as follows:

$$\begin{aligned} &\max_{y\in\mathbf{R}^m} g = y^T b \\ &\text{subject to} \quad y^T A \leq c^T. \end{aligned} \tag{11.4.16}$$

Here y are the **dual variables**. The initial problem will be called the **primal problem** and x the **primal variables**. If y satisfies the inequality in (11.4.16) y is called a feasible point of the dual problem. Note that the constraint matrix for the dual problem is the transposed constraint matrix of the primal, the right-hand side in the dual is the normal vector of the primal objective, and the normal vector of the dual objective is the right-hand side of the primal.

Note that the dual to a standard form linear programming problem is in all inequality form. However, the dual problem may also be written in standard form

$$\begin{aligned} &\max_{y\in\mathbf{R}^m} g = y^T b \\ &\text{subject to} \quad A^T y + z = c, \qquad z \geq 0, \end{aligned} \tag{11.4.17}$$

where z are the dual slack variables. The solution y^* to the dual problem is the Lagrange multiplier for the m linear equality constraints in the primal problem. The primal solution x^* is the Lagrange multiplier for the n linear equality constraints of the standard-form dual problem.

Let x and y be arbitrary feasible vectors for the primal and dual problems, respectively. Then

$$g(y) = y^T b = y^T Ax \leq c^T x = f(x). \tag{11.4.18}$$

The nonnegative quantity

$$c^T x - y^T b = x^T z$$

is called the **duality gap**. We will show it is zero if and only if x and y are optimal for the primal and dual.

Theorem 11.4.5.

The optimal values of the primal and dual problem are equal, i.e.,

$$\max g(y) = \min f(x). \tag{11.4.19}$$

The minimum value is obtained at a point $\hat{y}$ which is the solution of the m simultaneous equations

$$\hat{y}^T a_i = c_i, \quad i \in S, \tag{11.4.20}$$

where the set S is the set of integers defined previously.

Proof. By (11.4.18) it holds that

$$\max g(y) \leq \min f(x). \tag{11.4.21}$$

We shall show that $\hat{y}$ as defined by (11.4.20) is a feasible vector. Since $Ax = b$, we may write

$$f(x) = c^T x - \hat{y}^T (Ax - b) = \hat{y}^T b + (c^T - \hat{y}^T A)x.$$

Hence by (11.4.20)

$$f(x) = y^T b + \sum_{j \notin S} (c_j - \hat{y}^T a_j) x_j. \tag{11.4.22}$$

Now $f(x)$ is expressed in terms of the nonbasic variables corresponding to the optimal solution of the primal. It then follows form the optimality criterion (see Section 11.4.3) that $c_j - \hat{y}^T a_j \geq 0$, $j \notin S$. This together with (11.4.20), shows that $\hat{y}$ is a feasible point for the dual. Moreover, since $\hat{x}_j = 0$, $j \notin S$, then by (11.4.22) $f(\hat{x}) = \hat{y}^T b = g(\hat{y})$. This is consistent with (11.4.21) only if $\max g(y) = g(\hat{y})$. Hence $\max g(y) = \min f(x)$, and the theorem is proved. $\square$

A linear program initially given in the inequality form (11.4.16)–(11.4.18) can be converted to standard form by adding n slack variables. If the simplex method is used to solve this standard problem, each step involves solution of a linear system of sizes $n \times n$. If n is large it may be advantageous to switch instead to the primal problem, which is already in standard form. A simplex step for this problem involves solving linear systems of size $m \times m$, which may be much smaller size!

11.4.6 Barrier Functions and Interior Point Methods

Interior point methods for nonlinear optimization problems were introduced by Fiacco and McCormick [129, 1968]. They are ' characterized by the property that a sequence of approximation strictly inside the feasible region are generated. They work by augmenting the minimization objective function with a logarithmic term $-\mu \log c(x)$ for each constraint $c(x) \geq 0$. Whatever value of μ the **barrier function**, i.e. the objective function plus the logarithmic terms goes to ∞ as any constraint $c(x)$ goes to zero.This makes the approximations stay in the interior of the feasible region. As the parameter μ goes to zero, the minimizer generally converges to a minimizer of the original objective function, normally on the boundary of the feasible region.

A problem with the barrier function approach is that it seems to require the solution of a sequence of unconstrained problems which become increasingly more ill-conditioned. This can be avoided by following the barrier trajectory by exploiting duality properties and solving a sequence of linear systems, as shown by Margaret Wright [393],

Interest in interior point methods for linear programming did not arise until later, since solving a *linear* problem by techniques from *nonlinear* optimization was not believed to be a competitive approach. In 1984 Karmarkar [232, 1984] published a projection method for solving linear programming problems, which was claimed to be much more efficient than the simplex method. Karmarkar's projective method passes through the interior of the polygon as opposed to the simple method, which explores the vertices of the polygon. It was soon realized that Karmarkar's method was closely related to logarithmic barrier methods. No one uses Karmarkar's projective method any more. The simplex method, which is fast in practice, is still much used. However, it is generally accepted that for solving really huge linear programs a primal-dual interior point pathfollowing method is the most efficient. This approximates what is now called the **central path**, which is what was previously known as the barrier trajectory.

Adding a logarithmic barrier to the dual linear programming problem (11.4.16) we consider the problem

$$\begin{aligned} &\text{maximize} \quad g = y^T b + \mu \sum_{j=1}^{n} \ln z_j, \\ &\text{subject to} \quad y^T A \leq c^T. \end{aligned} \tag{11.4.23}$$

The first order optimality conditions for (11.4.23) can be shown to be

$$\begin{aligned} XZe &= \mu e, \\ Ax &= b \\ A^T y + z &= c, \end{aligned} \tag{11.4.24}$$

where $e = (1, 1, \ldots, 1)^T$, and $X = \operatorname{diag}(x)$, $Z = \operatorname{diag}(z)$. Let $\mu > 0$ be a parameter (which we will let tend to zero). Note that the last two sets of equations are the primal and dual feasibility equations, and in the limit $\mu \to 0$ the first set of equations expresses the complementarity condition $y^T x = 0$.

We then have a set of (partly nonlinear) equations for the unknown variables x, y, z. If we apply Newton's method the corrections will satisfy the following system of linear equations

$$\begin{aligned} Z\delta x + X\delta z &= \mu e - XZe, \\ A\delta x &= b - Ax, \\ A^T\delta y + \delta z &= c - A^T y - z. \end{aligned} \tag{11.4.25}$$

If $Z > 0$ we can solve to get

$$\begin{aligned} AZ^{-1}XA^T\delta y &= -AZ^{-1}(\mu e - XZe) + AZ^{-1}r_D + r_P, \\ \delta z &= -A^T\delta y + r_D, \\ \delta x &= Z^{-1}(\mu e - XZe) - Z^{-1}X\delta z. \end{aligned}$$

A sparse Cholesky factorization of ADA^T, where $D = Z^{-1}X$ is a positive diagonal matrix, is the main computational cost for the solution. Note that we need not worry about feasibility. The idea is to follow a **central path** $y(\mu)$ when $\mu \to 0$.

Review Questions

1. Give the standard form for a linear programming problem. Define the terms feasible point, basic feasible point, and slack variable.

2. State the basic theorem of linear optimization (Theorem 11.4.2). Can there be more than one optimal solution? Is every optimal solution a basic feasible point?

3. Describe the simplex method. What does one do in the case of a degenerate feasible vector?

4. Give the dual problem to $\min c^T x$ subject $Ax = b$, $x \geq 0$. How are the solutions to the dual and primal problems related?

Problems

1. (a) Find the maximum of $f = x_1 + 3x_2$ subject to the constraints $x_1 \geq 0$, $x_2 \geq 0$,

$$x_1 + x_2 \leq 2, \qquad x_1 + 2x_2 \leq 2.$$

First solve the problem graphically, and then use the simplex method with $x_1 = x_2 = 0$ as initial point. In the following variants, begin at the optimal vertex found in problem (a).

(b) Find the maximum of $f = 2x_1 + 5x_2$ under the same constraints as in (a).

(c) Find the maximum of $f = x_1 + x_2$ under the same constraints as in (a).

(d) Find the maximum of $f = x_1 + 3x_2$ after changing the second constraint in (a) to $2x_1 + 2x_2 \leq 3$.

2. Suppose that there is a set of programs LP that solves linear programming problems in standard form. One wants to treat the problem to minimize $f = d^T x$, $d^T = (1, 2, 3, 4, 5, 1, 1)$, where $x_i \geq 0$, $i = 1 : 7$,

$$\begin{aligned} |x_1 + x_2 + x_3 - 4| &\leq 12 \\ 3x_1 + x_2 + 5x_4 &\leq 6 \\ x_1 + x_2 + 3x_3 &\geq 3 \\ |x_1 - x_2 + 5x_7| &\geq 1 \end{aligned}$$

Give A, b, and c in the standard form formulation in this case.

3. At each stage in the simplex method a basic variable x_l is exchanged with a certain nonbasic variable x_r. Before the change we have for each basic variable x_i a linear relation

$$x_i = b_{ir} x_r + \sum b_{ik} x_k, \qquad i \in L,$$

where the sum is taken over all nonbasic variables except x_r. If the equation for $i = l$ is used to solve for x_r we get

$$x_r = \frac{1}{b_{lr}} x_l - \sum \frac{b_{lk}}{b_{lr}} x_k.$$

If this expression is substituted in the rest of the equations we obtain after the exchange a relation of the form

$$x_i = \hat{b}_{il} x_l + \sum \hat{b}_{ik} x_k, \qquad i \neq l,$$

(even for $i = r$), where the sum is now taken over all the nonbasic variables except x_l. Express the coefficients in the new relation in terms of the old coefficients.

4. (a) Put the dual problem in normal form, defined in Section 11.4.2. (Note that there is no non-negativity condition on y.)

(b) Show that the dual problem of the dual problem is the primal problem.

Notes and References

Section 11.1

As the name implies the Gauss–Newton method was used already by Gauss [1809]. The numerical solution of nonlinear equations by the methods of Newton, Brown and Brent is discussed by Moré and Cosnard [284, 1979]. An evaluation of numerical software that solves systems of nonlinear equations is given by Hiebert [207, 1982]. Here eight different available Fortran codes are compared on a set of test problems. Of these one uses a quasi-Newton method, two Brown's method, one Brent's method, and the remaining four Powell's hybrid method, see Powell [314, 1970]. A standard treatment of continuation methods is Allgower and Georg [3, 1990]. Different aspects of automatic differentiation are discussed by Rall [317], Griewank [186] and Corliss et al. [76].

Section 11.2

A review of developments and applications of the variable projection approach for separable nonlinear least squares problems is given by Golub and Pereyra [180].

Section 11.3

A standard textbook on numerical methods for unconstrained optimization, nonlinear systems, and nonlinear least squares is Dennis and Schnabel [101, 1983]. Trust region methods are discussed by Conn, Gould and Toint [74, 2005].

Section 11.4

A classical reference on linear optimization is Danzig [86, 1965]. For nonlinear optimization the book by Luenberger [269, 1979] is a good introduction. Excellent textbooks on optimization are Gill, Murray and Wright [165, 1981], and Fletcher [133, 1987]. A very useful reference on software packages for large scale optimization is Moré and Wright [285, 1993]. See also Jorge Nocedal and Stephen J. Wright [292]

For difficult optimization problems where the gradient is not available direct search methods may be an the only option. A review of this class of methods is given in [244]. Two recent book on interior point methods for linear programming are Vanderbei [373] and Stepehn Wright [394].

Appendix A

Calculus in Vector Spaces

We shall introduce some notions and notations from the calculus in vector spaces that will be useful in this and in later chapters. A more general and rigorous treatment can be found, e.g., in Dieudonné [106]. In this book the reader may find some proofs that we omit here. There are in the literature several different notations for these matters, e.g., **multilinear mapping** notation, **tensor** notation, or, in some cases, **vector-matrix** notation. None of them seems to be perfect or easy to handle correctly in some complex situations. This may be a reason to become familiar with several notations.

