Contents

Chapter 7

Direct Methods for Linear System

7.1 Introduction

The problem treated in this chapter is the numerical solution of a system of m linear equations in n variables,

$$Ax = b, \tag{7.1.1}$$

for $A \in \mathbf{R}^{m\times n}$ and one or possibly several right hand sides $b \in \mathbf{R}^m$. Systems of linear equations enters at some stage in almost every scientific computing problem. Often their solution is the dominating part of the work to solve the problem. Even the solution of a *nonlinear* problem is usually accomplished by solving a *sequence* of linear systems obtained, e.g., by Newton's method. Since algorithms for solving linear systems are perhaps the most widely used in scientific computing, it is of great importance that they are efficient and reliable.

Only in the case that A has full row and column rank, i.e., $\operatorname{rank}(A) = n = m$ does the linear system $Ax = b$ have a unique solution for all right hand sides b. Note that inaccuracy of the elements of A mean that the "numerical" rank may not be well defined. When $\operatorname{rank}(A) < n$ the system either has many solutions (is underdetermined) or no solution (is overdetermined). The treatment of such systems will be deferred to Chapter 8.

Two quite different classes of methods for solving systems of linear equations are of interest: **direct** methods and **iterative** methods. In a direct method the system is transformed by a sequence of elementary transformed into a system of simpler form, e.g., triangular or diagonal form, which can be solved in an elementary way. The most important direct method is Gaussian elimination.

Disregarding rounding errors, direct methods give the exact solution after a finite number of arithmetic operations. Iterative methods, on the other hand, compute a sequence of approximate solutions, which (assuming exact arithmetic) in the limit converges to the exact solution x. Iterative methods have the advantage that in general they only require a subroutine for computing the matrix-vector product Ax for any given vector x. Hence they may be much more efficient than direct methods when the matrix A is large and matrix-vector multiplication cheap. The

distinction is not sharp since iterative methods are usually applied to a so called preconditioned version of the system that may involve the solution of a sequence of simpler auxiliary systems by a direct method. Iterative methods and preconditioning techniques are treated in Chapter 11.

Many applications give rise to linear systems where the matrix has some special property that can be used to achieve savings in work and storage. In the important case is when A is symmetric positive definite about half the work and storage can be saved; see Section 7.4. If only a small fraction of the elements in A are nonzero the linear system $Ax = b$ 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 Section 7.6. Indeed, without the exploitation of sparsity many important problems would be intractable!

There are also some classes of structured matrices, which although not sparse, have a structure, which can be used to develop fast solution methods. One important example is Vandermonde matrices, which we have seen are related to polynomial interpolation. Other important examples of structured matrices are Toeplitz and Hankel matrices. In all these instances the n^2 elements in the matrix are derived from only $(n-1)$ quantities.

Numerical methods for linear systems are a good illustration of the difference between classical mathematics and practical numerical analysis. Even though the mathematical theory is simple and the algorithms have been known for centuries, decisive progress in the development of algorithms has been made during the last few decades. It is important to note that methods, which are perfectly acceptable for theoretical use, may be useless for the numerical solution. For example, the explicit determinant formula (Cramer's rule) for the inverse matrix and for the solution of linear systems of equations is extremely uneconomical except for matrices of order two or three, and matrices of very special structure.

Since critical details in the algorithms can influence the efficiency and accuracy in a way the beginner can hardly expect the reader is strongly advised to use the efficient and well-tested software available in the public domain; see Notes and References at the end of this chapter.

The emphasis in this chapter will be on algorithms for *real* linear systems, since these occur most commonly in applications. However, the algorithms can readily be generalized to the complex case.

7.2 Gaussian Elimination and LU Factorization

7.2.1 Linear Systems

It is assumed that the reader is already familiar with the elementary properties of finite dimensional vector spaces and matrix algebra. We refer to Section 1.6 for a review of notations and definitions. In particular, we follow i n this book a convention introduced by Householder and use capital letters (e.g. A, B) to denote matrices. The corresponding lower case 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.

Consider a linear system $Ax = b$, where $A \in \mathbf{R}^{m\times n}$, $x \in \mathbf{R}^n$ and $b \in \mathbf{R}^m$ are column vectors, i.e.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, \tag{7.2.1}$$

We recall that if $r = \text{rank}\,(A)$ then A has r linearly independent columns and the same number r of linearly independent rows. The **range** of A is a subspace of $\mathbf{R}^m$ of dimension r denoted by $\mathcal{R}(A)$:

$$\mathcal{R}(A) = \{y \in \mathbf{R}^m |\ y = Ax,\ x \in \mathbf{R}^n\}. \tag{7.2.2}$$

The **null space** $\mathcal{N}(A)$ of A is a subspace of $\mathbf{R}^n$ of dimension $n - r$:

$$\mathcal{N}(A) = \{x \in \mathbf{R}^n |\ Ax = 0\}. \tag{7.2.3}$$

There are three possibilities: the system (7.2.1) may have no solution, a unique solution, or an infinite set of solutions. If $b \in \mathcal{R}(A)$, or equivalently $\text{rank}\,(A,\ b) = \text{rank}\,(A)$, the system is said to be **consistent**. If $r = m$ then $\mathcal{R}(A)$ equals $\mathbf{R}^m$ and the system is consistent for all b. 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. It follows that if a solution to an inhomogeneous system $Ax = b$ exists, it is unique only if $r = n$, whence $\mathcal{N}(A) = \{0\}$.

In the following we will use some new concepts. 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

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

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

7.2.2 Gaussian Elimination

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.

The idea behind **Gaussian elimination**[1] is to use such elementary operations to eliminate the unknowns in the system $Ax = b$ in a systematic way, 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

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

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

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

If $a_{22}^{(2)} \neq 0$, we can next in a similar way eliminate x_2 from the last $(n-2)$ of these equations. After $k-1$ steps, $k \leq \min(m, n)$, of Gaussian elimination the matrix A has been reduced to the form

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

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 (7.2.4)

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

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

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

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

Collecting the first equation from each step, and setting $a_{ij}^{(1)} = a_{ij}$, $b_i^{(1)} = b_i$, we obtain an upper triangular system. We also have

$$\det(A) = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}. \tag{7.2.4}$$

Denoting the upper triangular matrix $A^{(n)} = U = (u_{ij})$ and the reduced right hand side $b^{[n]} = c$, the unknowns can be computed recursively from

$$x_n = c_n/u_{nn} \qquad x_i = \Big(c_i - \sum_{k=i+1}^{n} u_{ik} x_k\Big)/u_{ii}, \quad i = n-1, \ldots, 1. \tag{7.2.5}$$

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

We have seen that 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 and it seems the algorithm breaks down.

Example 7.2.1. Consider the system

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

For $\epsilon \neq 1$ this system is nonsingular and has the unique solution $x_1 = -x_2 = -1/(1-\epsilon)$. However, when $a_{11} = \epsilon = 0$ the first step in Gaussian elimination cannot be carried through. The remedy here is obviously to interchange the two equations, which directly gives an upper triangular system.

Suppose that in step k of Gaussian elimination we have $a_{kk}^{(k)} = 0$. (The equations may have been reordered in previous steps, but we assume that the notations have been changed accordingly.) If A is 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 $a_{ik}^{(k)}$, $i = k, \ldots, n$ 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. The important conclusion is that *any nonsingular system of equations can be reduced to triangular form by Gaussian elimination if appropriate row interchanges are used.*

Note that when rows are interchanged in A the same interchanges must be made in the elements of the right hand side, b. Note also that the determinant formula (7.2.4) must be modified to

$$\det(A) = (-1)^s a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}, \tag{7.2.6}$$

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

If $\operatorname{rank}(A) < n$ then it is possible that at some step k we have $a_{ik}^{(k)} = 0$, $i = k, \ldots, n$. If the entire submatrix $a_{ij}^{(k)}$, $i, j = k, \ldots, n$, 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 to upper **trapezoidal form**,

$$A^{(r)} = \left(\begin{array}{ccc|ccc} a_{11}^{(1)} & \cdots & a_{1r}^{(1)} & a_{1,r+1}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & \ddots & \vdots & & \vdots & \vdots \\ \vdots & & a_{rr}^{(r)} & a_{r,r+1}^{(r)} & \cdots & a_{rn}^{(r)} \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0, \end{array}\right) \qquad b^{(r)} = \left(\begin{array}{c} b_1^{(1)} \\ \vdots \\ b_r^{(r)} \\ \hline b_{r+1}^{(r+1)} \\ \vdots \\ b_m^{(r+1)}. \end{array}\right) \tag{7.2.7}$$

in $r = \operatorname{rank}(A)$ steps

From the reduced form (7.2.7) we can read off the rank r of A. The two rectangular zero blocks in $A^{(r)}$ have dimensions $(m-r) \times r$ and $(m-r) \times (n-r)$, respectively. We deduce the following:

1. The system $Ax = b$ is consistent if and only if $b_k^{(r+1)} = 0$, $k = r+1, \ldots, m$. We can assign arbitrary values to the last $n-r$ components of (the possibly permuted) solution vector x. The first r components are then uniquely determined and obtained using back-substitution with the nonsingular triangular matrix in the upper left corner.

2. The system $Ax = b$ has a unique solution if and only if $r = m = n$.

7.2.3 Algorithms for Gaussian Elimination.

When implementing a matrix algorithm 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 use an informal programming language, which is sufficiently precise for our purpose but allows the suppression of cumbersome details. We illustrate these concepts on the back-substitution algorithm (7.2.5). We describe the back-substitution algorithm so that the solution vector x overwrites the data vector b

Algorithm 7.2.1 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 $Ux = b$:

$$\textbf{for } i = n : (-1) : 1$$
$$s := \sum_{k=i+1}^{n} u_{ik} b_k;$$

$$b_i := (b_i - s)/u_{ii};$$
end

Here := is the assignment symbol and $x := y$ means that the value of y is assigned to x. Note that in order to minimize round-off errors b_i is added *last* to the sum; compare the error bound (2.4.3).

Another possible sequencing of the operations in Algorithm 7.2.3:

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

Here the elements in U are accessed column-wise instead of row-wise as in the previous algorithm. Such differences can influence the efficiency when implementing matrix algorithms, see Problem 7.3.5.

We will often use the concept of a **flop**, 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.[2] With this notation solving a triangular system requires $\frac{1}{2}n^2$ flops.

When the matrix A is square and of a full rank Gaussian Elimination can be described as follows:

Algorithm 7.2.2 Gaussian Elimination; square case.

Given a matrix $A = A^{(1)} \in \mathbf{R}^{n\times n}$ and a vector $b = b^{(1)} \in \mathbf{R}^n$, the following algorithm reduces the system $Ax = b$ to the upper triangular form, provided that the pivotal elements $a_{kk}^{(k)} \neq 0$, $k = 1, \ldots, n$:

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

[2] Not that in several recent textbooks (e.g., Higham [11, 1996]) a flop is instead defined as a floating point add *or* multiply, doubling all the flop counts in this book.

end

end

We remark that no extra memory space is needed to store the multipliers. When l_{ik} is computed the element $a_{ik}^{(k)}$ is put equal to zero, so the multipliers can be stored in the lower triangular part of the matrix. Note also that if the multipliers l_{ik} are saved, then the operations on the right hand side b can be carried out at a later stage. This observation is important in that it shows that *when solving a sequence of linear systems*

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

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

From Algorithm 7.2.3 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

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

for A and each right hand side respectively. Comparing with the approximately $\frac{1}{2}n^2$ 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.* This conclusion may not be true for banded and sparse systems, see Sections 7.3 and 7.6.[3]

Algorithm 7.2.3 for Gaussian Elimination algorithm has three nested loops. It is possible to reorder these loops in $3 \cdot 2 \cdot 1 = 6$ ways. In each of those version the operations does the basic operation

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

and only the ordering in which they are done differs. 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 7.2.3 (see Problem 5).

7.2.4 Pivoting and Stability

In principle, the reduced trapezoidal form (7.2.7) obtained by Gaussian elimination yields the rank of a the matrix A, and also answers the question whether the given

[3] Flop counts like these are meant only as a rough appraisal of the work and one should not assign too much meaning to their precise value. On modern computer architectures the rate of transfer of data between different levels of memory often limits the actual performance.

system is consistent or not. However, this is the case only if when exact arithmetic is used and it can be decided when pivot elements and elements in the transformed right hand side are zero or nonzero. In floating point calculations the same decisions turn out to be surprisingly difficult! For example, note that a zero pivot in exact arithmetic will almost invariably be polluted by rounding errors in such a way that it equals some small nonzero number. Unfortunately there is no general rule, which can be used to decide when a pivot should be taken to be zero. What tolerance to use in such a test should depend on the context. In order to treat this question in a satisfactory way we need the concepts of **numerical rank** and **pseudoinverse**, which will be introduced in Section 8.??.

Another question, which we have to leave unanswered for a while, concerns underdetermined and overdetermined systems, which arise quite frequently in practice! Underdetermined systems typically arise when there are more parameters than needed to span the right hand side. Then, the question is, which of these solutions should we pick? On the other hand, overdetermined systems arise when there is more data than needed to determine the solution. Then the system usually is inconsistent and has no solution. Now the question is, how to find a solution, which in some sense best approximates the right hand side? These questions are related, and both require the use of *orthogonal* transformations. This topic is again deferred to Chapter 8.

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.

To ensure the numerical stability in Gaussian elimination it will, in general, be necessary to perform row (and/or column) interchanges *not only when a pivotal element is exactly zero, but also when it is small.*

In **partial pivoting** the pivot is taken as the largest element in magnitude in the unreduced part of the kth column.

Partial Pivoting. Choose r as the smallest integer for which

$$|a_{rk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}| \tag{7.2.8}$$

and interchange rows k and r.

If $A \in \mathbf{R}^{m\times n}$ and rank $(A) < n$ then Gaussian elimination with partial pivoting may stop prematurely. In **complete pivoting** the element of largest magnitude in the whole unreduced part of the matrix is chosen as pivot.

Complete Pivoting. Choose r and s as the smallest integers for which

$$|a_{rs}^{(k)}| = \max_{k\le i,j\le n} |a_{ij}^{(k)}| \tag{7.2.9}$$

and interchange rows k and r and columns k and s.

Both of the above pivoting strategies ensure that multipliers in Gaussian elimination satisfy the inequality $|l_{ik}| \le 1$ and it will be shown in Section 7.7.6 that ensure numerical stability in Gaussian elimination. The growth of elements in the reduced matrices is usually measured by the **growth ratio**

$$g_n = \max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}|. \tag{7.2.10}$$

For partial and complete pivoting we have

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

and the bound $g_n \le 2^{n-1}$ follows by induction. For partial pivoting this bound is the best possible, and can be attained for special matrices, see Section 7.7.6. However, in practice large growth of elements in partial pivoting is rare. We quote Wilkinson [20, 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.

Except for linear systems arising from certain two-point boundary value problems, no reports in the literature of experiences contrary to those related by Wilkinson have appeared.

For complete pivoting a much better bound can be proved, and in practice the growth very seldom exceeds n. However, complete pivoting involves a fairly high overhead since about as many arithmetic comparisons as floating point operations has to be performed.

For certain important classes of matrices a bound independent of n can be given for the growth ratio in Gaussian elimination *without pivoting.*

Theorem 7.2.1. *If Gaussian elimination is performed without pivoting the following bounds hold:*

(i) *If A is nonsingular and* **diagonally dominant**,

$$|a_{ii}| \ge \sum_{j\ne i} |a_{ij}|, \quad i = 1, \ldots, n.$$

then $g_n(A) \le 2$.

(ii) *If A is* **symmetric and positive definite**, $A^T = A$, $x^T Ax > 0$ *for all* $x \neq 0$, *then* $g_n(A) \leq 1$

(iii) *If A is* **totally positive (nonnegative)**, *i.e., if the determinant of every square submatrix of A is positive (nonnegative) then* $g_n(A) \leq 1$.

The proof of first two results are given in Sections 7.3.3 and 7.4.1 respectively.

We remark that the choice of pivots is influenced by the scaling of equations and unknowns, see Section 7.7.7. If, for example, the unknowns are physical quantities a different choices 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, but in practice this turns out to be less satisfactory. Likewise, an unsuitable column scaling can also make complete pivoting behave badly.

7.2.5 The LU Factorization

In this section we discuss an interpretation of Gaussian elimination (GE) as a matrix factorization. This is important since it shows how the result can be reused to solve several problems and also suggests several generalizations.[4]

For simplicity we first consider the case when $m = n$ and GE can be carried out without pivoting. We show that in this case GE provides a factorization of A into the product of a unit lower triangular matrix L and an upper triangular matrix U.

Consider a certain element a_{ij} during the elimination. We have

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

depending on whether a_{ij} lies on or above or below the principal diagonal. Thus the elements a_{ij} are transformed according to

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

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

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

This can also be written

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

[4] 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.

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

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

However, these equations are equivalent to the matrix equation $A = LU$, where $L = (l_{ik})$ and $U = (u_{kj})$ are lower and upper triangular matrices, respectively. The factorization in (7.2.13), which expresses A as a product of a lower and an upper triangular matrix, is called the **LU factorization** of A. Since the unit diagonal elements in L need not be stored it is possible to store the L and U factors in an array of the same dimensions as A.

It was shown in Section 7.2.4 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

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

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 $I_{ij}^{-1} = I_{ij}$.

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. The transpose P^T of a permutation matrix is therefore again a permutation matrix Since P is uniquely represented by the integer vector $(p_1, \ldots, p_n)$ it need never be explicitly stored.

If P is a permutation matrix then PA is the matrix A with its rows permuted and AP is A with its columns permuted. 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 $I_{i_p,j_p}^{-1} = I_{i_p,j_p}$, we have

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

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

Assume that in the kth step, $k = 1, \ldots, n-1$, 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 $k \leftrightarrow p_k$, $k = 1, \ldots, n-1$, 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

$$PA = LU, \tag{7.2.13}$$

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

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

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

$$Ly = Pb, \qquad Ux = y, \tag{7.2.14}$$

which involves about $2 \cdot \frac{1}{2}n^2 = n^2$ 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 $Ux = y$.

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

$$U^T c = b, \qquad L^T \tilde{x} = c. \tag{7.2.15}$$

We then obtain $x = P^{-1}\tilde{x}$ by applying the interchanges $k \leftrightarrow p_k$, in reverse order $k = n-1, \ldots, 1$ 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!

Similarly it can be shown that in the general case, if $\operatorname{rank}(A) = r$, the matrix $P_r A P_c \in \mathbf{R}^{m\times n}$ can be factored in 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

$$P_r A P_c = LU = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \end{pmatrix}, \tag{7.2.16}$$

where the matrices L_{11} and U_{11} are non-singular and

$$L = \left(\begin{array}{cccc} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{r1} & l_{r2} & \cdots & 1 \\ \hline l_{r+1,1} & l_{r+1,2} & \cdots & l_{r+1,r} \\ \vdots & \vdots & \cdots & \vdots \\ l_{m,1} & l_{m,2} & \cdots & l_{m,r} \end{array}\right), \tag{7.2.17}$$

$$U = \left(\begin{array}{cccc|ccc} u_{11} & u_{12} & \cdots & u_{1r} & u_{1,r+1} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2r} & u_{2,r+1} & \cdots & u_{2n} \\ & & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & & u_{rr} & u_{r,r+1} & \cdots & u_{rn} \end{array}\right) \tag{7.2.18}$$

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

To solve the system

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

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

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

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

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

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

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

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

7.2.6 Matrix Representation of Gaussian Elimination

In this section we will show how the reduction of a matrix to triangular form by Gaussian elimination can be expressed entirely in matrix notations. This way of looking at Gaussian elimination was first systematically exploited by Wilkinson [20, 1965]. It has the advantage that it suggests ways of deriving other matrix factorization, for example the QR decomposition, see Section 8.4. It will also provide an independent proof of the Theorem 7.2.2. We first introduce some useful concepts.

Elementary elimination matrices, are lower triangular matrices

$$L_j = I + l_j e_j^T = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{j+1,j} & 1 & & \\ & & \vdots & & \ddots & \\ & & l_{n,j} & & & 1 \end{pmatrix}, \tag{7.2.19}$$

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

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

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

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

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

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

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:

$$L_j L_k = I + l_j e_j^T + l_k e_k^T, \quad j \leq k. \tag{7.2.20}$$

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 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

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

this will zero out the elements under the main diagonal in the first column

$$A^{(2)} = L_1^{-1} P_1 A = \left(\begin{array}{c|ccc} \tilde{a}_{11} & \tilde{a}_{12} & \cdots & \tilde{a}_{1,n} \\ \hline 0 & \tilde{a}_{12}^{(2)} & \cdots & \tilde{a}_{1,n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{a}_{n2}^{(2)} & \cdots & \tilde{a}_{n,n}^{(2)} \end{array}\right)$$

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 $l_{i2} = \tilde{a}^{(2)}_{i2}/\tilde{a}^{(2)}_{22}$, $i = 3, \ldots, n$. Continuing, we have after $n-1$ steps reduced A to upper triangular form

$$U = L_{n-1}^{-1} P_{n-1} \cdots L_2^{-1} P_2 L_1^{-1} P_1 A. \tag{7.2.21}$$

To see that (7.2.21) 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 (7.2.21) implies

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

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

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

Using the result in (7.2.20), 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.

We now consider the general case when $A \in \mathbf{R}^{m \times n}$ and $\operatorname{rank}(A) = r \le \min(m, n)$. Proceeding as before we have after r steps factorized AP_c in a product of a square lower triangular matrix

$$L = \tilde{L}_1 \tilde{L}_2 \ldots \tilde{L}_r \in \mathbf{R}^{m \times m}$$

and an upper triangular matrix $U \in \mathbf{R}^{m \times n}$. In block form the factorization reads

$$L = \begin{pmatrix} L_{11} & \\ L_{21} & I \end{pmatrix}, \qquad U = \begin{pmatrix} U_{11} & U_{12} \\ 0 & 0 \end{pmatrix}. \tag{7.2.22}$$

Clearly we can delete the last block column in L and the last block row in U. Then we recover the factorization (7.2.22).

7.2.7 Gauss–Jordan Elimination

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** we eliminate simultaneously the elements *above* the main diagonal. After $n-1$ steps the matrix A has then been transformed into a *diagonal* matrix containing the nonzero pivot elements. Gauss–Jordan elimination can be described by introducing the elementary matrices

$$M_j = \begin{pmatrix} 1 & & & l_{1j} & & & \\ & \ddots & & \vdots & & & \\ & & 1 & l_{j-1,j} & & & \\ & & & 1 & & & \\ & & & l_{j+1,j} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & l_{n,j} & & & 1 \end{pmatrix}. \tag{7.2.23}$$

If partial pivoting is carried out we can write, cf. (7.2.21)

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

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

$$A^{-1} = D^{-1} M_n M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1. \tag{7.2.24}$$

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 $\approx n^3/2$ 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 right hand side to obtain

$$x = A^{-1}b = D^{-1} M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1 b. \tag{7.2.25}$$

This requires n^2 flops. Note that no back-substitution is needed!

Gauss–Jordan elimination will give about the same numerical accuracy in the computed solution $\bar{x}$ as Gaussian elimination. However, for Gaussian elimination the relative size of the residual vector of the computed solution $\|b - A\bar{x}\|/\|A\|\,\|\bar{x}\|$ will be of order machine precision even when A is ill-conditioned, see Theorem 7.7.9. *This does not hold for the solution computed by Gauss–Jordan elimination.*

7.2.8 Compact Schemes for Gaussian Elimination

Using the equivalence of Gaussian elimination and LU factorization it is easy to see that it is possible to arrange the calculations so that the elements in L and U are determined directly. The resulting algorithm were of particular interest in the days of hand computations, since they do away with the necessity in Gaussian elimination to write down $\approx n^3/3$ intermediate results—one for each multiplication.

For simplicity we assume that any row or column interchanges on A have been carried out in advance. The matrix equation $A = LU$ written in component-wise form (see (7.2.12))

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

together with the normalization conditions $l_{ii} = 1$, $i = 1, \ldots, n$, can be thought of as $n^2 + n$ equations for the $n^2 + n$ unknown elements in L and U. We can solve these equations in n steps, $k = 1, \ldots, n$, where in the kth step we use the equations

$$a_{kj} = \sum_{p=1}^{k} l_{kp} u_{pj}, \quad j \ge k, \qquad a_{ik} = \sum_{p=1}^{k} l_{ip} u_{pk}, \quad i > k \tag{7.2.26}$$

to determine the kth *row* of U and the kth *column* of L.

Algorithm 7.2.3 Compact LU Factorization.

for $k = 1 : n$
  for $j = k : n$
    $$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj};$$
  end
  for $i = k + 1 : n$
    $$l_{ik} = \Big(a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk} \Big) / u_{kk};$$
  end
end

Since the LU factorization is unique the compact algorithm produces the same factors L and U as Gaussian elimination. In fact, successive partial sums in the equations (7.2.26) are the elements $a_{ij}^{(k)}$, $j > k$, in Gaussian elimination. It follows that if each term in (7.2.26) 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. Algorithm 7.2.3 is usually referred to as Doolittle's algorithm. In Crout's algorithm the upper triangular matrix U is normalized to have a unit diagonal, $u_{ii} = 1$, $i = 1, \ldots, n$.

It is possible to sequence the computations in Doolittle's and Crout's algorithms in many different ways. Indeed any element in $(L\backslash U)$ can be computed as soon as the corresponding elements in L to the left and in U above have been deter-

mined. For example, three possible orderings are schematically illustrated below,

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

Here the entries indicate in which step a certain element l_{ij} and r_{ij} is computed, so the first example corresponds to the ordering in the algorithm given above. (Compare the comments after Algorithm 7.2.3.) A reason why one of the first two of these orderings may be preferred is they can be combined with partial pivoting in a straightforward way. Note that it is *not* easy to do complete pivoting with any of these methods.

7.2.9 Inverse Matrices

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 actually compute* A^{-1}. Nevertheless, there are some applications where A^{-1} is required, and we discuss below several methods for computing the matrix inverse.

The work required to compute A^{-1} is about n^3 flops, i.e., three times greater than for computing the LU factorization. (If A is a band matrix, then the savings are much more spectacular; see Section 7.3.) However, the matrix vector multiplication $A^{-1}b$ requires n^2 flops, which 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.)

The computed vector $A^{-1}b$ is usually less accurate than the approximate solution computed from the LU factorization. Moreover, using LU factorization by Gaussian elimination the residual vector of the computed solution will be of order machine precision even when A is ill-conditioned (see remarks to Theorem 7.7.9)! For methods which,as the Gauss–Jordan method, compute an approximation Z to A^{-1}, the residual $b - A(Zx)$ will in general *not be small.*

The inverse is sometimes needed in its own right—for example, in order to obtain estimates of the covariances in regression analysis. Even in such applications, however, usually only certain elements of A^{-1} are needed and not the whole inverse matrix.

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. Two alternative ways to obtain error estimates are the use of a condition estimator (Section 7.7.5) or iterative improvement (Section 7.7.8).

We briefly outline some methods to compute the inverse matrix $X = A^{-1}$. First assume that a factorization $A = LU$ has been computed. From $AX =$

$(LU)X = I$ we have, partitioning $X = (x_1, \ldots, x_n)$ and $I = (e_1, \ldots, e_n)$ by columns,

$$Ax_j = L(Ux_j) = e_j, \quad j = 1, \ldots, n. \tag{7.2.27}$$

Hence the columns of A^{-1} are obtained by solving n linear systems where the right-hand sides equal the columns in the unit matrix. The number of operations needed for computing A^{-1} by (7.2.27) seems to be $n^3/3 + nn^2 = 4n^3/3$. However, by taking the zeros in the right-hand sides into account this number can be reduced to n^3, see Problem 10. If row interchanges have been performed during the LU factorization, we have $PA = LU$, where $P = P_{n-1} \cdots P_2 P_1$ where P_k are transposition matrices. Then $A^{-1} = (LU)^{-1}P$. Hence to get A^{-1} we must perform the interchanges in *reverse order on the columns* of $(LU)^{-1}$.

A second method uses the relation $A^{-1} = (LU)^{-1}P^T = U^{-1}L^{-1}P^T$. The inverse of the triangular matrices are easy to compute. If we put $Y = (y_1, \ldots, y_n) = L^{-1}$ then y_j satisfies

$$Ly_j = e_j, \quad j = 1, \ldots, n,$$

and can be computed by forward-substitution. Since the vector e_j has $(j-1)$ leading zeros, also 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: for $j = 1, 2, \ldots, n$

$$y_{jj} = 1/l_{jj}, \qquad y_{ij} = (\delta_{ij} - \sum_{k=j}^{i-1} l_{ik} y_{kj})/l_{ii}, \quad i = j+1, \ldots, n, \tag{7.2.28}$$

where δ_{ij} denote the elements in the unit matrix. Note that the diagonal elements in L^{-1} are just the inverses of the diagonal elements of L and that 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: for $j = n, n-1, \ldots, 1$,

$$z_{jj} = 1/u_{jj}, \qquad z_{ij} = (\delta_{ij} - \sum_{k=i+1}^{j} u_{ik} z_{kj})/u_{ii}, \quad i = j-1, \ldots, 1. \tag{7.2.29}$$

Here Z can overwrite U in storage.

The number of flops required to compute L^{-1} or U^{-1} is approximately equal to $n^3/6$. Since the matrix multiplication $U^{-1}L^{-1}$ requires $n^3/3$ flops (show this!) the total work to compute A^{-1} by this method also is n^3 flops. If we note that $y_{jj} = 1/l_{jj} = 1$, $j = 1, \ldots, n$, and carefully sequence the computations so that L^{-1}, U^{-1} and A^{-1} can overwrite A then no extra storage is needed. A variation of this method is used in LINPACK, where once U^{-1} has been obtained by a column oriented algorithm, the system $XL = U^{-1}$ is solved to obtain $X = (LU)^{-1}$. All the methods described have roughly similar numerical properties.

The inverse can also be obtained from the Gauss–Jordan factorization. Using (7.2.25) where b is taken to be the columns of the unit matrix, we compute

$$A^{-1} = D^{-1}M_{n-1}^{-1}P_{n-1} \cdots M_2^{-1}P_2 M_1^{-1}P_1(e_1, \ldots, e_n).$$

Again n^3 flops are required if the computations are properly organized. However, this method has not as good numerical properties as the methods described above.

Review Questions

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. Show that if the kth diagonal entry of an upper triangular matrix is zero, then its first k columns are linearly dependent.
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?
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.
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.
4. Show that the permutation matrix $P = (e_n, \ldots, e_2, e_1)$, called the **exchange matrix**, has the property that PA (AP) reverses the order of the rows (columns). How is $B = PAP^T$ related to A?

Problems

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. In Algorithm 7.2.3 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 and hence successive elements in a row may be separated by a large increment which depends on the declared dimension of the array. By interchanging the two loops over the indices i and j in Algorithm 7.2.3, give a version of Gaussian elimination where the innermost loop involves a fixed column index and a varying row index.
3. What does M_j^{-1}, where M_j is defined in (7.2.23), look like?
4. 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 Banded Systems

7.3.1 Properties of Banded Matrices

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.

In banded linear systems each variable x_i is coupled by an equation only to a few other variables x_j such that $|j - i|$ is small. We make the following definition (note that it applies also to matrices which are not square):

Definition 7.3.1.
A matrix A is said to have **upper bandwidth** *r and* **lower bandwidth** *s if*

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

respectively. This means that the number of non-zero 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. An important, but hard, problem is to find optimal orderings that minimize the bandwidth. However, there are good heuristic algorithms that can be used in practice and give almost optimal results; see Section 7.6.3.

Several classes of band matrices that occur frequently have special names. Thus, a matrix for which $r = s = 1$ is called **tridiagonal**, if $r = 0$, $s = 1$ it is called (lower) **bidiagonal** etc. 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. In particular a matrix for which $r = s = 1$ is called **tridiagonal**. An example of a matrix with unsymmetric band structure is an a matrix with $s = 1$ ($r = 1$), which is called an upper (lower) **Hessenberg** matrix.

Band matrices are particularly well-suited for Gaussian elimination, since *the band structure is preserved if no pivoting is carried out.* As remarked previously pivoting is not needed for stability, e.g., when A is diagonally dominant.

Theorem 7.3.2. *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 (7.2.20). Assume that the first $k-1$ rows of U and columns of L have bandwidth r and s, that is, for $p = 1, \ldots, k-1$

$$l_{ip} = 0, \quad i > p+s, \qquad u_{pj} = 0, \quad j > p+r. \tag{7.3.1}$$

The proof is by induction in k. The assumption is trivially true for $k = 1$. Since $a_{kj} = 0$, $j > k+r$ we have from (7.2.12) and (7.3.1)

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj} = 0 - 0 = 0, \quad j > k+r.$$

Similarly it follows that $l_{ik} = 0$, $i > k+s$, which completes the induction step. □

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

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

Notice that except for a few elements in the initial and final rows, only the nonzero elements of A are stored. 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!

7.3.2 Gaussian Elimination for Banded Linear Systems

For a general band matrix Algorithm 7.2.3, Gaussian elimination without pivoting, should be modified as follows to operate only on nonzero elements:

Algorithm 7.3.1 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 $i = k+1, \ldots, \min(k+s, n)$
    $l_{ik} := a_{ik}^{(k)} / a_{kk}^{(k)}$;
    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

$$t = \begin{cases} nr(s+1) - \frac{1}{2}rs^2 - \frac{1}{6}r^3, & \text{if } r \le s; \\ ns(s+1) - \frac{2}{3}s^3, & \text{if } r = s; \\ ns(r+1) - \frac{1}{2}sr^2 - \frac{1}{6}s^3, & \text{if } r > s. \end{cases}$$

Whenever $r \ll n$ *or* $s \ll n$ this is much less than the $n^3/3$ flops required in the full case. For a Hessenberg matrix $r = n-1$, $s = 1$, and we have $t = \frac{1}{2}n(n+1)$ flops.

Analogous savings can be made in forward- and back-substitution. Let L and U be the triangular factors computed by Algorithm 7.4.2.

Algorithm 7.3.2 Banded Forward- and Back-Substitution.

The following algorithm computes the solution of the two banded triangular systems $Ly = b$ and $Ux = y$:

$$\textbf{for } i = 1 : n \quad y_i := b_i - \sum_{max(1,i-s)}^{i-1} l_{ij} y_j;$$

$$\textbf{end}$$

$$\textbf{for } i = n : (-1) : 1 \quad x_i := \Big(y_i - \sum_{j=i+1}^{\min(i+r,n)} u_{ij} x_j \Big) / u_{ii};$$

$$\textbf{end}$$

These algorithms require $ns - \frac{1}{2}s^2$ and $(n - \frac{r}{2})(r+1)$ 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.