A.1 Multilinear Mappings

Consider $k+1$ vector spaces $X_1, X_2, \ldots, X_k, Y$, and let $x_\nu \in X_\nu$. A function $A\colon X_1 \times X_2 \ldots \times X_k \to Y$ is called k-**linear**, if it is linear in each of its arguments x_i separately. For example, the expression $(Px_1)^T Qx_2 + (Rx_3)^T Sx_4$ defines a 4-linear function, mapping or operator (provided that the constant matrices P, Q, R, S have appropriate size). If $k = 2$ such a function is usually called **bilinear**, and more generally one uses the term **multilinear**. , , , ,

Let $X_\nu = \mathbf{R}^{n_\nu}$, $\nu = 1, 2, \ldots, k$, $Y = \mathbf{R}^m$, and let e_{j_i} be one of the basis vectors of X_i. We use *superscripts* to denote coordinates in these spaces. Let $a^i_{j_1,j_2,\ldots,j_k}$ denote the ith coordinate of $A(e_{j_1}, e_{j_2}, \ldots, e_{j_k})$. Then, because of the linearity, the ith coordinate of $A(x_1, x_2, \ldots, x_k)$ reads

$$\sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \cdots \sum_{j_k=1}^{n_k} a^i_{j_1,j_2,\ldots,j_k} x_1^{j_1} x_2^{j_2} \ldots x_k^{j_k}, \quad x_\nu \in X_\nu. \tag{A.1.1}$$

We shall sometimes use the **sum convention** of tensor analysis; if an index occurs both as a subscript and as a superscript, the product should be summed over the range of this index, i.e., the ith coordinate of $A(x_1, x_2, \ldots, x_k)$ reads shorter $a^i_{j_1,j_2,\ldots,j_k} x_1^{j_1} x_2^{j_2} \ldots x_k^{j_k}$. (Remember always that the superscripts are no exponents.)

Suppose that $X_i = X$, $i = 1, 2, \ldots, k$. Then, the set of k-linear mappings from X^k to Y is itself a linear space called $L_k(X, Y)$. For $k = 1$, we have the space of linear functions, denoted more shortly by $L(X, Y)$. Linear functions can, of course, also be described in vector-matrix notation; $L(\mathbf{R}^n, \mathbf{R}^m) = \mathbf{R}^{m\times n}$, the set of matrices defined in Section 6.2. Matrix notation can also be used for each coordinate of a bilinear function. These matrices are in general unsymmetric.

Norms of multilinear operators are defined analogously to subordinate matrix norms. For example,

$$\|A(x_1, x_2, \dots, x_k)\|_\infty \le \|A\|_\infty \|x_1\|_\infty \|x_2\|_\infty \dots \|x_k\|_\infty,$$

where

$$\|A\|_\infty = \max_{i=1}^{m} \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \dots \sum_{j_k=1}^{n_k} |a^i_{j_1, j_2, \dots, j_k}|. \tag{A.1.2}$$

A multilinear function A is called *symmetric*, if $A(x_1, x_2, ..., x_k)$ is symmetric with respect to its arguments. In the cases mentioned above, where matrix notation can be used, the matrix becomes symmetric, if the multilinear function is symmetric.

We next consider a function $f: X \to Y$, not necessarily multilinear, where X and Y are normed vector spaces. This function is *continuous*, at the point $x_0 \in X$ if $\|f(x) - f(x_0)\| \to 0$ as $x \to x_0$, (i.e. as $\|x - x_0\| \to 0$). The function f satisfies a **Lipschitz condition** in a domain $D \subset X$, if a constant α, called a *Lipschitz constant*, can be chosen so that $\|f(x') - f(x'')\| \le \alpha \|x' - x''\|$ for all points x', $x'' \in D$.

The function f is *differentiable* at x_0, in the sense of Fréchet, if there exists a *linear* mapping A such that

$$\|f(x) - f(x_0) - A(x - x_0)\| = o(\|x - x_0\|), \quad x \to x_0.$$

This linear mapping is called the **Fréchet derivative** of f at x_0, and we write $A = f'(x_0)$ or $A = f_x(x_0)$. Note that (the value of) $f'(x_0) \in L(X, Y)$. (Considered as a function of x_0, $f'(x_0)$ is, of course, usually non-linear.)

These definitions apply also to infinite dimensional spaces. In the finite dimensional case, the Fréchet derivative is represented by the **Jacobian** matrix, the elements of which are the partial derivatives $\partial f^i / \partial x^j$, also written f^i_j, in an established notation, e.g., in tensor analysis; superscripts for coordinates and subscripts for partial derivation. If vector-matrix notation is used, it is important to note that the derivative g' of a real-valued function g is a *row* vector, since

$$g(x) = g(x_0) + g'(x_0)(x - x_0) + o(\|x - x_0\|).$$

We suggest that the notation gradient, or $\operatorname{grad} g$ is used for the transpose of $g'(x)$.

A *differential* reads, in the multilinear mapping notation, $df = f' dx$ or $df = f_x dx$. In tensor notation with the sum convention, it reads $df^i = f^i_j dx^j$.

Many results from elementary calculus carry over to vector space calculus, such as the rules for the differentiation of products. The proofs are in principle the same.

If $z = f(x, y)$ where $x \in \mathbf{R}^k$, $y \in \mathbf{R}^l$, $z \in \mathbf{R}^m$ then we define *partial derivatives* f_x, f_y with respect to the *vectors* x, y by the differential formula

$$df(x, y) = f_x dx + f_y dy, \quad \forall dx \in \mathbf{R}^k, \quad dy \in \mathbf{R}^l. \tag{A.1.3}$$

If x, y are functions of $s \in \mathbf{R}^n$, then a general version of the *chain rule* reads

$$f'(x(s), y(s)) = f_x x'(s) + f_y y'(s). \tag{A.1.4}$$

The extension to longer chains is straightforward. These equations can also be used in infinite dimensional spaces.

Consider a function $f\colon \mathbf{R}^k \to \mathbf{R}^k$, and consider the equation $x = f(y)$. By formal differentiation, $dx = f'(y)dy$, and we obtain $dy = \big(f'(y)\big)^{-1}dx$, provided that the Jacobian $f'(y)$ is non-singular. In Section 13.2.4, we shall see sufficient conditions for the solvability of the equation $x = f(y)$, so that it defines, in some domain, a differentiable *inverse function of* f, such that $y = g(x)$, $g'(x) = \big(f'(y)\big)^{-1}$.

Another important example: if $f(x, y) = 0$ then, by (A.1.4), $f_x dx + f_y dy = 0$. If $f_y(x_0, y_0)$ is a non-singular matrix, then, by the *implicit function theorem* (see Dieudonné [106, Section 10.2]) y becomes, under certain additional conditions, a differentiable function of x in a neighborhood of (x_0, y_0), and we obtain $dy = -(f_y)^{-1} f_x dx$, hence $y'(x) = -(f_y)^{-1} f_x|_{y=y(x)}$.

One can also show that

$$\lim_{\epsilon \to +0} \frac{f(x_0 + \epsilon v) - f(x_0)}{\epsilon} = f'(x_0)v.$$

There are, however, functions f, where such a *directional derivative* exists for any v but, for some x_0, is not a linear function of v. An important example is $f(x) = \|x\|_\infty$, where $x \in \mathbf{R}^n$. (Look at the case $n = 2$.) The name *Gateaux derivative* is sometimes used in such cases, in order to distinguish it from the Fréchet derivative $f'(x_0)$ previously defined.

If $f'(x)$ is a differentiable function of x at the point x_0, its derivative is denoted by $f''(x_0)$. This is a linear function that maps X into the space $L(X, Y)$ that contains $f'(x_0)$, i.e., $f''(x_0) \in L(X, L(X, Y))$. This space may be identified in a natural way with the space $L_2(X, Y)$ of bilinear mappings $X^2 \to Y$; if $A \in L(X, L(X, Y))$ then the corresponding $\bar{A} \in L_2(X, Y)$ is defined by $(Au)v = \bar{A}(u, v)$ for all $u, v \in X$; in the future it is not necessary to distinguish between A and $\bar{A}$. So,

$$f''(x_0)(u, v) \in Y, \qquad f''(x_0)u \in L(X, Y), \qquad f''(x_0) \in L_2(X, Y).$$

It can be shown that $f''(x_0)$: $X^2 \to Y$, *is a symmetric bilinear mapping*, i.e. $f''(x_0)(u, v) = f''(x_0)(v, u)$. The second order partial derivatives are denoted f_{xx}, f_{xy}, f_{yx}, f_{yy}. One can show that

$$f_{xy} = f_{yx}.$$

If $X = \mathbf{R}^n$, $Y = \mathbf{R}^m$, $m > 1$, $f''(x_0)$ reads $f^p_{ij}(x_0) = f^p_{ji}(x_0)$ in tensor notation. It is thus characterized by a three-dimensional array, which one rarely needs to store or write. Fortunately, most of the numerical work can be done on a lower level, e.g., with directional derivatives. For each fixed value of p we obtain a symmetric $n \times n$ matrix, named the **Hessian** matrix $H(x_0)$; note that $f''(x_0)(u, v) = u^T H(x_0)v$. The Hessian can be looked upon as the derivative of the gradient. An element of this Hessian is, in the multilinear mapping notation, the pth coordinate of the vector $f''(x_0)(e_i, e_j)$.

We suggest that the vector-matrix notation is replaced by the multilinear mapping formalism when handling derivatives of vector-valued functions of order higher than one. The latter formalism has the further advantage that it can be used also in infinite-dimensional spaces (see Dieudonné [106]). In finite dimensional spaces the tensor notation with the summation convention is another alternative.

Similarly, higher derivatives are recursively defined. If $f^{(k-1)}(x)$ is differentiable at x_0, then its derivative at x_0 is denoted $f^{(k)}(x_0)$ and called the kth derivative of f at x_0. One can show that $f^{(k)}(x_0) : X^k \to Y$ is a *symmetric* k-linear mapping. **Taylor's formula** then reads, when $a,\, u \in X,\, f : X \to Y$,

$$f(a+u) = f(a) + f'(a)u + \tfrac{1}{2}f''(a)u^2 + \ldots + \frac{1}{k!}f^{(k)}(a)u^k + R_{k+1}, \quad \text{(A.1.5)}$$
$$R_{k+1} = \int_0^1 \frac{(1-t)^k}{k!} f^{(k+1)}(a+ut)dtu^{k+1};$$

it follows that

$$\|R_{k+1}\| \le \max_{0\le t\le 1} \left\| f^{(k+1)}(a+ut) \right\| \frac{\|u\|^{k+1}}{(k+1)!}.$$

After some hesitation, we here use u^2, u^k, etc. as abbreviations for the lists of input vectors (u, u), $(u, u, \ldots, u)$ etc.. This exemplifies simplifications that you may allow yourself (and us) to use when you have got a good hand with the notation and its interpretation. Abbreviations that reduce the number of parentheses often increase the clarity; there may otherwise be some risk for ambiguity, since parentheses are used around the arguments for both the usually non-linear function $f^{(k)}\colon X \to L_k(X, Y)$ and the k-linear function $f^{(k)}(x_0)\colon X^k \to Y$. You may also write, e.g., $(f')^3 = f'f'f'$; beware that you do not mix up $(f')^3$ with f'''.

The mean value theorem of differential calculus and Lagrange's form for the remainder of Taylor's formula are not true, but they can in many places be replaced by the above *integral form of the remainder*. All this holds in complex vector spaces too.

Appendix B

Guide to Literature and Software

For many readers numerical analysis is studied as an important applied subject. Since the subject is still in a dynamic stage of development, it is important to keep track of recent literature. Therefore we give in the following a more complete overview of the literature than is usually given in textbooks. We restrict ourselves to books written in English. The selection presented is, however, by no means complete and reflects a subjective choice, which we hope will be a good guide for a reader who out of interest (or necessity!) wishes to deepen his knowledge. A rough division into various areas has been made in order to facilitate searching. A short commentary introduces each of these parts. Reviews of most books of interest can be found in reference periodical *Mathematical Reviews* as well as in *SIAM Review* and *Mathematics of Computation* (see Sec. 15.8).

B.1 Guide to Literature

The literature on linear algbera is very extensive. For a theoretical treatise a classical source is Gantmacher [147, 148, 1959]. Several nonstandard topics are covered in Lancaster and Tismenetsky [249, 1985] and in two excellent volumes by Horn and Johnson [215, 1985] and [216, 1991]. A very complete and useful book on and perturbation theory and related topics is Stewart and Sun [348, 1990]. Analytical aspects are emphasized in Lax [259, 1997].

An interesting survey of classical numerical methods in linear algebra can be found in Faddeev and Faddeeva [125, 1963], although many of the methods treated are now dated. A compact, lucid and still modern presentation is given by Householder [220, 1964]. Bellman [27, 1970] is an original and readable complementary text. Marcus and Minc [270] surveys a large part of matrix theory in a compact volume.