We now consider the important special case of a **tridiagonal** matrix

$$A = \begin{pmatrix} d_1 & c_2 & & & \\ a_2 & d_2 & c_3 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & d_{n-1} & c_n \\ & & & a_n & d_n \end{pmatrix}. \tag{7.3.2}$$

Note that the $3n-2$ nonzero elements in A are conveniently stored in three vectors a, c, and d. *If the LU factorization of A exists*, then by Theorem 7.3.2 it has the form

$$A = LU = \begin{pmatrix} 1 & & & & \\ \gamma_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & \gamma_{n-1} & 1 & \\ & & & \gamma_n & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 & c_2 & & & \\ & \alpha_2 & c_3 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{n-1} & c_n \\ & & & & \alpha_n \end{pmatrix}. \tag{7.3.3}$$

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

$$\alpha_1 = d_1, \quad \gamma_k = a_k/\alpha_{k-1}, \qquad \alpha_k = d_k - \gamma_k c_k, \quad k = 2, \ldots, n. \tag{7.3.4}$$

Note that the elements γ_k and α_k can overwrite a_k and d_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

$$y_1 = f_1, \qquad y_i = f_i - \gamma_i y_{i-1}, \quad i = 2, \ldots, n, \tag{7.3.5}$$

$$x_n = y_n/\alpha_n, \qquad x_i = (y_i - c_{i+1}x_{i+1})/\alpha_i, \quad i = n-1, \ldots, 1. \tag{7.3.6}$$

The total number of flops is about $1.5n$ for the factorization and $2.5n$ for the solution.

It is important to note that *the inverse A^{-1} of a band matrix in general has no zero elements*. Hence one should never attempt to compute the full inverse of a banded matrix. Although banded systems of very large dimension can be efficiently solved, just to store the elements in A^{-1} may be infeasible. For example, if $n = 10^4$, then A^{-1} will have $100 \cdot 10^6$ nonzero elements!

7.3.3 Diagonally Dominant Matrices

Many applications lead to linear systems where the matrix has dominating diagonal elements such that

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|, \quad i = 1, \ldots, n. \tag{7.3.7}$$

A matrix A, whose elements satisfy equations (7.3.7), is said to be **row diagonally dominant**.

Assume that the matrix $A = A^{(1)}$ is diagonally dominant and nonsingular. Then $a_{11} \neq 0$, and if we perform elimination without pivoting we have in the first stage,

$$|a_{ij}^{(2)}| \leq |a_{ij}^{(1)}| + |l_{i1}||a_{1j}^{(1)}|, \quad i,j \geq 2,$$

where

$$|l_{i1}| = |a_{i1}^{(1)}|/|a_{11}^{(1)}|. \tag{7.3.8}$$

Using this and equations (7.3.7) and (7.3.8)

$$\begin{aligned} |a_{ii}^{(2)}| &\geq |a_{ii}^{(1)}| - |l_{i1}||a_{1i}^{(1)}| \\ &\geq \sum_{j\neq i} |a_{ij}^{(1)}| - |l_{i1}|\Big(|a_{11}^{(1)}| - \sum_{j\neq 1,i} |a_{1j}^{(1)}|\Big) \\ &= \sum_{j\neq 1,i} \big(|a_{ij}^{(1)}| + |l_{i1}||a_{1j}^{(1)}|\big) \geq \sum_{j\neq 1,i} |a_{ij}^{(2)}|. \end{aligned}$$

Hence the diagonal elements dominate in exactly the same way in the reduced matrix $A^{(2)} = (a_{ij}^{(2)})$, which also must be nonsingular. It follows by induction that all matrices $A^{(k)} = (a_{ij}^{(k)})$, $k = 2, \ldots, n$ are diagonally dominant and nonsingular.

From equations (7.3.7) and (7.3.8) it further follows that for $i \geq 2$

$$\begin{aligned} \sum_{j=2}^{n} |a_{ij}^{(2)}| &\leq \sum_{j=2}^{n} |a_{ij}^{(1)}| + |l_{i1}| \sum_{j=2}^{n} |a_{1j}^{(1)}| \\ &\leq \sum_{j=2}^{n} |a_{ij}^{(1)}| + |l_{i1}||a_{11}^{(1)}| \\ &\leq \sum_{j=2}^{n} |a_{ij}^{(1)}| + |a_{i1}^{(1)}| = \sum_{j=1}^{n} |a_{ij}^{(1)}|, \end{aligned}$$

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

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

It follows that

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

where g_n is the growth ratio. This bound holds also for matrices which are *column* diagonally dominant, i.e., which satisfy

$$|a_{jj}| \geq \sum_{i\neq j} |a_{ij}|, \quad j = 1, \ldots, n. \tag{7.3.9}$$

For a proof see Wilkinson [19, pp.288–9]. *Hence for row or column diagonally dominant matrices Gaussian elimination without pivoting is stable.*

If equation (7.3.7) or (7.3.9) holds with *strict* inequality the matrix A is said to be **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 finally mention another useful result for diagonally dominant matrices.

Lemma 7.3.3. *Let A be strictly row diagonally dominant, and set*

$$\min_i \Big(|a_{ii}| - \sum_{j,j\neq i} |a_{ij}| \Big) = \alpha > 0.$$

Then A is nonsingular, and $\|A^{-1}\|_\infty \leq \alpha^{-1}$.
If A is strictly column diagonally dominant, then $\|A^{-1}\|_1 \leq \alpha^{-1}$.
If A is strictly diagonally dominant in both directions, then also $\|A^{-1}\|_2 \leq \alpha^{-1}$.

Proof. By the definition of a subordinate norm (see Section 1.6.6) we have

$$\frac{1}{\|A^{-1}\|_\infty} = \inf_y \frac{\|y\|_\infty}{\|A^{-1}y\|_\infty} = \inf_x \frac{\|Ax\|_\infty}{\|x\|_\infty} = \inf_{\|x\|_\infty=1} \|Ax\|_\infty.$$

Assume that equality holds in the definition of α for $i = k$. Then

$$\frac{1}{\|A^{-1}\|_\infty} = \inf_{\|x\|_\infty=1} \max_i \Big| \sum_j a_{ij}x_j \Big| \geq \inf_{\|x\|_\infty=1} \Big| \sum_j a_{kj}x_j \Big| = |a_{kk}| - \sum_{j,j\neq k} |a_{kj}| = \alpha.$$

The bound in the l_1-norm follows since $\|A\|_1 = \|A^T\|_\infty$. These bounds can be used in the inequality $\|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty}$ to show the bound in the l_2-norm. ∎

Review Questions

1. Let A be an n by n matrix with upper and lower bandwidth p and q respectively. Assuming that Gaussian elimination can be carried out without pivoting, what are the structures of the resulting L and U factors of A? What will in general be the structure of the inverses of L and U?

7.4 Symmetric Matrices

7.4.1 Symmetric Positive Definite Matrices

A square matrix A is called **symmetric**if its elements are symmetric about its main diagonal, i.e., $a_{ij} = a_{ji}$, or equivalently $A^T = A$. The product of two symmetric matrices is symmetric if and only if A and B commute, that is, $AB = BA$. If $A^T = -A$, then A is called **skew-symmetric**. A square matrix A is called **persymmetric** if it is symmetric about its antidiagonal, i.e., $a_{ij} = a_{n-j+1,n-i+1}$.

The matrix A is said to be **positive definite** if

$$(x, Ax) > 0, \quad x \in \mathbf{R}^n, \quad x \neq 0, \tag{7.4.1}$$

and **positive semidefinite** if $(x, Ax) \geq 0$, for all $x \in \mathbf{R}^n$. Otherwise it is called **indefinite**.

A symmetric matrix $A \in \mathbf{R}^{n\times n}$ has only $\frac{1}{2}n(n+1)$ independent elements. If A also is positive definite then, as we will show here, about half the operations and storage compared to the nonsymmetric case is needed in to carry out Gaussian elimination.

Lemma 7.4.1. *If Gaussian elimination can be carried trough without pivoting for a symmetric matrix A, then the sequence of reduced matrices $A = A^{(1)}, A^{(2)}, \ldots, A^{(n)}$ are all symmetric.*

Proof. Assume that $A^{(k)}$ is symmetric, for some k, where $1 \leq k < n$. Then by Algorithm 7.2.3 we have after the k-th elimination step

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

$k+1 \leq i, j \leq n$. This shows that $A^{(k+1)}$ *is also a symmetric matrix*, and the result follows by induction. □

From Lemma 7.4.1 it follows that in Gaussian elimination without pivoting only the elements in $A^{(k)}$, $k = 2, \ldots, n$, 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 $D = \mathrm{diag}(d_1, \ldots, d_n)$ such that