An excellent textbook on matrix computation are Stewart [340, 1973]. The recent book [346, 1998] by the same author is the first in a new series. A book which should be within reach of anyone interested in computational linear algebra is the monumental work by Golub and Van Loan [184, 1996], which has become a standard reference. The book by Higham [211, 1995] is another indispensible source book for information about the accuracy and stability of algorithms in numerical linear algebra. A special treatise on least squares problems is Björck [40, 1996].

Two classic texts on iterative methods for linear systems are Varga [375, 1962] and Young [396, 1971]. The more recent book by Axelsson [12, 1994], also covers conjugate gradient methods. Barret et al. [23, 1994] is a compact survey of iterative methods and their implementation. Advanced methods that may be used with computers with massiv parallel processing capabilities are treated by Saad [328, 2003].

A still unsurpassed text on computational methods for the eigenvalue problem is Wilkinson [387, 1965]. Wilkinson and Reinsch [390, 1971] contain detailed discussions and programs, which are very instructive. For an exhaustive treatment of the symmetric eigenvalue problem see the classical book by Parlett [305, 1980]. Large scale eigenvalue problems are treated by Saad [327, 1992]. For an introduction to the implementation of algorithms for vector and parallel computers, see also Dongarra et al. [110, 1998]. Many important pratical details on implementation of algorithms can be found in the documentation of LINPACK and EISPACK software given in Dongarra et al. [107, 1979] and Smith et al. [336, 1976]. Direct methods for sparse symmetric positive definite systems are covered in George and Liu [153, 1981], while a more general treatise is given by Duff et al. [116, 1986].

LAPACK95 is a Fortran 95 interface to the Fortran 77 LAPACK library documented in [7, 1999]. It is relevant for anyone who writes in the Fortran 95 language and needs reliable software for basic numerical linear algebra. It improves upon the original user-interface to the LAPACK package, taking advantage of the considerable simplifications that Fortran 95 allows. LAPACK95 Users' Guide [17, 2001] provides an introduction to the design of the LAPACK95 package, a detailed description of its contents, reference manuals for the leading comments of the routines, and example programs. For more information on LAPACK95 go to http://www.netlib.org/lapack95/.

B.2 Guide to Software

LINPACK and EISPACK software given in Dongarra et al. [107, 1979] and Smith et al. [336, 1976].

Specifications of the Level 2 and 3 BLAS were drawn up in 1984–88. LAPACK is a new software package designed to utilize block algorithms to achieve greater efficiency on modern high-speed computers. It also incorporates a number of algorithmic advances that have been made after LINPACK and EISPACK were written.

MATLAB, which stands for MATrix LABoratory is an interactive program specially constructed for numerical computations, in particular linear algebra. It has become very popular for computation-intensive work in research and engineering. The initial version of MATLAB was developed by Cleve Moler and built upon LINPACK and EISPACK. It now also has 2-D and 3-D graphical capabilities, and several *toolboxes* covering application areas such as digital signal processing, system identification, optimization, spline approximation, etc., are available. MATLAB is now developed by The MathWorks, Inc., USA. Future versions of Matlab will probably be built on LAPACK subroutines.

The National Institute of Standards and Technology (NIST) Guide to Available Mathematical Software (GAMS) is available for public use at the Internet URL "gams.nist.gov". GAMS is an on-line cross-index of mathematical and statistical software. Some 9000 problem-solving software modules from nearly 80 packages are indexed.

GAMS also operates as a virtual software repository, providing distribution of abstracts, documentation, and source code of software modules that it catalogs; however, rather than operate a physical repository of its own, GAMS provides transparent access to multiple repositories operated by others. Currently four repositories are indexed, three within NIST, and netlib. Both public-domain and proprietary software are indexed. Although source code of proprietary software is not redistributed by GAMS, documentation and example programs often are. The primary indexing mechanism is a tree-structured taxonomy of mathematical and statistical problems developed by the GAMS project.

Netlib is a repository of public domain mathematical software, data, address lists, and other useful items for the scientific computing community. Background about netlib is given in the article

Jack J. Dongarra and Eric Grosse (1987), Distribution of Mathematical Software Via Electronic Mail, *Comm. of the ACM*, **30**, pp. 403–407. and in a quarterly column published in the SIAM News and SIGNUM Newsletter.

Access to netlib is via the Internet URL "www.netlib.bell-labs.com" For access from Europe, use the mirror site at www.netlib.no" in Bergen,Norway. Two more mirror sites are "ukc.ac.uk" and "nchc.gov.tw".

A similar collection of statistical software is available from *statlib@temper.stat.cmu.edu*.

The TeX User Group distributes TeX-related software from *tuglib@science.utah.edu*.

The symbolic algebra system REDUCE is supported by *reduce-netlib@rand.org*.

A top level index is available for netlib, which describes the chapters of netlib, each of which has an individual index file. Note that many of these codes are designed for use by professional numerical analysts who are capable of checking for themselves whether an algorithm is suitable for their needs. One routine can be superb and the next awful. So be careful!

Below is a selective list indicating the scope of software available. The first few libraries here are widely regarded as being of high quality. The likelihood of your encountering a bug is relatively small; if you do, report be email to: ehg@research.att.com

Bibliography

[1] Jan Ole Aasen. On the reduction of a symmetric matrix to tridiagonal form. *BIT*, 11:233–242, 1971. (Cited on p. 82.)

[2] G. Allaire and S. M. Kaber. *Numerical Linear Algebra*. Springer, 2008. (Cited on p. 164.)

[3] E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*. Springer Series in Computational Mathematics, Vol. 13. Springer-Verlag, Berlin, 1990. (Cited on pp. 560, 561, 608.)

[4] G. Ammar and W. B. Gragg. Superfast solution of real positive definite Toeplitz systems. *SIAM J. Matrix Anal. Appl.*, 9(1):61–76, 1988. (Cited on p. 159.)

[5] Bjarne S. Andersen, Jerzy Waśniewski, and Fred G. Gustavson. A recursive formulation of Cholesky factorization or a packed matrix. *ACM Trans. Math. Software*, 27(2):214–244, 2001. (Cited on p. 108.)

[6] R. S. Andersen and Gene H. Golub. Richardson's non-stationary matrix iterative procedure. Technical Report STAN-CS-72-304, Computer Science Department, Stanford University, CA, 1972. (Cited on p. 512.)

[7] Edward Anderson, Zhaojun Bai, Christian Bischof, S. Blackford, James W. Demmel, Jack J. Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, A. McKenney, and D. C. Sorensen, editors. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999. (Cited on pp. 98, 451, 616.)

[8] Edward Anderssen, Zhaojun Bai, and J. J. Dongarra. Generalized QR factorization and its applications. *Linear Algebra Appl.*, 162–164:243–271, 1992. (Cited on p. 314.)

[9] Mario Arioli, James W. Demmel, and Iain S. Duff. Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Appl.*, 10(2):165–190, 1989. (Cited on pp. 125, 235.)

[10] Cleve Ashcraft, Roger G. Grimes, and John G. Lewis. Accurate symmetric indefinite linear system solvers. *SIAM J. Matrix Anal. Appl.*, 20(2):513–561, 1998. (Cited on p. 82.)

[11] Edgar Asplund. Inverses of matrices $\{a_{ij}\}$ which satisfy $a_{ij} = 0$ for $j > i + p$. *Math. Scand.*, 7:57–60, 1959. (Cited on p. 95.)

[12] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge, UK, 1994. (Cited on pp. 476, 541, 616.)

[13] R. W. Bader and Tamara G. Kolda. Efficient matlab computations with sparse and factored tensors. Technical Report, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, April 2006. (Cited on p. 276.)

[14] Zhaojun Bai and James W. Demmel. Computing the generalized singular value decomposition. *SIAM J. Sci. Comput.*, 14(6):1464–1486, 1993. (Cited on pp. 24, 448.)

[15] Zhaojun Bai, James W. Demmel, Jack J. Dongarra, Axel Ruhe, and Henk A. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, PA, 2000. (Cited on p. 452.)

[16] Zhaojun Bai and Hongyuan Zha. A new preprocessing algorithm for the computation of the generalized singular value decomposition. *SIAM J. Sci. Comput.*, 14(4):1007–1012, 1993. (Cited on p. 448.)

[17] Vincent A. Barker, L. S. Blackford, J. Dongarra, Jeremy Du Croz, Sven Hammarling, M. Marinova, Jerzy Waśniewski, and Plamen Yalamov. *LAPACK 95 Users' Guide*. SIAM, Philadelphia, PA, 2001. (Cited on p. 616.)

[18] Jesse Barlow and James W. Demmel. Computing accurate eigensystems of scaled diagonally dominant matrices. *SIAM J. Numer. Anal.*, 27:762–791, 1990. (Cited on p. 375.)

[19] Jesse Barlow and Hasan Erbay. Modifiable low-rank approximation to a matrix. *Numer. Linear Algebra Appl.*, to appear(??):??, 2008. (Cited on p. 249.)

[20] Jesse Barlow, Hasan Erbay, and Ivan Slapničar. An alternative algorithm for the refinement of ULV decompositions. *SIAM J. Matrix Anal. Appl.*, 27(1):198–214, 2005. (Cited on p. 249.)

[21] Jesse L. Barlow. More accurate bidiagonal reduction algorithm for computing the singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 23(3):761–798, 2002. (Cited on p. 314.)

[22] Jesse L. Barlow, Nela Bosner, and Zlatko Dramăk. A new stable bidiagonal reduction algorithm. *Linear Algebra Appl.*, 397:35–84, 2005. (Cited on p. 314.)

[23] R. Barret, M. W. Berry, Tony F. Chan, James W. Demmel, J. Donato, Jack Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. A. van der Vorst, editors. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, second edition, 1993. (Cited on pp. 541, 616.)

[24] Anders Barrlund. Perturbation bounds on the polar decomposition. *BIT*, 20(1):101–113, 1990. (Cited on p. 298.)

[25] Richard H. Bartels and G. W. Stewart. Algorithm 432: Solution of the equation $AX + XB = C$. *Comm. ACM*, 15:820–826, 1972. (Cited on p. 329.)

[26] F. L. Bauer. Genauigkeitsfragen bei der Lösung linearer Gleichungssysteme. *Z. Angew. Math. Mech.*, 46:7:409–421, 1966. (Cited on p. 165.)

[27] Richard Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, New York, second edition, 1970. Republished by SIAM, Philadelphia, PA, 1997. (Cited on pp. 164, 615.)

[28] Adi Ben-Israel and T. N. E. Greville. Some topics in generalized inverses of matrices. In M. Z. Nashed, editor, *Generalized Inverses and Applications*, pages 125–147. Academic Press, Inc., New York, 1976. (Cited on p. 312.)

[29] Adi Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications*. Springer-Verlag, Berlin–Heidelberg–New York, 2003. (Cited on p. 312.)

[30] Commandant Benoit. Sur la méthode de résolution des équations normales, etc. (procédés du commandant Cholesky). *Bull. Géodesique*, 2:67–77, 1924. (Cited on p. 195.)

[31] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. SIAM, Philadelphia, PA, 1994. Corrected republication, with supplement, of work first published in 1979 by Academic Press. (Cited on p. 332.)

[32] Rajendra Bhatia. *Perturbation Bounds for Matrix Eigenvalues*. Pitman Research Notes in Mathematics. Longman Scientific & Technical, Harlow, Essex, 1987. (Cited on p. 451.)

[33] Rajendra Bhatia. *Matrix Analysis*. Graduate Texts in Mathematics, 169. Springer-Verlag, New York, 1997. ISBN 0-387-94846-5. (Cited on p. 451.)

[34] David Bindel, James W. Demmel, and William Kahan. On computing Givens rotations reliably and efficiently. *ACM Trans. Math. Software*, 28(2):206–238, 12002. (Cited on p. 312.)

[35] R. E. Bixby. Implementing the simplex method: The initial basis. *ORSA J. Comput.*, 4:267–284, 1992. (Cited on p. 604.)

[36] Arne Bjerhammar. Rectangular reciprocal matrices with special reference to geodetic calculations. *Bulletin Géodésique*, 52:118–220, 1951. (Cited on p. 312.)

[37] Arne Bjerhammar. *Theory of Errors and Generalized Inverse Matrices*. Elsevier Scientific Publishing Co., Amsterdam, 1973. (Cited on p. 312.)

[38] Å. Björck. Solving linear least squares problems by Gram–Schmidt orthogonalization. *BIT*, 7:1–21, 1967. (Cited on p. 313.)

[39] Åke Björck. Numerics of Gram–Schmidt orthogonalization. *Linear Algebra Appl.*, 197–198:297–316, 1994. (Cited on p. 313.)

[40] Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA, 1996. (Cited on pp. 199, 201, 243, 274, 312, 616.)

[41] Åke Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM J. Numer. Anal.*, 8(2):358–364, 1973. (Cited on p. 418.)

[42] Åke Björck and Gene H. Golub. Iterative refinement of linear least squares solution by Householder transformation. *BIT*, 7:322–337, 1967. (Cited on p. 287.)

[43] Åke Björck and Christopher C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, 13(1):176–190, 1992. (Cited on p. 232.)

[44] Paul T. Boggs, R. H. Byrd, and R. B. Schnabel. A stable and efficient algorithm for nonlinear orthogonal regression. *SIAM J. Sci. Stat. Comput.*, 8:1052–1078, 1987. (Cited on p. 609.)

[45] A. W. Bojanczyk and R. P. Brent. Parallel solution of certain Toeplitz least-squares problems. *Linear Algebra Appl.*, 77:43–60, 1986. (Cited on p. 294.)

[46] A. W. Bojanczyk, R. P. Brent, P. Van Dooren, and F. de Hoog. A note on downdating the Cholesky factorization. *SIAM J. Sci. Stat. Comput.*, 8:210–221, 1987. (Cited on p. 314.)

[47] Z. Bothe. Bounds for rounding errors in the Gaussian elimination for band systems. *J. Inst. Maths. Applics.*, 16:133–142, 1975. (Cited on p. 91.)

[48] K. Braman, Ralph Byers, and R. Mathias. The multishift QR algorithm. Part I. Maintaining well-focused shifts and level 3 performance. *SIAM J. Matrix. Anal. Appl.*, 23(4):929–947, 2002. (Cited on p. 451.)

[49] K. Braman, Ralph Byers, and R. Mathias. The multishift QR algorithm. Part II. Aggressive early deflation. *SIAM J. Matrix. Anal. Appl.*, 23(4):948–973, 2002. (Cited on p. 451.)

[50] Richard P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973. Reprinted by Dover Publications, Mineola, NY, 2002. (Cited on p. 609.)

[51] Richard P. Brent, Colin Percival, and Paul Zimmermann. Error bounds on complex floating-point multiplication. *Math. Comp.*, posted on January 24, 2007, 2007. to appear in print. (Cited on p. 35.)

[52] Charles Broyden and Maria Theresa Vespucci. *Krylov Solvers for Linear Algebraic Systems*. Studies in Computational Mathematics 11. Elsevier, Amsterdam, 2004. (Cited on p. 541.)

[53] J. R. Bunch, Ch. P. Nielsen, and D. C. Sorenson. Rank-one modifications of the symmetric tridiagonal eigenproblem. *Numer. Math.*, 31(1):31–48, 1978. (Cited on p. 389.)

[54] James R. Bunch. Stability of methods for solving Toeplitz systems of equations. *SIAM J. Sci. Stat. Comp.*, 6(2):349–364, 1985. (Cited on p. 161.)

[55] James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Math. Comp.*, 31:163–179, 1977. (Cited on p. 81.)

[56] Angelica Bunse-Gerstner, Ralph Byers, and Volker Mehrmann. A chart of numerical methods for structured eigenvalue problems. *SIAM J. Matrix. Anal. Appl.*, 13(2):419–453, 1992. (Cited on p. 449.)

[57] Yair Censor. Row-action methods for huge and sparse systems and their applications. *SIAM Review*, 23:444–466, 1981. (Cited on p. 541.)

[58] Yair Censor and S. A. Zenios. *Parallel Optimization. Theory, Algorithms, and Applications*. Oxford University Press, Oxford, 1997. (Cited on p. 609.)

[59] J. M. Chambers. Regression updating. *J. Amer. Statist. Assoc.*, 66:744–748, 1971. (Cited on p. 285.)

[60] R. H. Chan, J. G. Nagy, and R. J. Plemmons. Generalization of Strang's preconditioner with application to Toeplitz least squares problems. *Numer. Linear Algebra Appl.*, 3(1):45–64, 1996. (Cited on p. 161.)

[61] T. Chan. An optimal circulant preconditioner for Toeplitz systems. *SIAM J. Sci. Stat. Comp.*, 9(4):766–771, 1988. (Cited on p. 161.)

[62] T. Chan. Toeplitz equations by conjugate gradients with circulant preconditioner. *SIAM J. Sci. Stat. Comp.*, 10(1):104–119, 1989. (Cited on p. 161.)

[63] Tony F. Chan. Rank revealing QR factorizations. *Linear Algebra Appl.*, 88/89:67–82, 1987. (Cited on p. 251.)

[64] Tony F. Chan and Per Christian Hansen. Some applications of the rank revealing QR factorization. *SIAM J. Sci. Stat. Comput.*, 13(3):727–741, 1992. (Cited on p. 251.)

[65] Tony F. Chan and Per Christian Hansen. Low-rank revealing QR factorizations. *Numer. Linear Algebra Appl.*, 1:33–44, 1994. (Cited on p. 249.)

[66] Shivkumar Chandrasekaran and Ilse C. F. Ipsen. Analysis of a QR algorithm for computing singular values. *SIAM J. Matrix Anal. Appl.*, 16(2):520–535, 1995. (Cited on p. 452.)

[67] Eleanor Chu and Alan George. A note on estimating the error in Gaussian elimination without pivoting. *ACM SIGNUM Newsletter*, 20:2:2–7, 1985. (Cited on p. 131.)

[68] Philippe G. Ciarlet. *Introduction to Numerical Linear Algebra and Optimization*. Cambridge University Press, Cambridge, UK, 1989. (Cited on p. 20.)

[69] Gianfranco Cimmino. Calcolo aprossimato per le soluzioni dei sistemi di equazioni lineari. *Ricerca Sci. II*, 9:326–333, 1938. (Cited on p. 504.)

[70] A. K. Cline. An elimination method for the solution of linear least squares problems. *SIAM J. Numer. Anal.*, 10(2):283–289, 1973. (Cited on p. 223.)

[71] Alan K. Cline, Cleve B. Moler, George W. Stewart, and James H. Wilkinson. An estimate for the condition number of a matrix. *SIAM J. Numer. Anal.*, 16(2):368–375, 1979. (Cited on p. 123.)

[72] R. E. Cline and Robert J. Plemmons. ℓ_2-solutions to underdetermined linear systems. *SIAM Review*, 18(1):92–106, 1976. (Cited on p. 223.)

[73] Paul Concus, Gene H. Golub, and Gérard Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.*, 6:220–252, 1985. (Cited on p. 536.)

[74] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trusr Region Methods*. SIAM Publications., Philadelphia, PA, 2000. (Cited on pp. 569, 609.)

[75] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9:251–280, 1990. (Cited on p. 164.)

[76] G. Corliss, C. Faure, Andreas Griewank, L. Hascoet, and U. Naumann. *Automatic Differentiation of Algorithm. From Simulation to Optimization*. Springer-Verlag, Berlin, 2002. (Cited on p. 608.)

[77] Anthony J. Cox and Nicholas J. Higham. Stability of Householder qr factorization for weighted least squares problems. In D. F. Griffiths, D. J. Higham, and G. A. Watson, editors, *Numerical Analysis 1997: Proceedings of the 17th Dundee Biennial Conference*, Pitman Research Notes in mathematics, vol. 380, pages 57–73. Longman Scientific and Technical, Harlow, Essex, UK, 1998. (Cited on p. 281.)

[78] Anthony J. Cox and Nicholas J. Higham. Backward error bounds for constrained least squares problems. *BIT*, 39(2):210–227, 1999. (Cited on p. 313.)

[79] C. R. Crawford. Reduction of a band-symmetric generalized eigenvalue problem. *Comm. Assos. Comput. Mach.*, 16:41–44, 1973. (Cited on p. 445.)

[80] P. D. Crout. A short method for evaluating determinants and solving systems of linear equations with real or complex coefficients. *Trans. Amer. Inst. Elec. Engng.*, 60:1235–1240, 1941. (Cited on p. 60.)

[81] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36(2):177–195, 1981. (Cited on pp. 386, 405.)

[82] Alan R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974. (Cited on p. 563.)

[83] G. Cybenko. The numerical stability of the Levinson–Durbin algorithm for Toeplitz systems of equations. *SIAM J. Sci. Stat. Comp.*, 1(3):303–309, 1980. (Cited on p. 161.)

[84] G. Dahlquist, S. C. Eisenstat, and Gene H. Golub. Bounds for the error in linear systems of equations using the theory of moments. *J. Math. Anal. Appl.*, 37:151–166, 1972. (Cited on p. 500.)

[85] G. Dahlquist, Gene H. Golub, and S. G. Nash. Bounds for the error in linear systems. In R. Hettich, editor, *Workshop on Semi-Infinite Programming*, pages 154–172, Berlin, 1978. Springer-Verlag. (Cited on p. 500.)

[86] George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, second edition, 1965. (Cited on p. 609.)

[87] Philip I. Davies, Nicholas J. Higham, and Françoise Tisseur. Analysis of the Choleski method with iterative refinement for solving the symmetric definite generalized eigenproblem. *SIAM J. Matrix Anal. Appl.*, 23(2):472–493, 2001. (Cited on p. 445.)

[88] Timothy A. Davis. *Direct Methods for Sparse Linear Systems*, volume 2 of *Fundamental of Algorithms*. SIAM, Philadelphia, PA, 2006. (Cited on p. 165.)

[89] Carl de Boor and Allan Pinkus. Backward error analysis for totally positive linear systems. *Numer. Math.*, 27:485–490, 1977. (Cited on p. 131.)

[90] F. R. de Hoog and R. M. M. Mattheij. Subset selection for matrices. *Linear Algebra Appl..*, 422:349–359, 2007. (Cited on p. 313.)

[91] A. de la Garza. An iterative method for solving systems of linear equations. Report K-731, Oak Ridge Gaseous Diffusion Plant, Oak Ridge, TN, 1951. (Cited on p. 505.)

[92] James W. Demmel, Ming Gu, Stanley Eisenstat, Ivan Slapničar, Kresimir Verselić, and Zlatko Drmač. Computing the singular value decomposition with high relative accuracy. *Linear Algebra Appl.*, 299:21–80, 1999. (Cited on p. 452.)

[93] James W. Demmel, Yozo Hida, William Kahan, Xiaoye S. Li, Sonil Mukherjee, and E. Jason Riedy. Error bounds from extra-precise iterative refinement. *ACM Trans. Math. Software*, 32(2):325–351, 2006. (Cited on p. 136.)

[94] James W. Demmel, Yozo Hida, Xiaoye S. Li, and E. Jason Riedy. Extra-precise iterative refinement for overdetermined least squares problems. Research Report, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, 2007. (Cited on p. 238.)

[95] James W. Demmel, Nicholas J. Higham, and Robert S. Schreiber. Stability of block LU factorizations. *Numer. Linear Algebra Appl.*, 2:173–190, 1995. (Cited on p. 101.)

[96] James W. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Statist. Comput.*, 11(5):873–912, 1990. (Cited on pp. 24, 343, 383, 383.)

[97] James W. Demmel and Kreŏsimir Veselić. Jacobi's method is more accurate than QR. *SIAM J. Matrix Anal. Appl.*, 13(4):1204–1245, 1992. (Cited on p. 396.)

[98] Eugene D. Denman and Alex N. Beavers. The matrix sign function and computations in systems. *Appl. Math. Comput.*, 2:63–94, 1976. (Cited on p. 416.)

[99] John E. Dennis, David M. Gay, and R. E. Welsh. An adaptive nonlinear least-squares algorithm. *ACM. Trans. Math. Software*, 7:348–368, 1981. (Cited on p. 609.)

[100] John E. Dennis and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Review*, 19:46–89, 1977. (Cited on p. 555.)

[101] John E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Number 16 in Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1995. (Cited on pp. 549, 570, 609.)

[102] Inderjit S. Dhillon. *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*. PhD thesis, University of California, Berkeley, CA, 1997. (Cited on p. 401.)

[103] Inderjit S. Dhillon. Reliable computation of the condition number of a tridiagonal matrix in $O(n)$ time. *SIAM J. Matrix Anal. Appl.*, 19(3):776–796, 1998. (Cited on p. 95.)