$$A = LU = LDL^T. \tag{7.4.2}$$

Algorithm 7.4.1 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
```

This algorithm requires approximately $n^3/6$ flops, which is only half as much as for the unsymmetric case.

Here, in the last line 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

$$Ly = b, \qquad L^T x = D^{-1} y. \tag{7.4.3}$$

The cost of solving these triangular systems is about n^2 operations.

Example 7.4.1. *It may not always 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!

A matrix $A \in \mathbf{R}^{n\times n}$ is **symmetric positive definite** (s.p.d.) if $A^T = A$ and (7.4.1) holds. Such matrices arise naturally in many applications. Positive definite (semidefinite) matrices have the following important property:

Lemma 7.4.2. *Let $A \in \mathbf{R}^{n\times n}$ be symmetric positive definite (semidefinite). Then any principal $p \times p$ submatrix*

$$\tilde{A} = \begin{pmatrix} a_{i_1 i_1} & \dots & a_{i_1 i_p} \\ \vdots & & \vdots \\ a_{i_p i_1} & \dots & a_{i_p i_p} \end{pmatrix} \in \mathbf{R}^{p\times p}$$

is positive definite (semidefinite). In particular, taking $p = 1$, all diagonal elements in A are positive (nonnegative).

Proof. Take

$$\tilde{x} = (\tilde{x}_{i_1}, \dots, \tilde{x}_{i_p})^T \in \mathbf{R}^p, \quad \tilde{x} \neq 0$$

and define $x \in \mathbf{R}^n$

$$x_k = \begin{cases} \tilde{x}_{i_j}, & \text{if } k = i_j, \quad j = 1, \dots, p; \\ 0, & \text{otherwise.} \end{cases}$$

Then by construction we have $x \neq 0$ and $\tilde{x}^T \tilde{A} \tilde{x} = x^T A x > 0$ (≥ 0). ∎

We will prove that Gaussian elimination without pivoting can be carried through with positive pivot elements if and only if A is s.p.d. (The same consideration apply to complex Hermitian matrices, but since the modifications necessary for this case are straightforward, we discuss here only the real case.) The *semidefinite and indefinite* symmetric case will be discussed in Sections 7.4.3 and 7.4.5.

Assume first that we are given a symmetric matrix A, for which Algorithm 7.4.1 yields a factorization $A = LDL^T$ with positive pivotal elements $d_k > 0$, $k = 1, \ldots, n$. Then for all $x \neq 0$ we have $y = L^T x \neq 0$ and

$$x^T A x = x^T L D L^T x = y^T D y > 0,$$

and it follows that A is positive definite.

The next theorem states that if A is a real, s.p.d. matrix then the (unique) LDL^T factorization has a positive diagonal.

Theorem 7.4.3.

Let the matrix $A \in \mathbf{R}^{n\times n}$ be symmetric and positive definite. Then there exists a unique unit lower triangular matrix L and a diagonal matrix D with positive elements such that

$$A = LDL^T, \quad D = \operatorname{diag}(d_1, \ldots, d_n),$$

Proof. The proof is by induction on the order n of A. The result is trivial if $n = 1$, since then $D = d_1 = A = a_{11} > 0$ 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 $d_1 = a_{11}$, $B = \tilde{A} - d_1^{-1} a a^T$. 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 7.4.2. The Hilbert matrix $H_n \in \mathbf{R}^{n\times n}$ with elements

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

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

$$D = \begin{pmatrix} 1 & & & \\ & \frac{1}{12} & & \\ & & \frac{1}{180} & \\ & & & \frac{1}{2800} \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & & & \\ \frac{1}{2} & 1 & & \\ \frac{1}{3} & 1 & 1 & \\ \frac{1}{4} & \frac{9}{10} & \frac{3}{2} & 1 \end{pmatrix}.$$

Theorem 7.4.3 also yields the following useful characterization of a positive definite matrix.

Theorem 7.4.4. 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 $A_k \in \mathbf{R}^{k\times k}$, $k = 1, 2, \ldots, n$, 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 $\det(A_k) > 0$, $k = 1, 2, \ldots, n$, if and only if all pivots are positive. However, from Lemma 7.4.2 it follows that this is the case if and only if A is positive definite. □

Theorem 7.2.1 (ii) gives a bound on the growth ratio for the s.p.d. case. To prove this result we first show the following:

Lemma 7.4.5. *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. Lemma 7.4.2 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. □

In Algorithm 7.4.1 the diagonal elements are transformed in the k:th step of Gaussian elimination according to

$$a_{ii}^{(k+1)} = a_{ii}^{(k)} - (a_{ki}^{(k)})^2/a_{kk}^{(k)} = a_{ii}^{(k)}\left(1 - (a_{ki}^{(k)})^2/(a_{ii}^{(k)}a_{kk}^{(k)})\right).$$

If A is positive definite so are $A^{(k)}$ and $A^{(k+1)}$. Using Lemma 7.4.5 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

$$\max_{i,j,k} |a_{ij}^{(k)}| = \max_{i,k} a_{ii}^{(k)} \leq \max_i a_{ii} = \max_{i,j} |a_{ij}|,$$

which implies that $g_n \leq 1$ and proves Theorem 7.2.1 (ii).

7.4.2 Cholesky Factorization

Let A be a symmetric positive definite matrix, and L a lower triangular matrix with positive diagonal elements such that

$$A = LL^T. \tag{7.4.4}$$

Then L is called the **Cholesky factor** of A and the factorization (7.4.4) is called the **Cholesky factorization** of A.[5] The Cholesky factorization can be obtained from the factorization in (7.4.2). Since $D > 0$ we have

$$A = LDL^T = (LD^{1/2})(LD^{1/2})^T, \quad D^{1/2} = \operatorname{diag}(\sqrt{d}_1, \ldots, \sqrt{d}_n), \tag{7.4.5}$$

and hence the Cholesky factor equals $LD^{1/2}$. The uniqueness of the Cholesky factor follows from the uniqueness of the factorization $A = LDL^T$.

The Cholesky factorization can be computed by symmetric Gaussian elimination taking $d_k = l_{kk} = (a_{kk}^{(k)})^{1/2}$ in Algorithm 7.4.1. If l_k denotes the vector of multipliers in the kth step, the reduced matrix is modified by a rank-one outer product

$$A^{(k+1)} = A^{(k)} - l_k l_k^T.$$

In analogy to the compact schemes for LU factorization (see Section 7.2.8) it is possible to arrange the computations so that the elements in the Cholesky factor $L = (l_{ij})$ are determined directly. The matrix equation $A = LL^T$ with L lower triangular can be written

$$a_{ij} = \sum_{k=1}^{j} l_{ik} l_{jk} = \sum_{k=1}^{j-1} l_{ik} l_{jk} + l_{ij} l_{jj}, \quad 1 \leq j \leq i \leq n. \tag{7.4.6}$$

This is $n(n+1)/2$ equations for the unknown elements in L. We remark that for $j = i$ this gives

$$l_{ij}^2 \leq \sum_{k=1}^{i} l_{ik}^2 = a_{ii} \leq \max_i a_{ii},$$

which shows that the elements in L are bounded in terms of the diagonal elements in A.

[5] 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.

If we solve for l_{ij} from the corresponding equation in (7.4.6), we obtain

$$l_{ij} = \Big(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}\Big)/l_{jj}, \quad i > j, \qquad l_{ii} = \Big(a_{ii} - \sum_{k=1}^{j-1} l_{ik}^2\Big)^{1/2}.$$

If properly sequenced, these equations can be used in a recursive fashion to compute the elements in L. For example the elements in L can be determined one row or one column at a time. The resulting algorithms require n square roots and approximately $n^3/6$ multiplications.

Algorithm 7.4.2 Cholesky Algorithm, Row-wise Order.

for $i = 1 : n$
  for $j = 1 : i-1$
    $l_{ij} = \Big(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}\Big)/l_{jj};$
  end
  $l_{ii} = \Big(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2\Big)^{1/2};$
end

The row-wise ordering has the advantage of giving the Cholesky factors of all leading principal submatrices of A. The algorithm for the column-wise order is obtained by reversing the two loops in the above code.

Algorithm 7.4.3 Cholesky Algorithm, Column-wise Order.

for $j = 1 : n$
  $l_{jj} = \Big(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\Big)^{1/2};$
  for $i = j+1 : n$
    $l_{ij} = \Big(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}\Big)/l_{jj};$
  end
end

The two versions of the Cholesky algorithm are numerically equivalent, i.e., they will compute the same factor L even taking rounding errors into account.

7.4.3 Positive Semidefinite Matrices.

Some applications lead to a linear systems where $A \in \mathbf{R}^{n\times n}$ is a symmetric positive semidefinite matrix ($x^TAx \geq 0\ \forall x \neq 0$) with $\text{rank}(A) = r < n$. One example is

rank deficient least squares problems; see Section 8.5. [6] In this case a Cholesky factorization still exists, but pivoting need to be incorporated.

In Algorithm 7.4.1, the outer product version, at each elimination step *the maximal diagonal element* of the reduced matrix is chosen as pivot. Note that all reduced matrices are positive semidefinite and therefore 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 equal to $r < n$ the resulting factorization has the form

$$P^T A P = L L^T, \qquad L = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \tag{7.4.7}$$

where P is a permutation matrix and $L_{11} \in \mathbf{R}^{r\times r}$ with positive diagonal elements. The linear system $Ax = b$, or $P^T A P (P^T x) = P^T b$, then becomes

$$L L^T \tilde{x} = \tilde{b}, \quad \tilde{x} = P^T x, \quad \tilde{b} = P^T b.$$

Setting $z = L^T \tilde{x}$ the linear system reads

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

From the first r equations we obtain $z = L_{11}^{-1}\tilde{b}_1$ and substituting this in the last $n - r$ equations we find that for the linear system $Ax = b$ to be consistent we must have

$$L_{21} L_{11}^{-1} \tilde{b}_1 - \tilde{b}_2 = \begin{pmatrix} L_{21} L_{11}^{-1} & -I \end{pmatrix} \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix} = 0.$$

Otherwise the system has no solution.

It remains to solve $L^T \tilde{x} = z$, which gives

$$L_{11}^T \tilde{x}_1 = z - L_{21}^T \tilde{x}_2.$$

For an arbitrarily chosen $\tilde{x}_2$ we can uniquely determine $\tilde{x}_1$ so that these equations are satisfied. This expresses the fact that, if consistent, the system has an infinite number of solutions. Finally the permutations are undone to obtain $x = P\tilde{x}$.

Rounding errors can cause negative elements to appear on the diagonal in the Cholesky algorithm even when A is positive semidefinite. Similarly, because of rounding errors the reduced matrix will in general be nonzero after r steps even when $\operatorname{rank}(A) = r$. The question arises when to terminate the Cholesky factorization of a semidefinite matrix. One possibility is to continue until

$$\max_{k\le i\le n} a_{ii}^k \le 0,$$

but this may cause unnecessary work in eliminating negligible elements. Further discussion of this aspect has to be postponed until Chapter 8.

[6] Another example is when the finite element method is applied to a problem where rigid body motion occurs.

7.4.4 Banded Symmetric Matrices

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 7.4.2 we easily derive the following banded version:

Algorithm 7.4.4 Band-Cholesky Algorithm, Row-wise Order.

$$
\begin{aligned}
&\textbf{for} i = 1 : n \\
&\quad \textbf{for } j = \max(1, i-r) : i-1 \\
&\qquad l_{ij} = \Big(a_{ij} - \sum_{k=\max\{1,i-r\}}^{j-1} l_{ik} l_{jk}\Big)/l_{jj}; \\
&\quad \textbf{end} \\
&\quad l_{ii} = \Big(a_{ii} - \sum_{k=\max\{1,i-r\}}^{j-1} l_{ik}^2\Big)^{1/2}; \\
&\textbf{end}
\end{aligned}
$$

If $r \ll n$ this algorithm requires about $\frac{1}{2}nr(r+3)$ flops and n square roots. We just need to store the lower triangular part of A in a $n \times (r+1)$ array.

An important special case is when A is symmetric positive definite and tridiagonal (cf (7.3.2))

$$
A = \begin{pmatrix}
d_1 & a_2 & & & \\
a_2 & d_2 & a_3 & & \\
 & \ddots & \ddots & \ddots & \\
 & & a_{n-1} & d_{n-1} & a_n \\
 & & & a_n & d_n
\end{pmatrix}. \tag{7.4.8}
$$

Then we can write the Cholesky factorization

$$
A = LDL^T, \quad D = \operatorname{diag}(\alpha_1, \ldots, \alpha_n), \tag{7.4.9}
$$

where L is as in (7.3.3), and the algorithm reduces to

$$
\alpha_1 = d_1, \quad \gamma_k = a_k/\alpha_{k-1}, \qquad \alpha_k = d_k - \gamma_k a_k, \quad k = 2, \ldots, n. \tag{7.4.10}
$$

Sometimes it is more convenient to write

$$
A = U^T D^{-1} U, \quad D = \operatorname{diag}(d_1, \ldots, d_n).
$$

In the scalar case U is given by (7.3.3) (with $c_k = a_k$), and the elements in U and D are computed from

$$
\alpha_1 = d_1, \qquad \alpha_k = d_k - a_k^2/\alpha_{k-1}. \quad k = 2, \ldots, n. \tag{7.4.11}
$$

Example 7.4.3. We recall that the inverse A^{-1} of a band matrix in general is full and has no zero elements. Consider the symmetric, positive definite tridiagonal matrix and its inverse

$$A = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \qquad A^{-1} = \begin{pmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 4 & 3 & 2 & 1 \\ 3 & 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Although the Cholesky factor L of A is bidiagonal the inverse A^{-1} is full. Note the regular structure of A^{-1}. It can be shown that for a symmetric, positive definite and tridiagonal matrix A, with $a_i \neq 0$, $i = 2, \ldots, n$, there exist two vectors $u, v \in \mathbf{R}^n$ such that the inverse can be written as

$$(A^{-1})_{ij} = u_i v_j, \quad i \leq j.$$

The vectors u and v can be determined up to a scaling factor from the first and last columns of A^{-1}, see Problem 5.

7.4.5 Symmetric Indefinite Systems

As shown by Example 7.4.1, the LDL^T factorization of a symmetric indefinite matrix, although efficient computationally, may not exist and can be unstable. 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, then we give up the savings of a factor one half in operation count and storage, which we achieved for positive definite matrices.

To enable a stable symmetric factorization for a real, symmetric indefinite matrix A we will allow also the use of 2×2 principal submatrices as pivots. It is quite straightforward to generalize Gaussian elimination to use a nonsingular submatrix as pivot. We illustrate this on the symmetric block 2×2 system

$$\begin{pmatrix} A & B \\ B^T & D \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}.$$

where A and D are square symmetric matrices. If A is nonsingular we obtain from the first block equation $x = A^{-1}(b - By)$. Multiplying by B^T and substituting in the last block equation we obtain the reduced system

$$(D - B^T A^{-1} B)y = c - B^T A^{-1} b.$$

In matrix form the operation performed on the block matrix can be expressed as

$$\begin{pmatrix} I & 0 \\ -B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ B^T & D \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & S \end{pmatrix}, \tag{7.4.12}$$

where $S = S^T = D - B^T A^{-1} B$, is the **Schur complement** of A. The block lower triangular matrix on the left hand side can trivially be inverted and we obtain the symmetric block factorization

$$\begin{pmatrix} A & B \\ B^T & D \end{pmatrix} = \begin{pmatrix} I & 0 \\ L & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & L^T \\ 0 & I \end{pmatrix}, \tag{7.4.13}$$

where $L = B^T A^{-1}$. Assume now that in the first stage of elimination the matrix $A_{11} \in \mathbf{R}^{2\times 2}$ is nonsingular, where

$$A = \begin{pmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{pmatrix}, \qquad A_{11} = \begin{pmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{pmatrix}.$$

Using A_{11} as pivot in block Gaussian elimination 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 inverse of the matrix A_{11} is

$$A_{11}^{-1} = \frac{1}{\delta_{12}} \begin{pmatrix} a_{22} & -a_{21} \\ -a_{21} & a_{11} \end{pmatrix}, \quad \delta_{12} = \det(A_{11}) = a_{11}a_{22} - a_{21}^2. \tag{7.4.14}$$

The block A_{22} is transformed into the symmetric matrix $A_{22}^{(3)} = A_{22} - L_{21} A_{21}^T$, or component-wise

$$a_{ij}^{(3)} = a_{ij} - l_{i1}a_{1j} - l_{i2}a_{2j}, \quad 2 \le j \le i \le n. \tag{7.4.15}$$

Note that due to symmetry only the elements on and below the main diagonal of $A_{22}^{(3)}$ need to be computed.

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 (7.4.15), 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 problem in implementing the factorization above is to find a pivotal strategy that will give control of element growth without requiring too much search. The constraint of symmetry allows row and column permutations, which bring any element $d_r = a_{rr}$, or any 2×2 principal submatrix

$$D_{rs} = \begin{pmatrix} a_{rr} & a_{rs} \\ a_{sr} & a_{ss} \end{pmatrix}$$

to the pivot position. One pivotal strategy would be to estimate the size of d_r^{-1} and D_{rs}^{-1} over all possible r, s, choosing a pivot which minimizes such a quantity. However, since there are $O(n^2)$ possible 2×2 pivots the work required would increase the

total operation count to well over that required for unsymmetric LU factorization, which is unacceptable.

A more efficient strategy is the following. Search 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

$$\max(|a_{rr}|, |a_{ss}|) \geq \rho |a_{rs}|, \quad \rho = (\sqrt{17}+1)/8 \approx 0.6404.$$

The number ρ has here 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. With this choice the element growth can be shown to be bounded by

$$g_n \leq (1 + 1/\rho)^{n-1} < (2.57)^{n-1}. \tag{7.4.16}$$

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 (7.4.16) can be compared to the bound 2^{n-1}, which holds for Gaussian elimination with partial pivoting.

The bound for element growth in (7.4.16) can be achieved with less comparisons, using the following strategy due to **Bunch–Kaufman**. 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 $|a_{11}| \geq \rho\lambda$, 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 $|a_{11}| \geq \rho\lambda^2/\sigma$, then again take a_{11} as pivot, else if $|a_{rr}| \geq \rho\sigma$, 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 n^2 comparisons are needed in all.

Whenever a 2×2 pivot is used, we have

$$a_{11}a_{rr} \leq \rho^2 |a_{1r}|^2 < |a_{1r}|^2.$$

It follows that the corresponding 2×2 block in D has a negative determinant $\delta_{1r} = a_{11}a_{rr} - a_{1r}^2 < 0$. From Theorem 7.4.4 we know that this cannot occur if A is positive definite. It follows that when A is positive definite all pivots chosen by the Bunch-Kaufman strategy will be 1×1.

Unfortunately, the algorithms discussed here does not in general preserve the band structure of the matrix.

Review Questions

1. What is the definition of a symmetric, positive definite matrix A? Can negative elements occur along the main diagonal in such a matrix? What simplifications occur in Gaussian elimination applied to a symmetric, positive definite matrix?
2. What is the Cholesky factorization of a symmetric, positive definite matrix? What is its relation to Gaussian elimination? Give an example of a symmetric matrix A for which the Cholesky decomposition does not exist.

Problems

1. Show that if A is symmetric positive definite so is its inverse A^{-1}.
2. 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. The Hilbert matrix $H_n \in \mathbf{R}^{n\times n}$ with elements

$$a_{ij} = 1/(i+j-1), \quad 1 \le i, j \le n,$$

   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 $(A - \bar{A})$.
4. (a) Let $A, B \in \mathbf{R}^{n\times n}$ have lower bandwidth r and s respectively. Show that the product AB has lower 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.

   (c) Show that if $R \in \mathbf{R}^{n\times n}$ is strictly upper triangular, then $R^n = 0$.
5. (a) Let $A \in \mathbf{R}^{n\times n}$ be a symmetric, tridiagonal matrix such that $\det(A_k) \neq 0$, $k = 1, \ldots, n$. 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 $\det(A_k)$, $k = 1, \ldots, n$.

   (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.

6. (a) Compute the Cholesky factorization of the matrix

$$A = \begin{pmatrix} 1 & 1 & & & \\ 1 & 2 & 1 & & \\ & 1 & 3 & 1 & \\ & & 1 & 4 & 1 \\ & & & 1 & 5 \end{pmatrix}.$$

(b) Compute the first and last column of the inverse A^{-1} and then determine two vectors u, v such that $(A^{-1})_{ij} = u_i v_j$ for $i \leq j$.

7. (a) Show that for $\lambda \geq 2$ it holds that $B = \mu L L^T$, 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 and equation system $Ax = b$, where the matrix A differs from B in the element (1,1),

$$A = B + \delta e_1 e_1^T, \quad \delta = \lambda - \mu, \quad e_1^T = (1, 0, \ldots, 0).$$

Show that the solution $x = A^{-1}b$ can be computed form

$$x = y - \gamma L^{-T} f, \quad \gamma = \delta(e_1^T y)/(\mu + \delta f^T f)$$

where y and f satisfies $\mu L L^T y = b$, $Lf = e_1$.

8. (a) Show that any matrix $A \in \mathbf{R}^{n \times n}$ can be written as the sum of a symmetric and a skew-symmetric part, $A = H(A) + S(A)$, where

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

(b) Show that A is positive definite if and only if its symmetric part $H(A)$ is positive definite.

9. Boundary value problems, where the solution is subject to *periodic boundary conditions*, often lead to matrices of the form

$$A = \begin{pmatrix} d_1 & a_2 & & & a_1 \\ a_2 & d_2 & a_3 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & d_{n-1} & a_n \\ a_1 & & & a_n & d_n \end{pmatrix},$$

which is real symmetric and tridiagonal except for the corner elements a_1. Show that

$$A = T + \tau u u^T, \quad u^T = (1, 0, \ldots, 0, 1).$$

where T is a certain symmetric, tridiagonal matrix. Determine τ and T.

(b) If the matrix A is also positive definite it has a factorization $A = LDL^T$, where L has the form

$$L = \begin{pmatrix} 1 & & & & \\ \gamma_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & \gamma_{n-1} & 1 & \\ \delta_1 & \cdots & \delta_{n-2} & \gamma_n & 1 \end{pmatrix}.$$

Derive an algorithm for computing L by modifying the algorithm (7.4.10).

7.5 Block Matrix Algorithms

7.5.1 Partitioning

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

$$A = \begin{matrix} \\ p_1 \{ \\ p_2 \{ \\ \vdots \\ p_M \{ \end{matrix} \begin{matrix} \begin{matrix} q_1 & q_2 & \cdots & q_N \end{matrix} \\ \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M1} & A_{M2} & \cdots & A_{MN} \end{pmatrix} \end{matrix}, \tag{7.5.1}$$

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. Often, it is suggested by the structure of the underlying problem. For square matrices the most important partitionings are those for which $M = N$, and $p_I = q_I$. In this case the diagonal blocks $A_{II}, I = 1, \ldots, N$ are square matrices.

The convenience of block matrices lies in that the operations of addition and multiplication can be performed by treating the blocks A_{IJ} as *non-commuting scalars* and applying the usual definitions. Therefore many algorithms defined for matrices with scalar elements have a simple generalization to partitioned matrices. Of course the dimensions of the blocks must correspond in such a way that the operations can be performed. When this is the case, the matrices are said to be partitioned **conformally**.

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

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

Be careful to note the *order* of the factors in the products!

Many of the special types of matrices defined earlier extend to block matrices. For example

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}, \quad R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \tag{7.5.2}$$

are 2×2 block lower and upper triangular matrices, respectively. Similarly we can define block-diagonal matrices, block tridiagonal matrices etc. Note that for an upper block triangular matrix with square diagonal blocks $R_{II}, I = 1, \ldots, N$ we have

$$\det(R) = \det(R_{11}) \det(R_{22}) \cdots \det(R_{NN}), \tag{7.5.3}$$

Hence R 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.

It is sometimes convenient to partition a matrix before inversion. If L and R in (7.5.2) are nonsingular with square diagonal blocks, then their inverses are given by

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1} L_{21} L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}, \qquad R^{-1} = \begin{pmatrix} R_{11}^{-1} & -R_{11}^{-1} R_{12} R_{22}^{-1} \\ 0 & R_{22}^{-1} \end{pmatrix}, \tag{7.5.4}$$

which can be verified by forming the products $L^{-1}L$ and $R^{-1}R$. Expressions for the inverse of more general 2×2 block matrices are given in Section 7.5.2.

In some cases it may be desirable to avoid complex arithmetic when working with complex matrices. This can be achieved by replacing complex matrices and vectors by real ones of twice the order. Suppose that

$$A = B + iC, \qquad z = x + iy,$$

with real B, C, x and y. Then consider $\tilde{A}$ and $\tilde{z}$ defined by

$$\tilde{A} = \begin{pmatrix} B & -C \\ C & B \end{pmatrix}, \qquad \tilde{z} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

It is easy to verify the following rules

$$\widetilde{(Az)} = \tilde{A}\tilde{z}, \qquad \widetilde{(AB)} = \tilde{A}\tilde{B}, \qquad \widetilde{(A^{-1})} = (\tilde{A})^{-1},$$

etc. Thus we can solve complex valued problems using algorithms for the real case. However, this incurs a penalty in storage and arithmetic operations.

7.5.2 Schur Complements

Consider the block 2×2 linear system

$$\begin{pmatrix} A & B \\ C^T & D \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}.$$

where A and D are square matrices, and A nonsingular. The result of Gaussian elimination with A as block pivot can be expressed in matrix form as

$$\begin{pmatrix} I & 0 \\ -C^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ C^T & D \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & S \end{pmatrix}, \tag{7.5.5}$$

where

$$S = D - C^T A^{-1} B \tag{7.5.6}$$

is the the **Schur complement** of A. The block lower triangular matrix on the left hand side can trivially be inverted and we obtain the block triangular factorization

$$M = \begin{pmatrix} A & B \\ C^T & D \end{pmatrix} = \begin{pmatrix} I & 0 \\ C^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ 0 & S \end{pmatrix}. \tag{7.5.7}$$

Taking A to be the scalar $a_{11} \neq 0$ the above is precisely the first step in Gaussian elimination written in matrix form. In this case C and B are row vectors and the Schur complement matrix

$$S = D - \frac{1}{a_{11}} C^T B$$

equals D modified by a scaled outer product of two vectors.

From $M^{-1} = (LU)^{-1} = U^{-1}L^{-1}$ using the formulas in Section 7.5 for the inverses of 2×2 block triangular matrices we get the Schur–Banachiewicz inverse formula

$$M^{-1} = \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}C^T A^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}C^T A^{-1} & S^{-1} \end{pmatrix}. \tag{7.5.8}$$

Similarly, when D is assumed to be nonsingular we can factor M into a product of a block upper and a block lower triangular matrix

$$M = \begin{pmatrix} I & BD^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} T & 0 \\ C^T & D \end{pmatrix}, \qquad T = A - BD^{-1}C^T, \tag{7.5.9}$$

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,

$$M^{-1} = \begin{pmatrix} T^{-1} & -T^{-1}BD^{-1} \\ -D^{-1}C^T T^{-1} & D^{-1} + D^{-1}C^T T^{-1} BD^{-1} \end{pmatrix}. \tag{7.5.10}$$

If A and D are both nonsingular we obtain from the two triangular factorizations (7.5.7) and (7.5.9) using (7.5.3) $\det(M) = \det(A)\det(S) = \det(T)\det(D)$, and thus

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

7.5.3 Modified Linear Systems

Frequently it is required to solve a linear problem, where the matrix has been modified by a correction of low rank. In the solution of linear systems, let $A \in \mathbf{R}^{n\times n}$ we consider problems where A has been modified by a correction of rank one,

$$(A + \sigma uv^T)\hat{x} = b, \quad u, v \in \mathbf{R}^n, \tag{7.5.11}$$

where $\sigma \neq 0$. This linear system is related to the **bordered system** formed by appending to the matrix A a row v^T, a column u and a scalar $-1/\sigma$,

$$\begin{pmatrix} A & u \\ v^T & -\sigma^{-1} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \theta \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{7.5.12}$$

More generally we can consider a modification of rank p

$$(A + U\Sigma V^T)\hat{x} = b, \quad U, V \in \mathbf{R}^{n\times p}, \tag{7.5.13}$$

where Σ is a nonsingular diagonal matrix. This system is related to the block bordered system

$$\begin{pmatrix} A & U \\ V^T & -\Sigma^{-1} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \theta \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{7.5.14}$$

which can be verified by eliminating θ to get a linear system only involving $\hat{x}$.

If the solution $x = A^{-1}b$ has been computed by a direct method it is possible to solve these modified systems above very efficiently when $p \ll n$. For the special case of a rank one perturbation we obtain the simple formula

$$\det(A + \rho uv^T) = \det(A)(1 + \rho v^T A^{-1}u), \tag{7.5.15}$$

which is useful for the solution of modified eigenvalue problems.

By equating the $(1,1)$ blocks in the inverse M^{-1} in (7.5.8) and (7.5.10) we get the following useful relation

$$(A - BD^{-1}C^T)^{-1} = A^{-1} + A^{-1}B(D - C^TA^{-1}B)^{-1}C^TA^{-1}, \tag{7.5.16}$$

which is often called the **Woodbury formula**. This formula provides the inverse of A after it is modified by a matrix of rank p, and is very useful in situations where $p \ll n$. If we specialize this relation to the case $p = 1$, corresponding to the bordered system (7.5.11) we get the **Sherman-Morrison formula**

$$(A + \rho uv^T)^{-1} = A^{-1} - \alpha A^{-1}uv^TA^{-1}, \qquad \alpha = 1/(\rho^{-1} + v^TA^{-1}u). \tag{7.5.17}$$

This formula can be used to cheaply compute the inverse of $A + \rho uv^T$ from the inverse of A. Such formulas are called updating formulas and are widely used in many contexts. For the solution to the modified system (7.5.11) we obtain

$$(A + \rho uv^T)^{-1}b = A^{-1}b - \alpha A^{-1}uv^TA^{-1}b, \tag{7.5.18}$$

which can also be written as

$$\hat{x} = x - \beta w, \qquad \beta = v^Tx/(\rho^{-1} + v^Tw), \qquad w = A^{-1}u. \tag{7.5.19}$$

Hence the solution $\hat{x}$ can be obtained from x by solving the system $Aw = u$, and *computing A^{-1} can be avoided*. This is sometimes computationally very advantageous, see, e.g., Section 7.2.9 and 7.3. The formula (7.5.17) can be generalized for the modified system in (7.5.13) to yield

$$(A + U\Sigma V^T)^{-1}b = A^{-1}b - A^{-1}U(\Sigma^{-1} + V^TA^{-1}U)^{-1}V^TA^{-1}b. \tag{7.5.20}$$

7.5.4 Basic Linear Algebra Subprograms

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. One important key to this is to formulate algorithms using the Basic Linear Algebra Subprograms (BLAS).

The BLAS have also become an aid to clarity, portability and modularity in modern software for matrix computations. The original set of BLAS introduced in 1973 identified frequently occurring vector operations in linear algebra such as scalar product, adding of a multiple of one vector to another, etc. These were efficient for scalar computers but since they just perform $O(n)$ operations on $O(n)$ data items, where n is the dimension of the vectors, they do not lead to efficient use of modern high-performance computers. For example, a matrix vector product has to be implemented as n subroutine calls.

The original BLAS are now known as Level 1. Level 2 BLAS were designed for matrix-vector operations that occur frequently. These involve $O(mn)$ scalar operations, where m and n are the dimensions of the matrix involved. The following three types of basic operations are covered:

(a) Matrix-vector products

$$y = \alpha Ax + \beta y,$$
$$y = \alpha A^T x + \beta y,$$
$$y = \alpha A^H x + \beta y,$$

and

$$x = Tx, \quad x = T^T x, \quad x = T^H x,$$

where α and β are scalars, x and y are vectors, A a matrix and T an upper or lower triangular matrix.

7.5.5 Blocked Algorithms for Factorization

It is possible to reorganize the LU factorization so that it works on block matrices with square diagonal blocks. Let $A = (A_{IK})$ be a block matrix and let L and U be partitioned conformally. By equating blocks in the product $A = LU$, where

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{pmatrix}, \tag{7.5.21}$$

Algorithm 7.2.8 can be extended as follows:

Algorithm 7.5.1 Block LU Factorization.

for $K = 1 : N$

$$L_{KK}U_{KK} = A_{KK} - \sum_{P=1}^{K-1} L_{KP}U_{PK};$$

for $J = K + 1 : N$

$$U_{KJ} = L_{KK}^{-1}\Big(A_{KJ} - \sum_{P=1}^{K-1} L_{KP}U_{PJ}\Big);$$

$$L_{JK} = \Big(A_{JK} - \sum_{P=1}^{K-1} L_{JP}U_{PK}\Big)U_{KK}^{-1};$$

end

end

Here it is assumed that the LU-decompositions of the modified diagonal blocks A_{KK} exist. In practice the code must be modified to allow for row interchanges. Note that the main work now lies in the matrix multiplications when computing the block matrices in parenthesis. The inverse of the triangular matrices L_{KK}^{-1} and R_{KK}^{-1} are not formed but U_{KJ} and L_{JK} computed by solving triangular solves.

This way of organizing the LU decomposition is more efficient for computers with a hierarchical memory architecture because the main work is performed as matrix-matrix multiplications. This makes it possible to perform more arithmetic operations while data is kept in registers and fast cache memory. Block algorithms are also useful when the matrix A is too large to be stored in the main memory.

As with the scalar algorithms there are many possible ways of sequencing the block factorization. The variant above computes in the Kth major step the Kth block row of U and the Kth block column of L. Another possibility which has advantages is to compute in the Kth major step the Kth block column of L and U.

for $K = 1 : N$

$$L_{KK}U_{KK} = A_{KK} - \sum_{P=1}^{K-1} L_{KP}U_{PK};$$

for $J = K + 1 : N$

$$L_{JK} = \Big(A_{JK} - \sum_{P=1}^{K-1} L_{JP}U_{PK}\Big)U_{KK}^{-1};$$

end

for $J = 1 : K - 1$

$$U_{JK} = L_{KK}^{-1}\Big(A_{JK} - \sum_{P=1}^{K-1} L_{JP}U_{PJ}\Big);$$

end

end

In this variant at step K only the Kth block column of A is accessed. This is advantageous from the standpoint of data access. This variant is sometimes called **left-looking** and the previous version **right-looking**, where left and right refer to the data access.

Analogous to the Block LU Algorithm in Section 7.5.5 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 7.5.2 Block-Cholesky Algorithm.

for $J = 1 : N$

$$L_{JJ}L_{JJ}^T = A_{JJ} - \sum_{K=1}^{J-1} L_{JK}L_{JK}^T;$$

for $I = J+1 : N$

$$L_{IJ} = \Big(A_{IJ} - \sum_{K=1}^{J-1} L_{IK}L_{JK}^T\Big)(L_{JJ}^T)^{-1};$$

end

end

Note that the diagonal blocks L_{JJ} are obtained by computing the Cholesky factorizations of matrices of smaller dimensions. The right multiplication with $(L_{JJ}^T)^{-1}$ in the computation of L_{JK} is performed by solving the triangular equations of the form $L_{JJ}L_{IJ}^T = S^T$. The main work in the block Cholesky algorithm is in the matrix multiplications $L_{IK}L_{JK}$.

For a symmetric positive definite **block-tridiagonal** matrix the recursion (7.4.11) is easily generalized to compute the following block-factorization:

$$A = U^T D^{-1} U, \quad D = \operatorname{diag}(\Sigma_1, \ldots, \Sigma_n),$$

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

$$\Sigma_1 = D_1, \qquad \Sigma_k = D_k - A_k \Sigma_{k-1}^{-1} A_k^T, \quad k = 2, \ldots, N. \tag{7.5.22}$$

To perform the operations with Σ_k^{-1}, $k = 1, \ldots, N$ the Cholesky factorization of these matrices must be computed. 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.

In the above block versions of LU and Cholesky factorization it is assumed that the block sizes are determined in advance. However, this is by no means necessary. A more flexible way is to advance the computation by deciding at each block size the size of the current pivot block. The corresponding block formulation then uses a 3×3 block structure, but the partitioning changes after each step. Suppose that a partial LU factorization of a matrix A has been obtained so that

$$PA = \begin{pmatrix} L_{11} & & \\ L_{21} & I & \\ L_{31} & 0 & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & \tilde{A}_{22} & \tilde{A}_{23} \\ & A_{32} & A_{33} \end{pmatrix}.$$

To advance the factorization compute the LU factorization with row pivoting

$$P_2 \begin{pmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{pmatrix} = \begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} U_{22},$$

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 We then solve for U_{23} and update A_{33} by

$$L_{22}U_{23} = \tilde{A}_{23}, \qquad \tilde{A}_{33} = A_{33} - L_{32}U_{23}.$$

The factorization has now been advanced one step to become

$$P_2PA = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & A_{33} \end{pmatrix}.$$

We can now repartition so that the first two block-columns in L and the first two block-rows in U are joined. 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.

The above version is a right-looking algorithm. The corresponding left-looking algorithm goes as follows. Assume that we have computed

$$PA = \begin{pmatrix} L_{11} & & \\ L_{21} & I & \\ L_{31} & 0 & I \end{pmatrix} \begin{pmatrix} U_{11} & A_{12} & A_{13} \\ & A_{22} & A_{23} \\ & A_{32} & A_{33} \end{pmatrix}.$$

To advance the factorization we solve first a triangular system

$$L_{11}U_{12} = A_{12}$$

to obtain U_{12} and then 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 LU factorization with row pivoting

$$P_2 \begin{pmatrix} \tilde{A}_{22} \\ \tilde{A}_{32} \end{pmatrix} = \begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} U_{22},$$

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

$$P_2 P A = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & A_{13} \\ & U_{22} & A_{23} \\ & & A_{33} \end{pmatrix}.$$

Note that in this version the blocks in the last block column of A are referenced only in the pivoting operation, but this can be postponed.

7.5.6 Recursive Algorithms

In the 2×2 case the block LU factorization Algorithm 7.5.5 is obtained by equating

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}. \tag{7.5.23}$$

We get

$$\begin{aligned} &L_{11}U_{11} = A_{11}, \\ &L_{21} = A_{21}U_{11}^{-1}, \qquad U_{12} = L_{11}^{-1}A_{12}, \\ &\tilde{A}_{22} = A_{22} - L_{21}U_{12}, \\ &L_{22}U_{22} = \tilde{A}_{22}. \end{aligned}$$

Hence the LU factorization of A can be reduced to the LU factorization of two smaller matrices A_{11} and $\tilde{A}_{22}$, two triangular solves with matrix right hand sides, and one matrix update $A_{22} - L_{21}U_{12}$.

Similarly for the Cholesky factorization equating

$$\begin{pmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{pmatrix}, \tag{7.5.24}$$

gives

$$\begin{aligned} &L_{11}L_{11}^T = A_{11}, \\ &L_{21}^T = L_{11}^{-1}A_{21}^T, \\ &L_{22}L_{22}^T = A_{22} - L_{21}L_{21}^T. \end{aligned}$$

It is possible to derive "divide and conquer" algorithms for the LU and Cholesky algorithms, by using the 2×2 block versions recursively.

Algorithm 7.5.3 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);
[n, n] = size(A);
if n ≠ 1
%RecursiveCholesky
   k = floor(n/2)
   L11 = rchol(A(1 : k, 1 : k));
   L21 = (L11^{-1} A(1 : k, k + 1 : n))';
   L22 = rchol(A(1 : k, 1 : k) − L21 ∗ L21');
   L = [L11 zeros(k, n − k); L21 L22];
else
   L = sqrt(A);
end;
```

This is not a toy algorithm, but can be developed into an efficient algorithm for parallel high performance computers!

An intriguing question is whether it is possible to multiply two matrices $A \in \mathbf{R}^{m\times n}$, and $B \in \mathbf{R}^{n\times p}$ in less than mnp (scalar) multiplications. The answer is yes! Strassen [1969] developed a fast algorithm for matrix multiplication based on the following method for multiplying 2×2 block matrices. Assume that m, n, p are even and partition each of A, B and the product $C \in \mathbf{R}^{m\times p}$, into four equally sized blocks. Then, as can be verified by substitution, the product $C = AB$ can be computed using the following formulas:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 + P_3 - P_2 + P_6 \end{pmatrix},$$

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 more expensive (n^3) than addition (n^2) this will lead to a saving in operations.

Strassen's algorithm can be used recursively. to multiply two square matrices of dimension $n = 2^k$. The number of multiplications is then reduced from n^3 to $n^{\log_2 7} = n^{2.807\cdots}$. (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 2. However, there is some loss of numerical stability compared to conventional matrix multiplication, see Higham [1990].

By using the block formulation recursively, and Strassen's method for the matrix multiplication it is possible to perform the LU factorization in $O(n^{\log_2 7})$ operations.

7.5.7 Kronecker Systems

Linear systems where the matrix is a Kronecker product 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 Let $A \in \mathbf{C}^{m\times n}$ and $B \in \mathbf{C}^{p\times q}$ be two matrices. Then the **Kronecker product** of A and B is the $mp \times nq$ block matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}. \tag{7.5.25}$$

We now state without proofs some elementary facts about Kronecker products. From the definition it immediately follows that From the definition (7.5.25) 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 relation which is not so obvious:

Lemma 7.5.1.

If the ordinary multiplications AC and BD are defined, then

$$(A\otimes B)(C\otimes D) = AC \otimes BD. \tag{7.5.26}$$

PROOF: See Lancaster and Tismenetsky [1985] Ch. 12.1. □

It follows that if A and B are square and non-singular, then $A \otimes B$ is nonsingular and

$$(A\otimes B)^{-1} = A^{-1} \otimes B^{-1}.$$

We now introduce a function closely related to the Kronecker product, which converts a matrix into a vector. For a matrix $C = (c_1, c_2, \ldots, c_n) \in \mathbf{R}^{m\times n}$ we define

$$\text{vec}\,(C) = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}. \tag{7.5.27}$$

It 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 7.5.2.
If $A \in \mathbf{R}^{m\times n}$, $B \in \mathbf{R}^{p\times q}$, and $d = \text{vec}\,(D)$, where $D \in \mathbf{R}^{q\times m}$, then

$$(A \otimes B)d = \text{vec}\,(BDA^T). \tag{7.5.28}$$

If A and B are square and non-singular the solution of the linear system $(A\otimes B)x = d$ can be written

$$x = (A^{-1} \otimes B^{-1})\text{vec}\,(D) = \text{vec}\,(B^{-1}DA^{-T}). \tag{7.5.29}$$

where D is the matrix which satisfies $d = \text{vec}\,(D)$.

Review Questions

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?

Problems

1. (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 (7.5.17).
   (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.
   (c) Prove the Woodbury formula (7.5.16).

2. Let $A \in \mathbf{R}^{m\times n}$ have rows a_i^T, i.e., $A^T = (a_1, \ldots, a_m)$. 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?

3. 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.

4. The methods of forwards- 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. Write a recursive LU Factorization algorithm based on the 2×2 block LU algorithm.