[104] Inderjit S. Dhillon and Beresford N. Parlett. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra Appl.*, 387:1–28, 2004. (Cited on p. 401.)

[105] Inderjit S. Dhillon and Beresford N. Parlett. Orthogonal eigenvectors and relative gaps. *SIAM J. Matrix Anal. Appl.*, 25(3):858–899, 2004. (Cited on p. 401.)

[106] J. Dieudonné. *Foundations of Modern Analysis*. Academic Press, New York, NY, 1961. (Cited on pp. 546, 611, 613, 614.)

[107] J. J. Dongarra, James R. Bunch, Cleve B. Moler, and George W. Stewart. *LINPACK Users' Guide*. SIAM, Philadelphia, PA, 1979. (Cited on pp. 97, 123, 616, 616.)

[108] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 16:1–17, 1988. (Cited on p. 98.)

[109] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. A extended set of Fortran Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 14:1–17, 1988. (Cited on p. 98.)

[110] J. J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. *Numerical Linear Algebra for High Performance Computers*. SIAM, Philadelphia, PA, 1998. (Cited on p. 616.)

[111] J. J. Dongarra, Sven Hammarling, and Danny C. Sorensen. Block reduction of matrices to condensed forms for eigenvalue computation. *J. Assos. Comput. Mach.*, 27:215–227, 1989. (Cited on p. 365.)

[112] J. J. Dongarra and Danny C. Sorensen. A fully parallel algorithmic for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8(2):139–154, 1987. (Cited on p. 386.)

[113] M. H. Doolittle. Method employed in the solution of normal equations and the adjustment of a triangularization. In *U. S. Coast and Geodetic Survey Report*, pages 115–120. 1878. (Cited on p. 60.)

[114] Jeremy Du Croz and Nicholas J. Higham. Stability of methods for matrix inversion. *IMA J. Numer. Anal.*, 12:1–19, 1992. (Cited on p. 66.)

[115] I. S. Duff. Sparse numerical linear algebra: Direct methods and preconditioning. In I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 27–62. Oxford University Press, London, UK, 1997. (Cited on p. 165.)

[116] Iain S. Duff, A. M. Erisman, and John K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, London, 1986. (Cited on pp. 165, 616.)

[117] Iain S. Duff, Roger G. Grimes, and John G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Software*, 15(1):1–14, 1989. (Cited on p. 143.)

[118] J. Durbin. The fitting of time-series models. *Rev. Internat. Statist. Inst.*, 28:229–249, 1959. (Cited on p. 159.)

[119] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrica*, 1:211–218, 1936. (Cited on p. 182.)

[120] L. Eldén. A weighted pseudoinverse, generalized singular values, and constrained least squares problems. *BIT*, 22:487–502, 1982. (Cited on pp. 245, 288.)

[121] L. Eldén. Algorithms for the computation of functionals defined on the solution of a discrete ill-posed problem. *BIT*, 30(3):466–483, 1990. (Cited on p. 245.)

[122] Lars Eldén. An efficient algorithm for the regularization of ill-conditioned least squares problems with a triangular Toeplitz matrix. *SIAM J. Sci. Stat. Comput.*, 5(1):229–236, 1984. (Cited on pp. 293, 294.)

[123] Lars Eldén and Berkant Savas. A Newton–Grassman method for computing the best multi-linear rank-(r_1, r_2, r_3) approximation of a tensor. Technical Report LITH-MAT-R-2007-6-SE, Sandia National Laboratories, Linköping University, Sweden, April 2007. (Cited on p. 276.)

[124] Jerry Eriksson. Quasi-Nerwton methods for nonlinear least squares focusing on curvature. *BIT*, 39(2):228–254, 1999. (Cited on p. 554.)

[125] D. K. Faddeev and V. N. Faddeeva. *Computational Methods of Linear Algebra*. W. H. Freeman, San Francisco, CA, 1963. (Cited on pp. 164, 615.)

[126] Ky Fan and Alan J. Hoffman. Some metric inequalities in the space of matrices. *Proc. Amer. Math. Soc.*, 6:111–116, 1955. (Cited on p. 183.)

[127] K. V. Fernando. Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices. *SIAM J. Matrix Anal. Appl.*, 20(2):373–399, 1998. (Cited on p. 395.)

[128] K. V. Fernando and Beresford N. Parlett. Accurate singular values and differential QD algorithms. *Numer. Math.*, 67:191–229, 1994. (Cited on p. 401.)

[129] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley, New York, 1968. (Cited on p. 606.)

[130] Ricardo D. Fierro and Per Christian Hansen. UTV Expansion Pack: Special-purpose rank-revealing algorithms. *Numerical Algorithms*, 40:47–66, 2005. (Cited on p. 249.)

[131] Ricardo D. Fierro, Per Christian Hansen, and P. S. Hansen. UTV Tools: Matlab templates for rank-revealing UTV decompositions. *Numerical Algorithms*, 20:165–194, 1999. (Cited on p. 249.)

[132] Roger Fletcher. Conjugate gradient methods for indefinite systems. In G. A. Watson, editor, *Numerical Analysis 1975: Proceedings of the Dundee Conference on Numerical Analysis 1975*, Lecture Notes in Mathematics Vol. 506, pages 73–89, Berlin, 1976. Springer-Verlag. (Cited on pp. 165, 520.)

[133] Roger Fletcher. *Practical Methods of Optimization*. John Wiley, New York, second edition, 1987. (Cited on p. 609.)

[134] Roger Fletcher and Danny C. Sorensen. An algorithmic derivation of the Jordan canonical form. *American Mathematical Monthly*, 90, 1983. (Cited on p. 321.)

[135] IEEE Standard for Binary Floating-Point Arithmetic. ANSI/IEEE Standard 754-1985. *SIGPLAN Notices*, 22(2):9–25, 1987. (Cited on p. 31.)

[136] Anders Forsgren, Philip E. Gill, and Margaret H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44:4:525–597, 2002. (Cited on p. 609.)

[137] George E. Forsythe. Algorithms for scientific computation. *Comm. ACM.*, 9:255–256, 1966. (Cited on p. 1.)

[138] George E. Forsythe and Peter Henrici. The cyclic Jacobi method for computing the principal values of a complex matrix. *Trans. Amer. Math. Soc.*, 94:1–23, 1960. (Cited on p. 398.)

[139] George E. Forsythe and Cleve B. Moler. *Computer Solution of Linear Algebraic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1967. (Cited on p. 64.)

[140] J. G. F. Francis. The QR transformation. Part I and II. *Computer J.*, 4:265–271, 332–345, 1961–1962. (Cited on p. 361.)

[141] Roland W. Freund, Gene H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1991. (Cited on p. 522.)

[142] Carl-Erik Fröberg. *Numerical Mathematics. Theory and Computer Applications.* Benjamin/Cummings, Menlo Park, CA, 1985. (Cited on p. 351.)

[143] Walter Gander. Least squares with a quadratic constraint. *Numer. Math.*, 36:291–307, 1981. (Cited on p. 289.)

[144] Walter Gander. Algorithms for the polar decomposition. *SIAM J. Sc. Statist. Comput.*, 11(6):1102–1115, 1990. (Cited on p. 428.)

[145] Walter Gander, Gene H. Golub, and Rolf Strebel. Least squares fitting of circles and ellipses. *BIT*, 34(4):558–578, 1994. (Cited on p. 582.)

[146] Walter Gander and Jiři Hřebiček. *Solving Problems in Scientific Computing using Maple and Matlab*. Springer-Verlag, Berlin, fourth edition, 2004. (Cited on p. 295.)

[147] F. R. Gantmacher. *The Theory of Matrices. Vol. I.* Chelsea Publishing Co, New York, 1959. (Cited on pp. 164, 182, 452, 615.)

[148] F. R. Gantmacher. *The Theory of Matrices. Vol. II.* Chelsea Publishing Co, New York, 1959. (Cited on pp. 164, 452, 615.)

[149] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and G. W. Stewart. *Matrix Eigensystems Routines: EISPACK Guide Extension*. Springer-Verlag, New York, 1977. (Cited on pp. 98, 451.)

[150] C. F. Gauss. *The Theory of the Combination of Observations Least Subject to Errors. Pars Prior.* SIAM, Philadelphia, PA, G. W. Stewart, Translation 1995, 1821. (Cited on pp. 171, 173.)

[151] Carl Friedrich Gauss. *Theory of the Motion of of the Heavenly Bodies Moving about the Sun in Conic Sections*. Dover, New York, (1963), C. H. Davis, Translation, 1809. (Cited on p. 312.)

[152] Alan George, Kkakim D. Ikramov, and Andrey B. Kucherov. On the growth factor in Gaussian elimination for generalized Higham matrices. *Numer. Linear Algebra Appl.*, 9:107–114, 2002. (Cited on p. 74.)

[153] Alan George and Joseph W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981. (Cited on pp. 156, 157, 165, 616.)

[154] Alan George and Joseph W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989. (Cited on p. 156.)

[155] J. A. George and M. T. Heath. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra Appl.*, 34:69–83, 1980. (Cited on pp. 274, 275.)

[156] J. A. George and J. W.-H. Liu. Householder reflections versus Givens rotations in sparse orthogonal decomposition. *Linear Algebra Appl.*, 88/89:223–238, 1987. (Cited on p. 276.)

[157] J. A. George and E. G. Ng. On row and column orderings for sparse least squares problems. *SIAM J. Numer. Anal.*, 20:326–344, 1983. (Cited on p. 274.)

[158] S. A. Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Akademia Nauk SSSR, Mathematics and Natural Sciences*, 6:749–754, 1931. (Cited on p. 335.)

[159] J. R. Gilbert, E. G. Ng, and B. W. Peyton. Separators and structure prediction in sparse orthogonal factorization. *Linear Algebra Appl.*, 262:83–97, 1997. (Cited on p. 275.)

[160] John R. Gilbert, Cleve Moler, and Robert Schreiber. Sparse matrices in Matlab: Design and implementation. *SIAM J. Matrix Anal. Appl.*, 13(1):333–356, 1992. (Cited on p. 151.)

[161] John R. Gilbert and Tim Peierls. Sparse partial pivoting in time proportional to arithmetic operations. *SIAM J. Sc. Statist. Comput.*, 9(5):862–874, 1988. (Cited on p. 152.)

[162] Philip E. Gill, Gene H. Golub, Walter Murray, and Michael Saunders. Methods for modifying matrix factorizations. *Math. Comp*, 28:505–535, 1974. (Cited on p. 590.)

[163] Philip E. Gill and Walter Murray. Algorithms for the solution of the nonlinear least squares problem. *SIAM J. Numer. Anal.*, 15:977–992, 1978. (Cited on pp. 571, 572.)

[164] PHilip E. Gill, Walter Murray, D. B. Ponceleón, and Michael A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix. Anal. Appl.*, 13:292–311, 1992. (Cited on p. 528.)

[165] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, London, UK, 1981. (Cited on pp. 566, 609.)

[166] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Numerical Linear Algebra and Optimization. Volume 1*. Addison-Wesley, London, UK, 1991. (Cited on pp. 595, 601, 602.)

[167] Wallace G. Givens. Computation of plane unitary rotations transforming a general matrix to triangular form. *SIAM J. Appl. Math.*, 6(1):26–50, 1958. (Cited on pp. 186, 313.)

[168] G. H. Golub and R. J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra Appl.*, 34:3–28, 1980. (Cited on p. 199.)

[169] Gene H. Golub. Numerical methods for solving least squares problems. *Numer. Math.*, 7:206–216, 1965. (Cited on p. 313.)

[170] Gene H. Golub. Least squares, singular values and matrix approximations. *Aplikace Matematiky*, 13:44–51, 1968. (Cited on p. 451.)

[171] Gene H. Golub. Matrix computations and the theory of moments. In S. D. Chatterji, editor, *Proceedings of the International Congress of Mathematicians, Zürich 1994*, volume 2, pages 1440–1448, Basel, Switzerland, 1995. Birkhäuser-Verlag. (Cited on p. 500.)

[172] Gene H. Golub and Carl D. Meyer Jr. Using the QR factorization and group inversion to compute, differentiate, and estimate the sensitivity of stationary probabilties for Markov chains. *SIAM J. Alg. Disc. Meth.*, 7(2):273–281, 1975. (Cited on pp. 421, 423.)

[173] Gene H. Golub and W. Kahan. Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal. Ser. B*, 2:205–224, 1965. (Cited on pp. 314, 438, 451.)