6. Let $B \in \mathbf{R}^{n\times n}$ be a matrix for which $\|B\| < 1$. Show that the infinite series and product

$$(I-B)^{-1} = \begin{cases} I + B + B^2 + B^3 + B^4 \cdots, \\ (I+B)(I+B^2)(I+B^4)(I+B^8)\cdots \end{cases}$$

both converge to the indicated limit.

Hint: Use the identity $(I + B + \cdots + B^k)(I - B) = I - B^{k+1}$.

(b) Show that the matrix $(I-B)$ is nonsingular and that $\|(I-B)^{-1}\| \leq 1/(1-\|B\|)$.

7. (a) Let $A \in \mathbf{R}^{m\times n}$, $B \in \mathbf{R}^{n\times p}$, with m and n even. Show that whereas conventional matrix multiplication requires mnp multiplications (M) and $m(n-1)p$ additions (A) to form the product $C = AB \in \mathbf{R}^{m\times p}$, Strassen's algorithm, using conventional matrix multiplication at the block level, requires

$$\frac{7}{8}mnp \text{ M } + \frac{7}{8}m(n-2)p + \frac{5}{4}n(m+p) + 2mp \text{ A}.$$

(b) Show, using the result in (a), that if we assume that "M $\approx$ A", Strassen's algorithm is cheaper than conventional multiplication when $mnp \leq 5(mn+np+mp)$.

7.6 Direct Methods for General Sparse Systems

7.6.1 Introduction

A matrix $A \in \mathbf{R}^{n\times n}$ is called **sparse** if only a small fraction of its elements are nonzero. Similarly, a linear systems $Ax = b$ is called sparse if its matrix A is sparse. The simplest class of sparse systems is the class of banded systems treated

in Sect. 7.4. 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 500–100,000. Typically, A will have only a few nonzero elements in each row, regardless of the value of n. In Fig. 6.5.1 we show a symmetric matrix of order 478 with 7551 nonzeros from a model of a chemical distillation column.

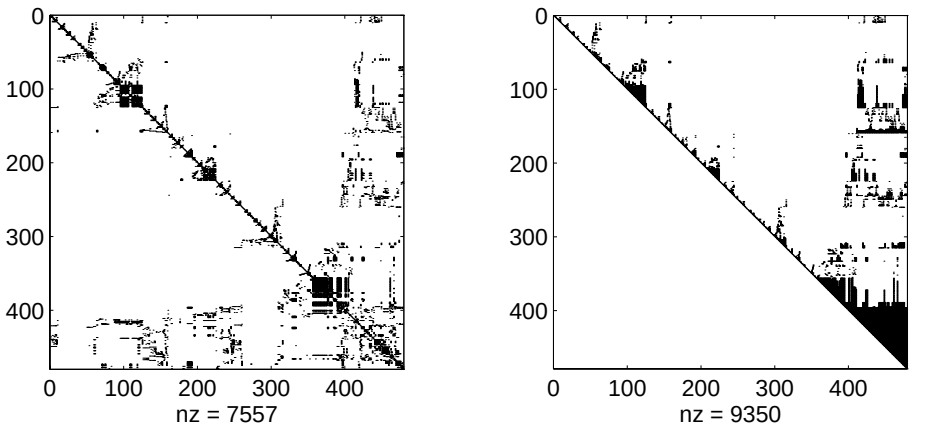


Figure 7.6.1. *Nonzero pattern of a matrix and its Cholesky factor.*

For some classes of 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, whereas iterative methods 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. The object is not only to reduce cost, but rather that without exploitation of sparsity, many large problems would be totally intractable.

7.6.2 Storage Schemes for Sparse Vectors and Matrices

The simplest class of sparse matrices is the class of band matrices (see Section 7.3), which have the property that in each row all nonzero elements are 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.

In some applications, one encounters matrices where the bandwidth differs

greatly from row to row. For this class of matrices, called **variable-band matrices**, it is convenient to use a storage scheme storing every element between the first nonzero element in a row and the diagonal and between the first nonzero element in a column and the diagonal. For such a matrix A we define

$$f_i = f_i(A) = \min\{j \mid a_{ij} \neq 0\}, \quad l_j = l_j(A) = \min\{i \mid a_{ij} \neq 0\}. \tag{7.6.1}$$

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 $f_i(A) = l_i(A)$, $i = 1, \ldots, n$.

Definition 7.6.1.

The **envelope** *(or profile) of A is the index set*

$$\text{Env}\,(A) = \{(i,j) \mid f_i \leq j \leq i; \text{ or } l_j \leq i < j;\}. \tag{7.6.2}$$

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 a storage scheme is used where all elements a_{ij} with $(i,j) \in \text{Env}\,(A)$ are 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 7.6.2.

Assume that the triangular factors L and U of A exist. Then it holds that

$$\text{Env}\,(L+U) = \text{Env}\,(A),$$

i.e., the nonzero elements in L and U are contained in the envelope of A.

Often sparse matrices occur with a pattern of nonzero elements which is much less regular, as was illustrated in Fig. 6.5.1. Other application areas may give pattern of quite different characteristics. We now consider storage schemes that are suitable when solving such general sparse linear systems.

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 first consider a storage scheme for a sparse vector x. We store the nonzero elements of x in **compressed form** in a vector xc with dimension nz, where nz 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 (nz, xc, ix), where

$$xc_k = x_{ix(k)}, \quad k = 1, \ldots, nz.$$

Example 7.6.1. The vector $x = (0, 4, 0, 0, 1, 0, 0, 0, 6, 0)$ can be stored in compressed form as

$$xc = (1, 4, 6), \qquad ix = (5, 2, 9), \qquad nz = 3$$

Operations on sparse vectors are simplified if *one* of the vectors is first **uncompressed**, i.e, stored as a dense vector. This 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 coordinate 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

$$\text{for } k = 1, \ldots, nnz, \quad y(ix(k)) := a * xc(k) + y(ix(k));$$

There are several ways to generalize this storage scheme to store a sparse matrix in compressed form. A simple scheme 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.

$$ac(k) = a_{i,j}, \quad i = ix(k), \quad j = jx(k), \quad k = 1, \ldots, nz.$$

Hence A is stored in "coordinate form" as an unordered set of triples consisting of a numerical value and two indices. For the initial representation of a general sparse matrix this scheme is very convenient, and further nonzero elements are easily added to the structure. It has the drawback that storage overhead is large since two extra integer vectors are needed. More important is that it is difficult to access the matrix A by rows or by columns, which is needed for the implementation of Gaussian elimination.

Another possibility is to store the matrix as a collection of sparse row vectors, where for each vector its nonzero elements are 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. Alternatively a similar scheme storing A as a collection of column vectors may be used.

Example 7.6.2. 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 as

$$AC = (a_{11}, a_{13} \mid a_{21}, a_{22}, a_{24} \mid a_{32}, a_{33}, a_{35} \mid a_{42}, a_{44} \mid a_{54}, a_{55})$$

$$jx = (1, 3, 1, 2, 4, 2, 3, 5, 2, 4, 4, 5)$$
$$ia = (1, 3, 6, 9, 11, 13)$$

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.

In the general sparse storage scheme only nonzero elements are stored. This saving is however bought at the cost of storage for the vector jx of column subscripts. This overhead storage can often be decreased by using a clever compressed scheme due to Sherman, see George and Liu [1981, pp. 139–142].

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.

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.

7.6.3 Orderings for Sparsity

One fact that makes Gaussian elimination for general sparse matrices difficult is that it is necessary to reorder rows and columns to avoid fill-in in the factors L and U. The total number of nonzeros in these factors may be far greater than the number of nonzeros in A.

Example 7.6.3. That the choice of pivot elements in Gaussian elimination may greatly affect storage and computation as is illustrated by the matrices

$$A = \begin{pmatrix} \times & \times & \times & \dots & \times \\ \times & \times & & & \\ \times & & \times & & \\ \vdots & & & \ddots & \\ \times & & & & \times \end{pmatrix}, \qquad PAP^T = \begin{pmatrix} \times & & & & \times \\ & \ddots & & & \vdots \\ & & \times & & \times \\ & & & \times & \times \\ \times & \dots & \times & \times & \times \end{pmatrix}$$

If the $(1,1)$ element is chosen as the first pivot in A, then the fill in will be total and $n^3/3$ operations are required. In PAP^T the orderings of rows and columns have been reversed. Here there is no fill-in except in the last step of Gaussian elimination if pivots are chosen in natural order, and only about $2n$ flops are required to perform the factorization.

Matrices, or block matrices of the same structure as PAP^T are sometimes called **arrowhead matrices**. They are important in many applications, e.g., structural or domain decomposition, see Golub and Van Loan [1989], Section 10.3.4.

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. Orderings which minimize fill-in usually also nearly minimize operation counts and vice versa. A common strategy is to choose P and Q to approximately minimize the bandwidth or envelope of PAQ. (Note that the reordered matrix PAP^T in Example 7.6.3 has a small envelope but A has a full envelope!) The number of possible orderings of rows and columns is very large, $n! \times n!$. To find the *optimal* ordering, i.e., one which minimizes the number of nonzero in L and U is unfortunately in general a much harder problem than solving the linear system!

For variable-band matrices no fill-in occurs in L and U outside the envelope. An important problem is therefore to find row and column permutations P and Q such that the matrix PAQ has a small envelope. There are heuristic algorithms for approximately minimizing the envelope, which usually produce acceptable orderings for practical purposes.

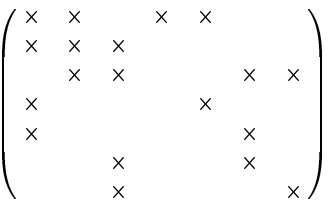
When the matrix A has a symmetric structure it is only necessary to consider symmetric permutations PAP^T. Note that in general the ordering for sparsity may not give pivotal elements which are acceptable from the point of numerical stability. For now we disregard this problem and will comment on it later. If A is symmetric positive definite then the Cholesky factorization $A = LL^T$ is numerically stable for *any* symmetric permutation PAP^T. This, as we shall see, leads to a great simplification in the algorithm.

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. To represent a structurally symmetric matrix A we will use the **undirected graph of** A.

Definition 7.6.3.

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* $1, 2, \ldots, n$ *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.*

Example 7.6.4. Below is an example of a matrix A and its ordered graph $G(A)$, where nonzero elements in A are denoted by $\times$.



A $G(A)$

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 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 $i = 1, \ldots, n-1$ 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 was suggested by 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 [9, Section 4.3.3]. In Fig. 6.5.2 we show the structure of the matrix from Fig. 6.5.1 and its Cholesky factor after reverse Cuthill–McKee reordering. The number of non-zero elements in the Cholesky factor is 23,866.

Surprisingly, the orderings that approximately minimize the total fill-in in Gaussian elimination tend *not* to give a small bandwidth. Typically, the factors L and U instead have their nonzeros scattered throughout their triangular parts. For some problems, such orderings can reduce fill-in by one or more orders of magnitude over the corresponding minimum bandwidth ordering.

By far the most used heuristic algorithm for finding an ordering with low fill-in is the **Markowitz algorithm**. Let $A^{(k)}$ be the submatrix that has yet to be factorized at the beginning of the kth step of Gaussian elimination. In the Markowitz algorithm one performs a row and column interchange so that the product

$$(r_i - 1)(c_j - 1),$$

is minimized, where r_i is the number of nonzero elements in the ith row and c_j is the number of nonzero elements in the jth column of $A^{(k)}$. (Some rules for tie-breaking are also needed.) This is equivalent to a *local minimization* of the fill in at each stage.

Figure 7.6.2. *Matrix and its Cholesky factor after reverse Cuthill–McKee reordering.*

Figure 7.6.3. *Matrix and its Cholesky factor after minimum-degree reordering.*

In the symmetric case we consider only symmetric orderings, and we have $r_i = c_i$. The Markowitz ordering is then equivalent to minimizing r_i, and the resulting algorithm is called the **minimum-degree algorithm**. Remarkably fast *symbolic* implementations of the minimum-degree algorithm exist, which use a graph model of the Cholesky factorization. In Fig. 6.5.3 shows the structure of the matrix from Fig. 6.5.1 and its Cholesky factor after minimum-degree reordering. The number of non-zero elements in the Cholesky factor is here only 12064. For nested dissection orderings, see George and Liu [9, Chapter 8].

7.6.4 Symbolic and Numerical Factorization of Sparse Matrices

We noticed earlier that for symmetric positive definite matrices the ordering problem is greatly simplified since any choice of pivots along the diagonal is numerically stable. This means that the permutation P can be chosen with regard only to sparsity, and without any numerical checks for stability. It is therefore possible to determine the minimum degree ordering using an graph model of the Cholesky factorization. At the same time the nonzero structure of the Cholesky factor L can be determined and a storage structure for L generated. We remark that in predicting the structure of L from that of $B = P^T AP$ by performing the Cholesky factor symbolically, $L + L^T$ will be at least as full as PAP^T. For details of the implementation of the minimum-degree algorithm and the symbolic factorizations we refer to George and Liu [9, *Chapter* 5].

Hence the structure of an algorithm for the Cholesky factorization of a sparse symmetric positive definite matrix A is as follows:

Algorithm 7.6.1 Sparse Cholesky Factorization

1. Determine a permutation P such that PAP^T has a sparse Cholesky factor L.
2. Perform the Cholesky factorization of PAP^T symbolically to generate a storage structure for L.
3. Compute PAP^T numerically and store in the data structure of L.
4. Compute the Cholesky factor L numerically and solve $Lz = c$, $L^T y = z$, giving the solution $x = P^T y$.

We stress again that steps 1 and 2 are done symbolically, only working on the structure of A. In steps 3 and 4 a static storage scheme can be used, since we have predicted where all the fill in will occur. For unsymmetric systems such a scheme is not in general possible, since some kind of stability check on the pivot elements must be performed, and storage cannot be predicted from the structure of A only. Hence for unsymmetric problems the storage structure must be determined dynamically during the numerical elimination phase.

Usually a **threshold pivoting** scheme is used in the nonsymmetric case. This means that the chosen pivot is restricted by the inequality

$$|a_{ij}^{(k)}| \geq \tau \max_r |a_{rj}^{(k)}| \tag{7.6.3}$$

where τ is a predetermined threshold value, $0 < \tau \leq 1$. The choice of τ is a delicate balance between sparsity and stability, and value $\tau = 0.1$ is usually recommended as a good compromise. (Note that the usual partial pivoting is obtained for $\tau = 1$.) The condition (7.6.3) 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 Section 7.7.4.

For symmetric positive definite sparse systems we want to preserve symmetry and so only permutations PAP^T are considered. Here things are much simplified by the fact that no pivoting is needed for numerical stability.

Review Questions

1. Give an extreme example of fill in.
2. What does the block LU factorization of an arrow matrix look like?
3. Describe threshold pivoting.
4. Describe the Markowitz algorithm and Cuthill–McKee ordering.

Problems

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 $C = \sum_{i=1}^{n} a_{\cdot i} b_i^T$.
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:

$$\begin{aligned} &\text{for } k = 1, \ldots, nx \\ &\quad y(\text{index}(k)) = a * x(k) + y(\text{index}(k)); \end{aligned}$$

   (b) Give an efficient algorithm for computing the inner product of two compressed vectors.
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.

7.7 Perturbation Theory and Error Analysis

7.7.1 Introduction

In practice the matrix A and right hand side b in a linear system $Ax = b$ 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 could in general not be represented exactly as floating-point numbers in the computer.) It is therefore of interest to know how uncertainties in the data A and b influence the solution x. This question is studied in the next two subsections. The perturbation analysis will make heavy use of the matrix and vector norms introduced in Section 1.6.6.

Rounding errors, which will further perturb the solution, are introduced when a linear system is solved by a Gaussian elimination in floating point arithmetic. In Section 7.7.6 we analyze rounding errors for Gaussian elimination. It is shown that these are equivalent to small perturbations of the data A and b, and hence their influence on the computed solution can be bounded using the perturbation results from Sections 7.7.2–7.7.3.

7.7.2 Perturbation Analysis

Let x be the solution x to a system of linear equations $Ax = b$, where A is nonsingular and $b \neq 0$. We shall investigate the sensitivity of x to perturbations δA and δb in A and b. The perturbation δx satisfies

$$(A + \delta A)(x + \delta x) = b + \delta b.$$

Subtracting out $Ax = b$ we get $(A+\delta A)\delta x = (-\delta A x + \delta b)$. If also the matrix $A+\delta A$ is nonsingular, we can multiply by A^{-1} and solve for δx which yields

$$\delta x = (I + A^{-1}\delta A)^{-1} A^{-1}(-\delta A x + \delta b). \tag{7.7.1}$$

This is the basic identity for the perturbation analysis.

In the simple case that $\delta A = 0$ we have $\delta x = A^{-1}\delta b$ and taking norms we obtain

$$\|\delta x\| \leq \|A^{-1}\| \, \|\delta b\|.$$

Usually it is more appropriate to consider *relative* perturbations,

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A, x) \frac{\|\delta b\|}{\|b\|}, \quad \kappa(A, x) := \frac{\|Ax\|}{\|x\|} \|A^{-1}\|$$

where $\kappa(A, x)$ is the condition number. This inequality is sharp in the sense that for any matrix norm and for any A and b there exists a perturbation δb such that equality holds. Using $\|b\| = \|Ax\| \leq \|A\| \, \|x\|$ we obtain

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \, \|A^{-1}\| \frac{\|\delta b\|}{\|b\|}. \tag{7.7.2}$$

Note that here equality will hold only for rather *special right hand sides* b. For given x (or b) the bound (7.7.2) may be unachievable for any δb. Equation (7.7.2) shows that a relative perturbation in the right hand side can at most be amplified by the factor $\|A\|\,\|A^{-1}\|$.

Definition 7.7.1.

The **condition number** *for a nonsingular matrix A is*

$$\kappa = \kappa(A) = \|A\|\,\|A^{-1}\|, \tag{7.7.3}$$

where $\|\cdot\|$ denotes any matrix norm.

As we will show below $\kappa(A)$ is the condition number with respect to inversion of A, and also measures the sensitivity of the solution x to perturbations in A. Clearly the condition number $\kappa(A)$ 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. 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 $\kappa(AB) \le \kappa(A)\kappa(B)$. Further, for all l_p-norms it follows from the identity $AA^{-1} = I$ that

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

that is, the condition number is always greater or equal to one.

Matrices with small condition numbers are said to be **well-conditioned**. For any real, orthogonal matrix Q we have $\kappa_2(Q) = \|Q\|_2\|Q^{-1}\|_2 = 1$, so Q is perfectly conditioned in the l_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. Using the singular value decomposition (see Section 8.3) it follows that

$$\kappa_2(A) = \sigma_1/\sigma_n,$$

where σ_1 and σ_n are the largest and smallest singular values of A.

In order to make a strict analysis of the case when the perturbation $\delta A \neq 0$, we need the following result:

Lemma 7.7.2.

Let $E \in \mathbf{R}^{n\times n}$ be a matrix for which $\|E\| < 1$. Then the matrix $(I - E)$ is nonsingular and for its inverse we have the estimate

$$\|(I - E)^{-1}\| \le 1/(1 - \|E\|). \tag{7.7.4}$$

Proof. If $(I-E)$ is singular there exists a vector $x \neq 0$ such that $(I-E)x = 0$. Then $x = Ex$ and $\|x\| = \|Ex\| \le \|E\|\,\|x\| < \|x\|$, which is a contradiction since $\|x\| \neq 0$. Hence $(I - E)$ is nonsingular. Now consider the identity $(I - E)(I - E)^{-1} = I$ or $(I-E)^{-1} = I+E(I-E)^{-1}$. Taking norms we get $\|(I-E)^{-1}\| \le 1+\|E\|\,\|(I-E)^{-1}\|$, and (7.7.4) follows. (For another proof see hint given in Problem 6.2.19.) ∎

Corollary 7.7.3.
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 7.7.2. From the identity $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$ we have $\|A^{-1} - B^{-1}\| \leq \|A^{-1}\|\,\|B - A\|\,\|B^{-1}\|$. The second inequality now follows from the first. □

We are now ready to prove the main result of this section.

Theorem 7.7.4.
Consider the linear system $Ax = b$, *where the matrix* $A \in \mathbf{R}^{n\times n}$ *is nonsingular. Let* $(A + \delta A)(x + \delta x) = b + \delta b$, *be a perturbed system, and assume that*

$$\eta = \|A^{-1}\|\,\|\delta A\| = \kappa(A)\|\delta A\|/\|A\| < 1.$$

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

$$\|\delta x\| \leq \frac{\kappa(A)}{1-\eta}\left(\frac{\|\delta A\|}{\|A\|}\|x\| + \frac{\|\delta b\|}{\|A\|}\right). \tag{7.7.5}$$

Proof. We have $\|A^{-1}\delta A\| \leq \|A^{-1}\|\,\|\delta A\| = \eta < 1$. Hence by (7.7.4) $(A + \delta A) = A(I + A^{-1}\delta A)$ is nonsingular and

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

Taking norms in equation (7.7.1)

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

The theorem now follows by noting that $\|A^{-1}\| = \kappa(A)/\|A\|$. □

By substituting $b = I$, $\delta b = 0$ and $x = A^{-1}$ and proceeding similarly from $(A + \delta A)(X + \delta X) = I$ we obtain the perturbation bound for $X = A^{-1}$

$$\frac{\|\delta X\|}{\|X\|} \leq \frac{\kappa(A)}{1-\eta}\frac{\|\delta A\|}{\|A\|}. \tag{7.7.6}$$

This shows that $\kappa(A)$ is indeed the condition number of A with respect to inversion.

Theorem 7.7.4 can be used to derive upper bounds for $\|\delta x\|$, when bounds for $\|\delta A\|$ and $\|\delta b\|$ are known. For example, if

$$\|\delta A\| \leq \epsilon_A\|A\|, \qquad \|\delta b\| \leq \epsilon_b\|b\|, \tag{7.7.7}$$

then it holds that

$$\|\delta x\| \leq \frac{\kappa(A)}{1-\eta}\left(\epsilon_A\|x\| + \epsilon_b\frac{\|b\|}{\|A\|}\right).$$

This analysis 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 7.7.1. 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 right hand side is subject to a perturbation δb such that $|\delta b| \leq (1, 10^{-4})^T$. Using the ∞-norm we have $\|\delta b\|_\infty = 1$, $\|A^{-1}\|_\infty = 1$ (neglecting terms of order 10^{-8}). Theorem 7.7.4 then gives the gross overestimate $\|\delta x\|_\infty \leq 1$.

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 right hand is now $|\delta\hat{b}| \leq 10^{-4}(1,1)^T$, and from $\|\delta\hat{b}\|_\infty = 10^{-4}$, $\|(\hat{A})^{-1}\|_\infty = 1$, we get the sharp estimate $\|\delta x\|_\infty \leq 10^{-4}$. The original matrix A is only *artificially ill-conditioned.*

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.

Theorem 7.7.5.

The reciprocal of the condition number $\kappa(A)$ of a nonsingular matrix A equals the distance of A to the set of singular matrices, i.e.,

$$\frac{1}{\kappa(A)} = \inf_{\delta A} \|\delta A\|/\|A\|, \quad (A + \delta A) \text{ singular}.$$

Proof. If $(A+\delta A)$ is singular, then there is a vector $x \neq 0$ such that $(A+\delta A)x = 0$, and

$$\|\delta A\| \geq \frac{\|\delta A x\|}{\|x\|} = \frac{\|Ax\|}{\|x\|} = \frac{\|Ax\|}{\|A^{-1}Ax\|} \geq \frac{1}{\|A^{-1}\|} = \frac{\|A\|}{\kappa(A)}.$$

It follows that

$$\kappa(A) \geq \|A\|/\|\delta A\|. \tag{7.7.8}$$

It can be shown that there exists a matrix δA such that the equality sign holds. For the l_2 norm this proof is left as Problem 4. ∎

The result in (7.7.8) can be used to get a *lower bound* for the condition number $\kappa(A)$, see for example, Problem 2. The theorem also holds for other subordinate matrix norms.

7.7.3 Component-wise perturbation analysis

Often the data is subject to perturbations which are bounded component-wise,

$$|\delta a_{ij}| \le \omega e_{ij}, \quad |\delta b_i| \le \omega f_i. \quad i,j = 1,\dots,n, \tag{7.7.9}$$

where $e_{ij} > 0$ and $f_i > 0$ are known. In order to write such component-wise bounds in a simple way we define the absolute value of a matrix A and vector b by

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

We also introduce the partial ordering "$\le$" for matrices A, B and vectors x, y, which is to be interpreted component-wise[7]

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

Further, 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 holds for matrix-vector multiplication.

With these notations we can write the bounds in (7.7.9) as

$$|\delta A| \le \omega E, \quad |\delta b| \le \omega f. \tag{7.7.10}$$

We now want to derive estimates for the corresponding perturbations in x. Taking absolute values in (7.7.1) we obtain the component-wise error bound

$$|\delta x| \le |(I + A^{-1}\delta A)^{-1}|\,|A^{-1}|(|\delta A||x| + |\delta b|)$$

Using the identity $(I+F)(I+F)^{-1} = I$ or $(I+F)^{-1} = I - F(I+F)^{-1}$ it follows easily that $|(I+F)^{-1}| \le (I - |F|)^{-1}$. (Compare Lemma 7.7.2.) Taking $F = A^{-1}\delta A$ we get the inequality

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

The matrix $(I - |A^{-1}||\delta A|)$ is guaranteed to be nonsingular if $\| \,|A^{-1}|\,|\delta A|\,\| < 1$. If the perturbations satisfy (7.7.10) we get

$$|\delta x| \le \omega(I - \omega|A^{-1}|E)^{-1}|A^{-1}|(E|x| + f), \tag{7.7.11}$$

provided that $\omega\kappa_E(A) < 1$, where $\kappa_E(A) = \| \,|A^{-1}|E\|$. Taking norms in (7.7.11) and using Lemma 7.7.2 we obtain

$$\|\delta x\| \le \frac{\omega}{1 - \omega\kappa_E(A)} \| \,|A^{-1}|(|E\,|x| + f)\|.$$

[7] Note that $A \le B$ in other contexts means that $B - A$ is positive semidefinite.

If we put $E = |A|$ and $f = |b|$ in (7.7.10) this corresponds to **component-wise relative error bounds** for A and b,

$$|\delta A| \leq \omega |A|, \qquad |\delta b| \leq \omega |b|. \tag{7.7.12}$$

For this special case we have

$$\|\delta x\| \leq \frac{\omega}{1 - \omega \kappa_{|A|}(A)} \| \, |A^{-1}| (|A| \, |x| + |b|) \, \|, \tag{7.7.13}$$

where

$$\kappa_{|A|}(A) = \| \, |A^{-1}| |A| \, \|,$$

(or cond (A)) is the **Bauer–Skeel condition number** of the matrix A. Note that since $|b| \leq |A| \, |x|$, it follows that

$$\|\delta x\| \leq 2\omega \| \, |A^{-1}| |A| \, |x| \, \| + O(\omega^2) \leq 2\omega \kappa_{|A|}(A) \|x\| + O(\omega^2).$$

If $\hat{A} = DA$, $\hat{b} = Db$ where $D > 0$ is a diagonal scaling matrix, then $|\hat{A}^{-1}| = |A^{-1}||D^{-1}|$. Since the perturbations scale similarly, $\delta\hat{A} = D\delta A$, $\delta\hat{b} = D\delta b$, it follows that

$$|\hat{A}^{-1}||\delta\hat{A}| = |A^{-1}||\delta A|, \qquad |\hat{A}^{-1}||\delta\hat{b}| = |A^{-1}||\delta b|.$$

Thus the bound in (7.7.13) and also $\kappa_{|A|}(A)$ are *invariant under row scalings.*

For the l_1-norm and l_∞-norm it holds that

$$\kappa_{|A|}(A) = \| \, |A^{-1}| |A| \, \| \leq \| \, |A^{-1}| \, \| \, \| \, |A| \, \| = \|A^{-1}\| \, \|A\| = \kappa(A),$$

i.e., the solution of $Ax = b$ is no more badly conditioned with respect to the 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 $\kappa(A)$.

Example 7.7.2. Consider the linear systems in Example 7.7.1. 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 cond (A) = cond $(\hat{A}) = 1 + 2 \cdot 10^{-4}$ in the ∞-norm.

7.7.4 Backward Error Bounds

Usually the data A and b of an equation system $Ax = b$ are subject to errors and not exact. Hence it is reasonable to regard a computed solution $\bar{x}$ as correct if it is the exact solution to a neighboring system

$$(A + \delta A)\bar{x} = b + \delta b, \tag{7.7.14}$$

where the backward errors δA and δb are small in comparison to the uncertainties in A and b. In Section 7.7.5 we will derive **a priori** bounds for $\|\delta A\|_\infty$ and $\|\delta b\|_\infty$,

so that (7.7.14) holds for the solution computed by Gaussian elimination. Such bounds, however, are usually gross overestimates. We now derive simple **a posteriori** bounds for the backward error of a computed solution $\bar{x}$. These bounds are usually much sharper, and hold regardless of the method used to compute $\bar{x}$.

Given $\bar{x}$, there are an infinite number of perturbations for which (7.7.14) is true. Clearly δA and δb must satisfy

$$\delta A\bar{x} - \delta b = b - A\bar{x} = r,$$

where r is the residual vector corresponding to $\bar{x}$. An obvious solution is $\delta A = 0$, and $\delta b = -r$. If we instead take $\delta b = 0$ then $\delta A\bar{x} = r$, and δA must satisfy $\|\delta A\|_2 \geq \|r\|_2/\|\bar{x}\|_2$. Since $(rp^T)\bar{x} = r(p^T\bar{x}) = r$ if $p^T\bar{x} = 1$ a particular perturbation is given by

$$\delta A = r\bar{x}^T/\|\bar{x}\|_2^2. \tag{7.7.15}$$

We have

$$\|r\bar{x}^T\|_2 = \sup_{\|y\|_2=1} \|r\bar{x}^T y\|_2 = \|r\|_2 \sup_{\|y\|_2=1} |\bar{x}^T y| = \|r\|_2\|\bar{x}\|_2,$$

and it follows that $\|\delta A\|_2 = \|r\|_2/\|\bar{x}\|_2$. Hence this perturbation must be of minimum l_2-norm, and gives a computable a posteriori backward error δA of $\bar{x}$. 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 7.7.6. (Oettli and Prager [1964]).

Let $r = b - A\bar{x}$, E and f be nonnegative and set

$$\omega = \max_i \frac{|r_i|}{(E|\bar{x}| + f)_i}, \tag{7.7.16}$$

where $0/0$ is interpreted as 0. If $\omega \neq \infty$, there is a perturbation δA and δb with

$$|\delta A| \leq \omega E, \qquad |\delta b| \leq \omega f, \tag{7.7.17}$$

such that

$$(A + \delta A)\bar{x} = b + \delta b. \tag{7.7.18}$$

Moreover, ω is the smallest number for which such a perturbation exists.

Proof. From (7.7.16) we have

$$|r_i| \leq \omega(E|\bar{x}| + f)_i,$$

which implies that $r = D(E|\bar{x}| + f)$, where $|D| \leq \omega I$. It is then readily verified that

$$\delta A = DE \operatorname{diag}\big(\operatorname{sign}(\bar{x}_1), \ldots, \operatorname{sign}(\bar{x}_n)\big), \qquad \delta b = -Df$$

are the required backward perturbations.

Further, given perturbations δA and δb satisfying equations (7.7.17)–(7.7.18) 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 (7.7.16) is optimal. □

In particular we can take $E = |A|$, and $f = |b|$ in Theorem 7.7.6, to get an expression for the component-wise relative backward error ω of a computed solution. This can then be used in (7.7.12) or (7.7.13) to compute a bound for $\|\delta x\|$.

7.7.5 Estimating Condition Numbers

It is important to note 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 the size of A^{-1} or the condition number of A.

Example 7.7.3. 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.[8]

To estimate the error in a computed solution we could compute the backward error using, e.g., (7.7.15), and then apply Theorem 7.7.4. Then we need to know $\kappa(A) = \|A\|\,\|A^{-1}\|$. To compute $\|A\|$ poses no problems, but computing A^{-1} costs about three times as much as computing x (see Section 7.2.9). In practice, therefore, a cheap **estimate** of the $\kappa(A)$ is computed instead.

The following widely used algorithm is due to Cline et al. [3, 1979]. and requires only about $2n^2$ flops if an LU factorization of A is known. It is based on the fact that if $Ax = w$ then $x = A^{-1}w$, then a *lower bound* for $\|A^{-1}\|$ is given by

$$\|A^{-1}\| \geq \|x\|/\|w\|. \tag{7.7.19}$$

[8] As remarked by a prominent expert in error-anaylysis "Anyone unlucky enough to encounter this sort of calamity has probably already been run over by a truck"!

Since the computation of x only requires the solution of the two triangular systems $Ly = w$, $Ux = y$, computing the bound (7.7.19) only requires n^2 flops.

To make this into a reliable estimate several modifications are necessary. The first is to start with a vector w computed from $A^T w = u$, which requires the solution of the two triangular systems $U^T v = u$, $L^T w = v$. The second modification is to choose the components of the right hand side vector u of the system $U^T v = u$ so that the growth of v is enhanced. One way to do so, although not quite foolproof, is to take $u_i = \pm 1$, $i = 1, \ldots, n$, where the sign is chosen to maximize $|v_i|$. Note that any ill-conditioning in A is likely to be reflected in U, whereas L, being unit upper triangular, tends to be well-conditioned. The final estimate is taken to be

$$1/\kappa(A) \leq \|w\|/(\|A\|\|x\|). \tag{7.7.20}$$

(It is preferable to estimate the quantity $1/\kappa(A)$ since then a singular matrix is signaled by zero rather than by ∞ and overflow is avoided.) We stress that (7.7.20) always gives an *underestimate* of $\kappa(A)$.

This condition estimate is used in LINPACK with the l_1-norm. This norm is chosen because the vector norm is cheap to compute and the corresponding subordinate matrix norm $\|A\|_1 = \max_j \|a_j\|_1$ can be computed from the columns a_j of A. In practice it has been found that this estimate seldom is off by a factor more than 10. The actual implementation of this condition estimator is not trivial; care must be taken to avoid overflows and underflows.

We point out here that since $Ax = w$, $A^T w = u$, we have $x = (A^T A)^{-1} u$. Hence we have carried out one step of the inverse power method on $A^T A$ using the special starting vector u. As shown in Section 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 $A^T A$.