[174] Gene H. Golub, Virginia Klema, and George W. Stewart. Rank degeneracy and least squares problems. Tech. Report STAN-CS-76-559, August 1976, Computer Science Department, Stanford University, CA, 1976. (Cited on p. 250.)

[175] Gene H. Golub and Randy J. LeVeque. Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems. In *Proceedings of the 1979 Army Numerical Analysis and Computers Conf.*, pages 1–12. White Sands Missile Range, White Sands, NM, ARO Report 79-3, 1979. (Cited on p. 574.)

[176] Gene H. Golub and Gérard Meurant. Matrices, moments and quadrature. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1993: Proceedings of the 13th Dundee Biennial Conference*, Pitman Research Notes in mathematics, pages 105–156. Longman Scientific and Technical, Harlow, Essex, UK, 1994. (Cited on p. 500.)

[177] Gene H. Golub, Stephen Nash, and Charles F. Van Loan. A Hessenberg–Schur method for the the matrix problem $AX + XB = C$. *IEEE Trans. Auto. Cont.*, AC-24:909–913, 1972. (Cited on p. 329.)

[178] Gene H. Golub and Dianne P. O'Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Review*, 31:50–102, 1989. (Cited on pp. 541, 542.)

[179] Gene H. Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.*, 10(2):413–432, 1973. (Cited on p. 573.)

[180] Gene H. Golub and Victor Pereyra. Separable nonlinear least squares: the variable projection method and its application. *Inverse Problems*, 19:R1–R26, 2003. (Cited on pp. 576, 608.)

[181] Gene H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numer. Math.*, 14:403–420, 1970. (Cited on pp. 19, 383, 451.)

[182] Gene H. Golub and Henk A. van der Vorst. Closer to the solution: iterative linear solvers. In I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 63–92. Clarendon Press, Oxford, 1997. (Cited on p. 522.)

[183] Gene H. Golub and Charles F. Van Loan. An analysis of the total least squares problem. *SIAM J. Numer. Anal.*, 17(6):883–893, 1980. (Cited on pp. 300, 314.)

[184] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996. (Cited on pp. 30, 164, 189, 250, 312, 340, 358, 405, 451, 615.)

[185] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, PA, 1997. (Cited on pp. 541, 542.)

[186] Andreas Griewank, editor. *Evaluating Derivatives; Principles and Techniques of Algorithmic Differentiation*. Frontier in Applied Mathematics, Volume 19. SIAM, Philadelphia, PA, 2000. (Cited on p. 608.)

[187] Ming Gu. Backward perturbation bounds for linear least squares problems. *SIAM J. Matrix. Anal. Appl.*, 20(2):363–372, 1998. (Cited on p. 313.)

[188] Ming Gu, James W. Demmel, and Inderjit Dhillon. Efficient computation of the singular value decomposition with applications to least squares problems. Tech. Report TR/PA/02/33, Department of Mathematics and Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720, 1994. (Cited on p. 389.)

[189] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal svd. *SIAM J. Matrix. Anal. Appl.*, 16(1):79–92, 1995. (Cited on p. 392.)

[190] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix. Anal. Appl.*, 16(1):172–191, 1995. (Cited on pp. 386, 389.)

[191] M. E. Gulliksson, Inge Söderkvist, and Per-Åke Wedin. Algorithms for constrained and weighted nonlinear least squares. *SIAM J. Optim.*, 7:208–224, 1997. (Cited on p. 576.)

[192] M. E. Gulliksson and Per-Åke Wedin. Perturbation theory for generalized and constrained linear least squares. *Numer. Linear Algebra Appl.*, 7:181–196, 2000. (Cited on p. 312.)

[193] Fred G. Gustavson. Recursion leads to automatic variable blocking for dense linear-algebra algorithms. *IBM J. Res. Develop.*, 41(6):737–754, 1997. (Cited on pp. 105, 109.)

[194] William W. Hager. Condition estimates. *SIAM J. Sci. Stat. Comput.*, 5(2):311–316, 1984. (Cited on pp. 124, 125.)

[195] William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989. (Cited on p. 13.)

[196] Nicholas Hale, Nicholas J. Higham, and Lloyd N. Trefethen. Computing A^α, $\log(A)$ and related matrix functions by contour integrals. *SIAM J. Matrix Anal. Appl.*, to appear, 2008. (Cited on p. 452.)

[197] Sven Hammarling. Numerical solution of the stable non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982. (Cited on p. 329.)

[198] Eldon Hansen. Interval arithmetic in matrix computations. ii. *SIAM J. Numer. Anal.*, 4(1):1–9, 1965. (Cited on p. 141.)

[199] Eldon Hansen. *Topics in Interval Analysis.* Oxford University Press, Oxford, 1969. (Cited on p. 140.)

[200] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion.* SIAM, Philadelphia, 1998. (Cited on pp. 245, 312, 313.)

[201] Richard J. Hanson and Michael J. Norris. Analysis of measurements based on the singular value decomposition. *SIAM J. Sci. Stat. Comput.*, 2(3):363–373, 1981. (Cited on p. 297.)

[202] G. I. Hargreaves. Interval analysis in MATLAB. Numer. anal. report 418, Department of Mathematics, University of Manchester, 2002. (Cited on p. 140.)

[203] M. D. Hebden. An algorithm for minimization using exact second derivatives. Tech. Report T. P. 515, Atomic Energy Research Establishment, Harwell, England, 1973. (Cited on p. 291.)

[204] H. V. Henderson and S. R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23(1):53–60, 1981. (Cited on p. 12.)

[205] Karl Hessenberg. Behandlung linearer Eigenvertaufgaben mit Hilfe der Hamilton–Cayleyschen Gleichung. Technical Report IPM-1, Institute of Practical Mathematics, Technische Hochschule, Darmstadt, 1940. (Cited on p. 6.)

[206] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear system. *J. Res. Nat. Bur. Standards, Sect. B*, 49:409–436, 1952. (Cited on pp. 492, 542.)

[207] K. L. Hiebert. An evaluation of mathematical software that solves systems of nonlinear equations. *ACM Trans. Math. Software*, 8:5–20, 1982. (Cited on p. 608.)

[208] Nicholas J. Higham. Computing the polar decomposition—with applications. *SIAM J. Sci. Stat. Comput.*, 7(4):1160–1174, 1986. (Cited on pp. 183, 420, 427.)

[209] Nicholas J. Higham. FORTRAN codes for estimating the one-norm of a real or complex matrix, with application to condition estimation. *ACM Trans. Math. Software*, 14(4):381–396, 1988. (Cited on p. 124.)

[210] Nicholas J. Higham. Evaluating Padé approximants of the matrix logarithm. *SIAM J. Matrix Anal. Appl.*, 22:4:1126–1135, 2001. (Cited on p. 415.)

[211] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, second edition, 2002. (Cited on pp. 18, 33, 35, 42, 66, 78, 81, 82, 105, 125, 126, 164, 164, 164, 222, 235, 616.)

[212] Nicholas J. Higham. J-Orthogonal matrices: Properties and generation. *SIAM Review*, 45(3):504–519, 2003. (Cited on pp. 284, 314.)

[213] Nicholas J. Higham. The scaling and squaring method for the matrix exponential function. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193, 2005. (Cited on p. 414.)

[214] Y. T. Hong and C. T. Pan. Rank-revealing QR decompositions and the singular value decomposition. *Math. Comp.*, 58:213–232, 1992. (Cited on p. 314.)

[215] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985. ISBN 0-521-30586-1. (Cited on pp. 26, 164, 615.)

[216] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1991. ISBN 0-521-30587-X. (Cited on pp. 164, 615.)

[217] Harold Hotelling. Some new methods in matrix calculus. *Ann. Math. Statist.*, 14:1–34, 1943. (Cited on p. 164.)

[218] A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.*, 5:339–342, 1958. (Cited on pp. 184, 313.)

[219] A. S. Householder and F. L. Bauer. On certain iterative methods for solving linear systems. *Numer. Math.*, 2:55–59, 1960. (Cited on p. 505.)

[220] Alston S. Householder. *The Theory of Matrices in Numerical Analysis*. Dover, New York, 1975. Corrected republication of work first published in 1964 by Blaisdell Publ. Co., New York. (Cited on pp. 164, 615.)

[221] Bruno Iannazzo. A note on computing the matrix square root. *Calcolo*, 40:273–283, 2003. (Cited on p. 417.)

[222] Bruno Iannazzo. A the Newton method for the matrix pth root. *SIAM J. Matrix Anal. Appl.*, 28(2):503–523, 2006. (Cited on p. 417.)

[223] Yasuhiko Ikebe. On inverses of Hessenberg matrices. *Linear Algebra Appl.*, 24:93–97, 1979. (Cited on p. 94.)

[224] Ilse Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 39:254–291, 1997. (Cited on p. 451.)

[225] C. G. J. Jacobi. Über ein leichtes Verfahren die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen. *J. reine angew. Math..*, 30:51–94, 1846. (Cited on p. 395.)

[226] E. R. Jessup and D. C. Sorenson. A parallel algorithm for computing the singular value decomposition of a matrix. *SIAM J. Matrix. Anal. Appl.*, 15(2):530–548, 1994. (Cited on p. 389.)

[227] L. W. Johnson and D. R. Scholz. On Steffensen's method. *SIAM J. Numer. Anal.*, 5(2):296–302, 1968. (Cited on p. 554.)

[228] S. Kaczmarz. Angenäherte auflösung von systemen linearen gleichungen. *Acad. Polon. Sciences et Lettres*, pages 355–357, 1937. (Cited on p. 507.)

[229] W. M. Kahan. Accurate eigenvalues of a symmetric tri-diagonal matrix. Tech. Report No. CS-41, Revised June 1968, Computer Science Department, Stanford University, CA, 1966. (Cited on p. 393.)

[230] W. M. Kahan. Numerical linear algebra. *Canad. Math. Bull.*, 9:757–801, 1966. (Cited on pp. 56, 117.)

[231] Rune Karlsson and Bertil Waldén. Estimation of optimal backward perturbation bounds for the linear least squares problem. *BIT*, 37:4:862–869, 1997. (Cited on p. 313.)

[232] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984. (Cited on p. 606.)

[233] T. Kato. *Perturbation Thepry for Linear Operators*. Springer-Verlag, New York, 1976. (Cited on p. 409.)

[234] L. Kaufman. Variable projection methods for solving separable nonlinear least squares problems. *BIT*, 15:49–57, 1975. (Cited on pp. 574, 575.)

[235] L. Kaufman and V. Pereyra. A method for separable nonlinear least squares problems with separable nonlinear equality constraints. *SIAM J. Numer. Anal.*, 15:12–20, 1978. (Cited on p. 574.)

[236] L. Kaufman and G. Sylvester. Separable nonlinear least squares problems with multiple right hand sides. *SIAM J. Matrix Anal. Appl.*, 13:68–89, 1992. (Cited on p. 574.)

[237] Charles S. Kenney and Alan J. Laub. Padé error estimates for the logarithm of a matrix. *Internat J. Control.*, 10:707–730, 1989. (Cited on p. 414.)

[238] Charles S. Kenney and Alan J. Laub. Rational iterative methods for the matrix sign function. *SIAM J. Matrix. Anal. Approx.*, 12(2):273–291, 1991. (Cited on p. 428.)

[239] Charles S. Kenney and Alan J. Laub. On scaling Newton's method for polar decomposition and the matrix sign function. *SIAM J. Matrix. Anal. Approx.*, 13(3):688–706, 1992. (Cited on pp. 420, 421.)

[240] Philip A. Knight and Daniel Ruiz. A fast method for matrix balancing. In Andreas Frommer, Michael W. Mahoney, and Daniel B. Szyld, editors, *Web Information Retrieval and Linear Algebra Algorithms*, Schloss Dagstuhl, Germany, 2007. Internationales Begegnungs- aunf Forschungszentrum für Informatik. (Cited on p. 372.)

[241] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Reading, MA, third edition, 1998. (Cited on p. 106.)

[242] E. G. Kogbetliantz. Solution of linear equations by diagonalization of coefficients matrix. *Quart. Appl. Math.*, 13:123–132, 1955. (Cited on p. 399.)

[243] Tamara G. Kolda. Multilinear operators for higher-order decompositions. Technical Report SAND 2006–2081, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, April 2006. (Cited on p. 276.)

[244] Tamara G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482, 2003. (Cited on p. 609.)

[245] F. T. Krogh. Efficient implementation of a variable projection algorithm for nonlinear least squares. *Comm. ACM*, 17:167–169, 1974. (Cited on p. 575.)

[246] A. N. Krylov. On the numerical solution of the equation by which, in technical matters, frequencies of small oscillations of material systems are determined. *Izv. Akad. Nauk. S.S.S.R. Otdel.Mat. Estest. Nauk*, VII:4:491–539, 1931. in Russian. (Cited on p. 440.)

[247] Vera N. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *USSR Comput. Math. Phys.*, 3:637–657, 1961. (Cited on p. 361.)

[248] H. P. Künzi, H. G. Tzschach, and C. A. Zehnder. *Numerical Methods of Mathematical Optimization*. Academic Press, New York, 1971. (Cited on p. 609.)

[249] Peter Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, New York, 1985. (Cited on pp. 452, 615.)

[250] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards, Sect. B*, 45:255–281, 1950. (Cited on pp. 436, 518.)

[251] Cornelius Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards, Sect. B*, 49:33–53, 1952. (Cited on p. 520.)

[252] L. Landweber. An iterative formula for Fredholm integral equations of the first kind. *Amer. J. Math.*, 73:615–624, 1951. (Cited on pp. 504, 511, 542.)

[253] P. S. Laplace. *Théorie analytique des probabilités. Premier supplément*. Courcier, Paris, 3rd edition, 1820. with an introduction and three supplements. (Cited on p. 199.)

[254] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000. (Cited on p. 276.)

[255] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank $(R_1, R_2, \ldots, R_N)$ approximation of hogher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000. (Cited on p. 276.)

[256] P. Läuchli. Jordan-Elimination und Ausgleichung nach kleinsten Quadraten. *Numer. Math.*, 3:226–240, 1961. (Cited on p. 205.)

[257] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1974. Reprinted by SIAM, Philadelphia, PA, 1995. (Cited on pp. 219, 253, 267, 267, 312.)

[258] Charles L. Lawson, Richard J. Hanson, D. R. Kincaid, and Fred T. Krogh. Basic Linear Algebra Subprograms for Fortran usage. *ACM Trans. Math. Software*, 5:308–323, 1979. (Cited on p. 97.)

[259] Peter D. Lax. *Linear Algebra*. John Wiley, New York, 1997. (Cited on p. 615.)

[260] Adrien-Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. Courcier, Paris, 1805. (Cited on p. 312.)

[261] Richard B. Lehoucq. The computations of elementary unitary matrices. *ACM Trans. Math. Software*, 22:393–400, 1996. (Cited on p. 312.)

[262] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944. (Cited on p. 569.)

[263] N. Levinson. The Wiener rms root-mean square error criterion in filter design and prediction. *J. Math. Phys.*, 25:261–278, 1947. (Cited on p. 159.)

[264] Ren-Cang Li. Solving secular equations stably and efficiently. Technical Report UCB/CSD-94-851, Computer Science Department, University of California, Berkeley CA 94720, 1994. (Cited on p. 389.)

[265] Ren-Cang Li. Relative perturbation theory: I. Eigenvalue and singular value variations. *SIAM J. Matrix Anal. Appl.*, 19(4):956–982, 1998. (Cited on p. 451.)

[266] Ren-Cang Li. Relative perturbation theory: II. Eigenspace and singular subspace variations. *SIAM J. Matrix Anal. Appl.*, 20(2):471–492, 1998. (Cited on p. 451.)

[267] X. S. Li, James W. Demmel, David H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, A. Kapur, M. C. Martin, T. Tung, and D. J. Yoo. Design, implementation and testing of extended and mixed precision BLAS. LAPACK working note 149 Tech. Report CS-00-451, Department of Computer Science, University of Tennessee, Knoxville, TN, 2000. (Cited on p. 136.)

[268] J. W.-H. Liu. On general row merging schemes for sparse Givens transformations. *SIAM J. Sci. Stat. Comput.*, 7:1190–1211, 1986. (Cited on p. 276.)

[269] D. G. Luenberger. *Introduction to Dynamic Systems. Theory, Models and Applications*. John Wiley, New York, 1979. (Cited on p. 609.)

[270] Marvin Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities*. Allyn and Bacon, Boston, MA, 1964. Reprinted by Dover, Mineola, NY, 1992. (Cited on pp. 16, 615.)

[271] A. A. Markov. *Wahrscheinlichheitsrechnung*. Liebmann, Leipzig, second edition, 1912. (Cited on p. 173.)

[272] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11:431–441, 1963. (Cited on p. 569.)

[273] R. S. Martin and James H. Wilkinson. Reduction of the symmetric eigenproblem $Ax = \lambda Bx$ and related problems to standard form. *Numer. Math.*, 11:99–110, 1968. Also in [390, pp. 303–314]. (Cited on p. 445.)

[274] Roy Mathias and G. W. Stewart. A block QR algorithm and the singular value decompositions. *Linear Algebra Appl.*, 182:91–100, 1993. (Cited on p. 249.)

[275] Pontus Matstoms. *Sparse QR Factorization with Applications to Linear Least Squares Problems*. Ph.D. thesis, Department of Mathematics, Linköping University, Sweden, 1994. (Cited on p. 276.)

[276] J. A. Meijerink and Henk A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31:148–162, 1977. (Cited on p. 534.)

[277] Jean Meinguet. Refined error analysis of Cholesky factorization. *SIAM J. Numer. Anal.*, 20(6):1243–1250, 1983. (Cited on p. 132.)

[278] Carl D. Meyer. The role of the group generalized inverse in the theory of finite Markov chains. *SIAM Review*, 17:443–464, 1975. (Cited on p. 423.)

[279] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, 2000. (Cited on p. 421.)

[280] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford*, 11:50–59, 1960. (Cited on p. 182.)

[281] Cleve Moler and Charles F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45:3–49, 2003. (Cited on pp. 412, 414.)

[282] Cleve Moler and G. W. Stewart. An algorithm for generalized eigenvalue problems. *SIAM J. Numer. Anal.*, 10:241–256, 1973. (Cited on p. 446.)

[283] E. H. Moore. On the reciprocal of the general algebraic matrix. *Bull. Amer. Math. Soc.*, 26:394–395, 1920. (Cited on p. 312.)

[284] J. J. Moré and M. Y. Cosnard. Numerical solution of nonlinear equations. *ACM. Trans. Math. Software*, 5:64–85, 1979. (Cited on pp. 554, 608.)

[285] J. J. Moré and Stephen J. Wright. *Optimization Software Guide*. SIAM, Philadelphia, PA, 1993. (Cited on p. 609.)

[286] N. M. Nachtigal, S. C. Reddy, and Lloyd N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM J. Matrix. Anal. Appl.*, 13:778–795, 1992. (Cited on p. 522.)

[287] Larry Neal and George Poole. A geometric analysis of Gaussian elimination: Part II. *Linear Algebra Appl.*, 173:239–264, 1992. (Cited on p. 165.)

[288] Larry Neal and George Poole. The rook's pivoting strategy. *J. Comput. Appl. Math.*, 123:353–369, 2000. (Cited on p. 165.)

[289] John von Neumann. Some matrix inequalities in the space of matrices. *Tomsk Univ. Rev.*, 1:286–299, 1937. in Russian. (Cited on p. 29.)

[290] John von Neumann and Herman H. Goldstein. Numerical inverting of matrices of high order. *Bull. Amer. Math. Soc.*, 53:1021–1099, 1947. (Cited on p. 164.)

[291] Ben Noble. Methods for computing the moore–penrose generalized inverse and related matters. In M. Z. Nashed, editor, *Generalized Inverses and Applications*, pages 245–302. Academic Press, Inc., New York, 1976. (Cited on pp. 209, 313.)

[292] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, New York, second edition, 2006. (Cited on p. 609.)

[293] James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970. (Cited on pp. 549, 550.)

[294] E. E. Osborne. On pre-conditioning of matrices. *J. Assoc. Comput. Mach.*, 7:338–345, 1960. (Cited on p. 372.)

[295] M. R. Osborne. *Finite Algorithms in Optimization and Data Analysis*. John Wiley, New York, 1985. (Cited on p. 309.)

[296] A. M. Ostrowski. *Solution of Equations in Euclidian and Banach Spaces*. Academic Press, New York, third edition, 1973. (Cited on p. 609.)

[297] C. C. Paige. Computer solution and perturbation analysis of generalized linear least squares problems. *Math. Comp.*, 33:171–184, 1979. (Cited on p. 281.)

[298] Christopher C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*. PhD thesis, University of London, 1971. (Cited on p. 438.)

[299] Christopher C. Paige. Some aspects of generalized QR factorizations. In M. G. Cox and S. J. Hammarling, editors, *Reliable Numerical Computation*, pages 71–91. Clarendon Press, Oxford, UK, 1990. (Cited on p. 314.)

[300] Christopher C. Paige and Michael Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975. (Cited on p. 499.)

[301] Christopher C. Paige and Michael A. Saunders. Toward a generalized singular value decomposition. *SIAM J. Numer. Anal.*, 18:398–405, 1981. (Cited on pp. 193, 448, 451.)

[302] Christopher C. Paige and Z. Strakoš. Unifying least squares, total least squares and data least squares. In S. Van Huffel and P. Lemmerling, editors, *Total Least Squares and Errors-in-Variables Modeling*, pages 25–34. Kluwer Academic Publishers, Dordrecht, 2002. (Cited on p. 257.)

[303] Christopher C. Paige and Zdeněk Strakoš. Scaled total least squares fundamentals. *Numer. Math.*, 91:1:117–146, 2002a. (Cited on p. 314.)

[304] Beresford N. Parlett. The new qd algorithm. *Acta Numerica*, 4:459–491, 1995. (Cited on p. 401.)

[305] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics 20. SIAM, Philadelphia, PA, 1998. Republication of work published in 1980 by Prentice-Hall, Englewood Cliffs. (Cited on pp. 185, 229, 341, 346, 356, 376, 378, 379, 429, 443, 451, 616.)

[306] Beresford N. Parlett and Christian Reinsch. Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numer. Math.*, 13:293–304, 1969. (Cited on p. 372.)

[307] Roger Penrose. A generalized inverse for matrices. *Proc. Cambridge Philos. Soc.*, 51:406–413, 1955. (Cited on p. 312.)

[308] G. Peters and J. H. Wilkinson. Inverse iteration, ill-conditioned equations and Newton's method. *SIAM Review*, 3:339–360, 1979. (Cited on p. 354.)

[309] G. Peters and James H. Wilkinson. The least squares problem and pseudo-inverses. *Comput. J.*, 13:309–316, 1970. (Cited on pp. 209, 313.)

[310] G. Peters and James H. Wilkinson. On the stability of Gauss–Jordan elimination with pivoting. *Comm. Assos. Comput. Mach.*, 18:20–24, 1975. (Cited on p. 67.)

[311] R. L. Placket. A historical note on the method of least squares. *Biometrika*, 36:456–460, 1949. (Cited on p. 173.)

[312] R. L. Placket. The discovery of the method of least squares. *Biometrika*, 59:239–251, 1972. (Cited on p. 173.)

[313] Robert J. Plemmons. Linear least squares by elimination and MGS. *J. Assoc. Comput. Mach.*, 21:581–585, 1974. (Cited on p. 223.)

[314] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations*. Gordon and Breach, London, UK, 1970. (Cited on p. 608.)

[315] M. J. D. Powell and J. K. Reid. On applying Householder's method to linear least squares problems. In A. J. H. Morell, editor, *Information Processing 68. Proceedings of the IFIP Congress 68*, pages 122–126. North-Holland, Amsterdam, 1969. (Cited on p. 281.)

[316] Philip Rabinowitz. *Numerical Methods for Non-Linear Algebraic Equations*. Gordon and Breach, London, UK, 1970. (Cited on p. 609.)

[317] Louis B. Rall. *Automatic Differentiation. Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1981. (Cited on p. 608.)

[318] J. K. Reid. A note on the least squares solution of a band system of linear equations by Householder reductions. *Comput J.*, 10:188–189, 1967. (Cited on p. 313.)

[319] John K. Reid. A note on the stability of Gaussian elimination. *J. Inst. Maths. Applics.*, 8:374–375, 1971. (Cited on p. 542.)

[320] Christian H. Reinsch. Smoothing by spline functions. *Numer. Math.*, 16:451–454, 1971. (Cited on pp. 291, 292.)

[321] Sara Robinson. Toward an optimal algorithm for matrix multiplication. *SIAM News*, 38(4):2–3, 2005. (Cited on p. 164.)

[322] Axel Ruhe and Per-Åke Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Rev.*, 22:3:318–337, 1980. (Cited on p. 575.)

[323] Sigfried M. Rump. Fast and parallel interval arithmetic. *BIT*, 39(3):534–554, 1999. (Cited on p. 141.)

[324] Sigfried M. Rump. INTLAB—INTerval LABoratory. In T. Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. (Cited on pp. 140, 141, 141.)

[325] Heinz Rutishauser. Solution of eigenvalue problems with the LR-transformation. *Nat. Bur. Standards Appl. Math. Ser.*, 49:47–81, 1958. (Cited on p. 360.)

[326] Heinz Rutishauser. The Jacobi method for real symmetric matrices. *Numer. Math.*, 9:1–10, 1966. (Cited on pp. 397, 432.)

[327] Yosef Saad. *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York, 1992. (Cited on p. 616.)

[328] Yosef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM Publications., Philadelphia, PA, second edition, 2003. (Cited on pp. 541, 616.)

[329] Yosef Saad and Henk A. van der Vorst. Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.*, 123:1–33, 2000. (Cited on p. 541.)

[330] Werner Sautter. Fehleranalyse für die Gauss-Elimination zur Berechnung der Lösung minimaler Länge. *Numer. Math.*, 30:165–184, 1978. (Cited on p. 313.)

[331] W. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrica.*, 31:1–10, 1966. (Cited on p. 297.)

[332] Issai Schur. Über die characteristischen Würzeln einer linearen Substitution mit einer Anwendung auf die Theorie der Integral Gleichungen. *Mathematische Annalen.*, 66:448–510, 1909. (Cited on p. 324.)

[333] Hubert Schwetlick and V. Tiller. Numerical methods for estimating parameters in nonlinear models with errors in the variables. *Technometrics*, 27:17–24, 1985. (Cited on p. 578.)

[334] Robert D. Skeel. Scaling for stability in Gaussian elimination. *J. Assoc. Comput. Mach.*, 26:494–526, 1979. (Cited on pp. 134, 165.)

[335] Robert D. Skeel. Iterative refinement implies numerical stability for Gaussian elimination. *Math. Comput.*, 35:817–832, 1980. (Cited on pp. 138, 165.)

[336] B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystems Routines—EISPACK Guide*. Springer-Verlag, New York, second edition, 1976. (Cited on pp. 98, 451, 616, 616.)

[337] Inge Söderkvist. Perturbation analysis of the orthogonal Procrustes problem. *BIT*, 33(4):687–694, 1993. (Cited on p. 298.)

[338] Torsten Söderström and G. W. Stewart. On the numerical properties of an iterative method for computing the Moore–Penrose generalized inverse. *SIAM J. Numer. Anal.*, 11(1):61–74, 1974. (Cited on p. 67.)

[339] Peter Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989. (Cited on p. 523.)

[340] G. W. Stewart. *Introduction to Matrix Computations*. Academic Press, New York, 1973. (Cited on pp. 181, 341, 356, 443, 448, 615.)

[341] G. W. Stewart. The economical storage of plane rotations. *Numer. Math.*, 25:137–138., 1976. (Cited on p. 189.)

[342] G. W. Stewart. On the perturbation of pseudo-inverses, projections, and linear least squares problems. *SIAM Review*, 19(4):634–662, 1977. (Cited on p. 179.)

[343] G. W. Stewart. An updating algorithm for subspace tracking. *IEEE Trans. Signal Process.*, 40:1535–1541, 1992. (Cited on p. 313.)

[344] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35(4):551–556, 1993. (Cited on p. 312.)

[345] G. W. Stewart. Updating a rank-revealing ULV decomposition. *SIAM J. Matrix Anal. Appl.*, 14:494–499, 1993. (Cited on p. 313.)

[346] George W. Stewart. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, Philadelphia, PA, 1998. (Cited on pp. 30, 129, 164, 164, 615.)

[347] George W. Stewart. *Matrix Algorithms Volume II: Eigensystems*. SIAM, Philadelphia, PA, 2001. (Cited on p. 452.)

[348] George W. Stewart and Ji guang. Sun. *Matrix Perturbation Theory*. Academic Press, Boston, MA, 1990. (Cited on pp. 25, 26, 28, 164, 164, 451, 452, 615.)

[349] M. Stewart and G. W. Stewart. On hyperbolic triangularization: Stability and pivoting. *SIAM J. Matrix Anal. Appl.*, 19:4:8471–860, 1998. (Cited on p. 240.)

[350] S. M. Stiegler. Gauss and the invention of least squares. *Ann. Statist.*, 9:465–474, 1981. (Cited on pp. 212, 312.)

[351] Volker Strassen. Gaussian elimination is not optimal. *Numer. Math..*, 13:354–356, 1969. (Cited on pp. 31, 105.)

[352] Ji-guang Sun and Zheng Sun. Optimal backward perturbation bounds for underdetermined systems. *SIAM J. Matrix Anal. Appl.*, 18(2):393–402, 1997. (Cited on p. 313.)

[353] A. H. Taub. *John von Neumann Collected Works*, volume V, Design of Computers, Theory of Automata and Numerical Analysis. Pergamon Press, Oxford, 1963. (Cited on p. 164.)

[354] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963. (Cited on p. 288.)

[355] Françoise Tisseur. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 22(4):1038–1057, 2001. (Cited on p. 445.)

[356] Sivan Toledo. Locality of reference in LU decomposition with partial pivoting. *SIAM J. Matrix Anal. Appl.*, 18(4):1065–1081, 1997. (Cited on p. 109.)

[357] Andrea Toselli and Olof Widlund. *Domain Decomposition Methods—Algorithms and Theory.* Number 34 in Computational Mathematics. Springer, New York, 2005. xv+450 pages. ISBN 978-3-540-20696-5. (Cited on p. 541.)

[358] Lloyd N. Trefethen. Three mysteries of Gaussian elimination. *SIGNUM Newsletter*, 20(4):2–5, 1985. (Cited on p. 40.)

[359] Lloyd N. Trefethen. Pseudospectra of linear operators. *SIAM Review*, 39:383–406, 1997. (Cited on p. 452.)

[360] Lloyd N. Trefethen and III David Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997. (Cited on p. 254.)

[361] Lloyd N. Trefethen and Robert S. Schreiber. Average-case stability of Gaussian elimination. *SIAM J. Matrix Anal. Appl.*, 11(3):335–360, 1990. (Cited on pp. 56, 128.)

[362] W. F. Trench. An algorithm for the inversion of finite Toeplitz matrices. *J. SIAM*, 12:515–522, 1964. (Cited on p. 165.)

[363] M. J. Tsatsomeros. Principal pivot transforms. *Linear Algebra Appl.*, 307:151–165, 2000. (Cited on p. 313.)

[364] A. M. Turing. Rounding-off errors in matrix processes. *Quart. J. Mech. Appl. Math.*, 1:287–308, 1948. (Cited on p. 164.)

[365] A. van der Sluis. Condition numbers and equilibration of matrices. *Numer. Math.*, 14:14–23, 1969. (Cited on p. 532.)

[366] A. van der Sluis and G. Veltkamp. Restoring rank and consistency by orthogonal projection. *Linear Algebra Appl.*, 28:257–278, 1979. (Cited on p. 180.)

[367] Henk A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput..*, 12:631–644, 1992. (Cited on p. 524.)

[368] Henk A. van der Vorst. Computational methods for large eigenvalue problems. In Philippe G. Ciarlet and F. Cucker, editors, *Handbook of Numerical Analysis*, volume VIII, pages 3–179. North Holland Elsevier, Amsterdam, 2002. (Cited on p. 451.)

[369] Henk A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press, Cambridge, UK, 2003. (Cited on p. 541.)

[370] Sabine Van Huffel and Joos Vandewalle. *The Total Least Squares Problem; Computational Aspects and Analysis.* SIAM, Philadelphia, PA, 1991. (Cited on pp. 301, 314, 386.)

[371] Charles F. Van Loan. Generalizing the singular value decomposition. *SIAM J. Numer. Anal.*, 13:76–83, 1976. (Cited on pp. 448, 452.)

[372] Raf Vandebril, Marc Van Barel, and Nicola Mastronardi. *Matrix Computations and Semiseparable Matrices*, volume 1; Linear Systems. The Johns Hopkins University Press, Baltimore, MD, 2007. (Cited on p. 159.)

[373] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer's International Series. Springer, New York, third edition, 2008. (Cited on p. 609.)

[374] James M. Varah. On the solution of block-tridiagonal systems arising from certain finite-difference equations. *Math. Comp.*, 26(120):859–869, 1972. (Cited on p. 104.)

[375] Richard S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, 1962. (Cited on pp. 332, 460, 541, 616.)

[376] Richard S. Varga. *Gerschgorin and his Circles*. Springer-Verlag, Berlin, Heidelberg, New York, 2004. (Cited on p. 335.)

[377] Urs von Matt. The orthogonal qd-algorithms. *SIAM J. Sci. Comput.*, 18(4):1163–1186, 1997. (Cited on p. 401.)

[378] Bertil Waldén, Rune Karlsson, and Ji guang. Sun. Optimal backward perturbation bounds for the linear least squares problem. *Numer. Linear Algebra Appl.*, 2:271–286, 1995. (Cited on p. 313.)

[379] David S. Watkins. Understanding the QR algorithm. *SIAM Review*, 24:427–440, 1982. (Cited on p. 362.)

[380] David S. Watkins. *Fundamentals of Matrix Computation*. Wiley-InterScience, New York, second edition, 2002. (Cited on pp. 164, 452.)

[381] David S. Watkins. *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*. SIAM, Philadelphia, PA, 2007. (Cited on p. 452.)

[382] David S. Watkins. The QR algorithm revisited. *SIAM Review*, 50(1):133–145, 2008. (Cited on p. 451.)

[383] Per-Åke Wedin. On pseudoinverses of perturbed matrices. Tech. Report, Department of Computer Science, Lund University, Sweden, 1969. (Cited on p. 180.)

[384] Per-Åke Wedin. Perturbation theory for pseudo-inverses. *BIT*, 9:217–232, 1973. (Cited on pp. 180, 203.)

[385] H. Wielandt. Das Iterationsverfahren be nicht selbstadjungierten linearen Eigenwertaufgaben. *Math. Z.*, 50:93–143, 1944. (Cited on p. 353.)

[386] James H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. Assos. Comput. Mach.*, 8:281–330, 1961. (Cited on pp. 55, 58, 127, 164.)

[387] James H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965. (Cited on pp. 32, 128, 186, 312, 342, 387, 451, 616.)

[388] James H. Wilkinson. Global convergence of tridiagonal QR algorithm with origin shifts. *Linear Algebra Appl.*, 1:409–420, 1968. (Cited on p. 378.)

[389] James H. Wilkinson. A priori error analysis of algebraic processes. In *Proceedings International Congress Math.*, pages 629–639. Izdat. Mir, Moscow, 1968. (Cited on p. 132.)

[390] James H. Wilkinson and C. Reinsch, editors. *Handbook for Automatic Computation. Vol. II, Linear Algebra*. Springer-Verlag, New York, 1971. (Cited on pp. 97, 372, 376, 376, 444, 445, 451, 616, 634.)

[391] Svante Wold, Axel Ruhe, Herman Wold, and W. J. Dunn. The collinearity problem in linear regression, the partial least squares (pls) approach to generalized inverses. *SIAM J. Sci. Stat. Comput.*, 5:735–743, 1984. (Cited on p. 314.)

[392] Max A. Woodbury. Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton, 1950. (Cited on p. 12.)

[393] Margaret H. Wright. *Numerical Methods for Nonlinearly Constrained Optimization.* PhD thesis, Stanford University, Stanford, CA, 1976. (Cited on p. 606.)

[394] Stephen J. Wright. *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, PA, 1997. (Cited on p. 609.)

[395] David M. Young. *Iterative methods for solving partial differential equations of elliptic type.* PhD thesis, Harward University, Cambridge, MA, 1950. (Cited on pp. 470, 473.)

[396] David M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971. (Cited on pp. 473, 476, 481, 541, 616.)

[397] David M. Young. A historical overview of iterative methods. *Computer Phys. Comm.*, 53:1–17, 1989. (Cited on p. 542.)

[398] Fuzhen Zhang, editor. *The Schur Complement and Its Application.* Number 4 in Numerical Methods and Algorithms. Springer-Verlag, New York, 2005. (Cited on p. 165.)

Index