The technique described above cannot be used to estimate the Bauer–Skeel condition number $\mathrm{cond}\,(A) = \| |A^{-1}||A| \|$, which appears in the component-wise relative perturbation analysis in Sec, 7.7.3. A more general estimator has been devised by Hager [1984], which estimates $\|B\|_1$, given that Bx and $B^T x$ can be obtained for an arbitrary vector x. Note that it is not necessary to know B explicitly. In particular, to estimate $\kappa_1(A)$ we take $B = A^{-1}$, and we are required to compute $A^{-1}x$ and $A^{-T}x$. This is equivalent to solving linear systems with coefficient matrices A and A^T, which is cheap if the LU factorization is known. Less obvious, Hager's estimator can also be used to estimate the bound in (7.7.13). The problem is then to estimate an expression of the form $\| |A^{-1}|g\|_\infty$, where $g > 0$. However, this can be reduced to estimating $\|B\|_1$ where $B = (A^{-1}G)^T$, $G = \mathrm{diag}\,(g_1, \ldots, g_n)$, using the equalities

$$\| |A^{-1}|g\|_\infty = \| |A^{-1}|Ge\|_\infty = \| |A^{-1}G|e\|_\infty = \| |A^{-1}G|\|_\infty = \|A^{-1}G\|_\infty.$$

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 (7.7.16) gives an approximate bound for the error in a computed solution $\bar{x}$. Hager's condition estimator is used in LAPACK and MATLAB.

7.7.6 Rounding Error Analysis for 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 computing around 1946 many mathematicians were pessimistic about the numerical stability of Gaussian elimination. It was widely thought that it would be impractical to solve even systems of fairly moderate order. Fortunately this is not so. Gaussian elimination with partial pivoting is in practice a remarkably stable method. A major breakthrough in the understanding of Gaussian elimination came with the famous backward rounding error analysis of Wilkinson [19, 1961].

Using the standard model for floating point computation Wilkinson showed that the *computed matrices* $\bar{L}$ and $\bar{U}$ *are the exact triangular factors of the perturbed matrix* $A + E$,

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

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

$$|e_{ij}| \leq 3u \min(i-1, j) \max_k |\bar{a}_{ij}^{(k)}|. \tag{7.7.21}$$

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 elements* $\bar{a}_{ij}^{(k)}$*, and that the size of the multipliers is of no consequence* (see the remark on possible large multipliers for positive-definite matrices, Section 7.4.1).

If we introduce the **growth ratio**

$$g_n = \max_{i,j,k} |\bar{a}_{ij}^{(k)}| / \max_{ij} |a_{ij}|, \tag{7.7.22}$$

and the matrix $F = (f_{ij})$, $f_{i,j} = \min\{i-1, j\}$, then $E = (e_{ij})$ can be bounded component-wise by $|E| \leq 3g_n u \max_{ij} |a_{ij}| F$. We have $\|F\|_\infty \leq \frac{1}{2}n(n+1) - 1$, and by slightly refining this estimate and using the estimate $\max_{ij} |a_{ij}| \leq \|A\|_\infty$, one obtains the result:

Theorem 7.7.7.

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*

$$\bar{L}\bar{U} = A + E, \quad \|E\|_\infty \leq 1.5n^2 g_n u \|A\|_\infty. \tag{7.7.23}$$

The bound obtained above is satisfactory unless the ratio g_n is large. The quantities $\max_k |\bar{a}_{ij}^{(k)}|$ are not known before the elimination. In order to obtain an *a priori bound* on g_n we assume that the computed multipliers satisfy $|\bar{m}_{ik}| \leq 1$, which is true if partial or complete pivoting is employed. It can then be shown that an estimate similar to (7.7.23) holds with the constant 1 instead of 1.5. Furthermore we have the inequality

$$|a_{ij}^{(k+1)}| = |a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}| \leq |a_{ij}^{(k)}| + |a_{kj}^{(k)}| \leq 2 \max_k |\bar{a}_{ij}^{(k)}|.$$

For partial pivoting we can guarantee only that $g_n \leq 2^{n-1}$, and this upper bound is attained for matrices $A_n \in \mathbf{R}^{n\times n}$ of the form

$$A_4 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{pmatrix}.$$

Already for $n = 41$ we can have $g_n = 2^{40} \approx 10^{12}$. Hence the worst-case behavior of partial pivoting is very unstable. Nevertheless, decades of experience and extensive experiments have shown that in practice Gaussian elimination with partial pivoting is highly stable. Substantial growth seems to occur only for a tiny proportion of matrices arising naturally.

For complete pivoting, Wilkinson [19, 1961] has proved that

$$g_n \leq (n \cdot 2^1 3^{1/2} 4^{1/3} \cdots n^{1/(n-1)})^{1/2} < 1.8\sqrt{n} n^{\frac{1}{4}\log n},$$

and that this bound is not attainable. This bound is *much* smaller than that for partial pivoting, for example $g_{50} < 530$. It was long conjectured that $g_n \leq n$ for real matrices and complete pivoting. Recently, after long investigations this was disproved and a matrix of order 13 found for which $g_n = 13.025$.

We recall, see Sec, 7.4, that for special classes of matrices, the growth factor g_n may be bounded independently of n. For example, if A is row or column diagonally dominant then $g_n \leq 2$. For tridiagonal matrices and partial pivoting it can be shown that $g_n \leq 2$.

If A is symmetric positive definite then $g_n \leq 1$, with no pivoting. With the additional assumption that the computed square root satisfies $fl(\sqrt{x}) = \sqrt{x}(1+\delta)$, $|\delta| \leq u$, Wilkinson has proved the following result.

Theorem 7.7.8.

Let $A \in R^{n\times n}$ be a symmetric positive definite matrix. Provided that $2n^{3/2}u\kappa(A) < 0.1$ the Cholesky factor of A can be computed without breakdown. The computed $\bar{L}$ satisfies

$$\bar{L}\bar{L}^T = A + E, \quad \|E\|_2 < 2.5n^{3/2}u\|A\|_2, \tag{7.7.24}$$

and hence is the exact Cholesky factor of a matrix close to A.

The results given above are essentially the best normwise bounds that can be obtained. Below we give a slightly different component-wise analysis. We first consider roundoff in an expression of the form

$$s = \Big(-\sum_{i=1}^{k-1} a_i b_i + c\Big)/d. \tag{7.7.25}$$

Note that we obtain a slightly sharper result we assume that the term c is added *last* in the formula above. A simple extension of the results in Sec, 2.4.1 shows that

the computed $\bar{s}$ satisfies

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

where

$$|\delta_k| \le 2\tilde{u}, \qquad |\delta_i| \le (i+1-k)\tilde{u}, \quad i = 1, \ldots, k-1,$$

and $\tilde{u} = 1.01u$. It follows that

$$|\bar{s}d + \sum_{i=1}^{k-1} a_i b_i - c| \le k\tilde{u}\Big(|\bar{s}d| + \sum_{i=1}^{k-1} |a_i||b_i|\Big), \tag{7.7.26}$$

and here the inequality holds independent of the summation order.

We have given several versions of LU factorization. They will all lead to the same error bounds, since each does the same operations with the same arguments. We base here the analysis on Doolittle's algorithm, in which we compute (see Sec, 7.2.8)

$$u_{ij} = a_{ij} - \sum_{p=1}^{i-1} l_{ip}u_{pj}, \quad j \ge i; \qquad l_{ij} = \Big(a_{ij} - \sum_{p=1}^{j-1} l_{ip}u_{pj}\Big)/u_{jj}, \quad i > j.$$

Using the above result we immediately obtain

$$\Big|a_{ij} - \sum_{p=1}^{i} \bar{l}_{ip}\bar{u}_{pj}\Big| \le i\tilde{u} \sum_{p=1}^{i} |\bar{l}_{ip}|\,|\bar{u}_{pj}| \quad j \ge i;$$

$$\Big|a_{ij} - \sum_{p=1}^{j} \bar{l}_{ip}\bar{u}_{pj}\Big| \le j\tilde{u} \sum_{p=1}^{i} |\bar{l}_{ip}|\,|\bar{u}_{pj}|, \quad i > j,$$

where we have defined $\bar{l}_{ii} = l_{ii} = 1$. These inequalities may be written in matrix form

$$A + E = \bar{L}\bar{U}, \qquad |E| \le n\tilde{u}|\bar{L}|\,|\bar{U}|. \tag{7.7.27}$$

To estimate the error in a computed solution $\bar{x}$ of a linear system of equations $Ax = b$ we must also take into account the rounding errors performed in the solution of the two triangular systems

$$\bar{L}y = b, \qquad \bar{U}x = y.$$

It has been observed that large errors almost never occur in the solution of the triangular systems. The lower triangular system $Ly = b$ is solved by forward substitution. If we let $\bar{y}$ denote the computed solution, then

$$\bar{y}_i = fl\big((-\sum_{k=1}^{i-1} \bar{l}_{ik}\bar{y}_k + b_i)/\bar{l}_{ii}\big), \quad i = 1, \ldots, n.$$

Using the result above, it follows that the computed $\bar{y}$ satisfies $(\bar{L}+\delta\bar{L})\bar{y} = b$, where

$$|\delta\bar{l}_{ik}| \leq \cdot 2\tilde{u}|\bar{l}_{ii}|, \quad i = j, \qquad |\delta\bar{l}_{ik}| \leq (i+1-k)\tilde{u}|\bar{l}_{ik}|, \quad i > j.$$

A similar analysis can be made for the error in solving $\bar{U}x = \bar{y}$. Hence the computed $\bar{x}$ satisfies $(\bar{U} + \delta\bar{U})\doteq\bar{y}$, where

$$|\delta\bar{L}| \leq n\tilde{u}|\bar{L}|, \qquad |\delta\bar{U}| \leq n\tilde{u}|\bar{U}|. \tag{7.7.28}$$

Hence for triangular systems the component-wise relative backward error is always small. 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 (7.7.27)–(7.7.28) gives the following backward error result for the computed solution $\bar{x}$

Theorem 7.7.9.

Let $\bar{x}$ denote the computed solution of the system $Ax = b$, and assume that the triangular factors $\bar{L}$ and $\bar{U}$ have been computed by Gaussian elimination. Then $\bar{x}$ satisfies exactly

$$(A + \delta A)\bar{x} = b,$$

where δA is a matrix depending on both A and b, such that

$$|\delta A| \leq n\tilde{u}(3 + n\tilde{u})|\bar{L}|\,|\bar{U}|. \tag{7.7.29}$$

Note that although the perturbation δA depends upon b the *bound* on $|\delta A|$ is independent on b.

Since the elements in $\bar{L}$ and $\bar{U}$ satisfy $|\bar{l}_{ij}| \leq 1$, $|\bar{u}_{ij}| \leq g_n\|A\|_\infty$, it follows from (7.7.29) that

$$\|\delta A\|_\infty \leq n^3 g_n \tilde{u}(1 + n\tilde{u}/2)\|A\|_\infty. \tag{7.7.30}$$

The residual for the computed solution satisfies $\bar{r} = b - A\bar{x} = \delta A\bar{x}$, and using (7.7.30) it follows that

$$\|\bar{r}\|_\infty \leq n^3 g_n \tilde{u}(1 + n\tilde{u}/2)\|A\|_\infty\|\bar{x}\|_\infty.$$

This shows the very important fact that *Gaussian elimination will give a small relative residual even for ill-conditioned systems.* The quantity $\|b-A\bar{x}\|_\infty/(\|A\|_\infty\|\bar{x}\|_\infty)$ will in practice be of the order nu.

7.7.7 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, $i = 1, \ldots, n$, 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 7.7.10.

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 7.2.3

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

□

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 7.7.4. The system $Ax = b$ in Example 7.7.1 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. An discussed in

Sec, 7.7.5, an estimate of $\kappa(A)$ is often used to access the accuracy of the computed solution. If, e.g., the perturbation bound (7.7.2) is applied to the scaled system $(D_r A D_c)x' = D_r b$

$$\frac{\|D_c^{-1}\delta x\|}{\|D_c^{-1}x\|} \leq \kappa(D_r A D_c)\frac{\|D_r \delta b\|}{\|D_r b\|}. \tag{7.7.31}$$

Hence if $\kappa(D_r A D_c)$ can be made smaller than $\kappa(A)$, then it seems that we might expect a correspondingly more accurate solution. Note however that in (7.7.31) 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 $D_r = \text{diag}\,(d_i)$ so that each row in $D_r A$ has the same l_1-norm, i.e.,

$$d_i = 1/\|a_i^T\|_1, \quad i = 1, \ldots, n. \tag{7.7.32}$$

(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 7.7.1. 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 7.7.5. 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 [15, 1979]. He shows a pivoting rule in Gaussian elimination should depend not only on the coefficient matrix but also on the solution. Hence separating the matrix factorization from the solution of the linear system may lead to instability. His scaling rule is based on minimizing a bound on the backward error that contains the quantity

$$\frac{\max_i(|D_r A||\bar{x}|)_i}{\min_i(|D_r A||\bar{x}|)_i}.$$

Scaling Rule: (R. D. Skeel)
Assume that $\min_i(|A||x|)_i > 0$. Then scale the rows of A and b by $D_r = \text{diag}\,(d_i)$, where

$$d_i = 1/(|A||x|)_i, \quad i = 1, \ldots, n. \tag{7.7.33}$$

A measure of **ill-scaling** of the system $Ax = b$ is

$$\sigma(A, x) = \max_i(|A||x|)_i / \min_i(|A||x|)_i. \tag{7.7.34}$$

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 $|x| = (1, \ldots, 1)^T$ in Eq (7.7.33), which corresponds to row equilibration. Note that this assumption is violated in Example 7.7.5.

7.7.8 Iterative Refinement of Solutions

So far we have considered ways of *estimating* the accuracy of computed solutions. We now consider methods for *improving* the accuracy. Let $\bar{x}$ be any approximate solution to the linear system of equations $Ax = b$ and let $r = b - A\bar{x}$ be the corresponding residual vector. Then one can attempt to improve the solution by solving the system $A\delta = r$ for a correction δ and taking $x_c = \bar{x} + \delta$ as a new approximation. If no further rounding errors are performed in the computation of δ this is the exact solution. Otherwise this refinement process can be iterated. In floating-point arithmetic with base β this process of **iterative refinement** can be described as follows:

$$s := 1; \quad x^{(s)} := \bar{x};$$

repeat
$r^{(s)} := b - Ax^{(s)}$; (inprecision $u_2 = \beta^{-t_2}$)
solve $A\delta^{(s)} = r^{(s)}$; (inprecision $u_1 = \beta^{-t_1}$)
$x^{(s+1)} := x^{(s)} + \delta^{(s)}$;
$s := s + 1$;
end

When $\bar{x}$ has been computed by Gaussian elimination this approach is attractive since we can use the computed factors $\bar{L}$ and $\bar{U}$ to solve for the corrections

$$\bar{L}(\bar{U}\delta^{(s)}) = r^{(s)}, \quad s = 1, 2, \ldots.$$

The computation of $r^{(s)}$ and $\delta^{(s)}$, therefore, only takes $n^2 + 2 \cdot \frac{1}{2}n^2 = 2n^2$ flops, which is an order of magnitude less than the $n^3/3$ 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 $t_2 = t_1$.

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 7.7.7 we have

$$\bar{L}\bar{U} = A + E, \quad \|E\|_\infty \leq n^2 g_n u_1 \|A\|,$$

and g_n is the growth factor. It follows that an upper bound for the initial rate of convergence is given by

$$\rho = \|(\bar{L}\bar{U})^{-1}E\|_\infty \leq n^2 g_n u_1 \kappa(A).$$

When also rounding errors in computing the residuals $r^{(s)}$ and the corrections $\delta^{(s)}$ are taken into account, the analysis becomes much more complicated. The behavior of iterative refinement, using t_1-digits for the factorization and $t_2 = 2t_1$ digits when computing the residuals, can be summed up as follows:

1. Assume that A is not too ill-conditioned so that the first solution has some accuracy, $\|x - \bar{x}\|/\|x\| \approx \beta^{-k} < 1$ in some norm. Then the relative error diminishes by a factor of roughly β^{-k} with each step of refinement until we reach a stage at which $\|\delta_c\|/\|x_c\| < \beta^{-t_1}$, when we may say that the solution is correct to working precision.

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* reflected in a corresponding decrease in the norm of the residual, which stays 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 (7.7.34) is large, or if the pivot strategy has been chosen for the preservation of sparsity, see Section 7.6.

Example 7.7.6. As an illustration consider again the badly scaled system in Example 7.7.4

$$\tilde{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 2 \cdot 10^{-6} & 2 \cdot 10^{-6} \\ 1 & 2 \cdot 10^{-6} & -10^{-6} \end{pmatrix}, \qquad \tilde{b} = \begin{pmatrix} 3 + 3 \cdot 10^{-6} \\ 6 \cdot 10^{-6} \\ 2 \cdot 10^{-6} \end{pmatrix},$$

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 (7.7.20) 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 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 7.7.11. (R. D. Skeel.)

As long as the product of $\text{cond}(A^{-1}) = \||A||A^{-1}|\|_\infty$ *and* $\sigma(A,x)$ *is sufficiently less than* $1/u$*, where* u *is the machine unit, it holds that*

$$(A+\delta A)x^{(s)} = b+\delta b, \quad |\delta a_{ij}| < 4n\epsilon_1|a_{ij}|, \quad |\delta b_i| < 4n\epsilon_1|b_i|, \tag{7.7.35}$$

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 [16, 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 (7.7.16) 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.

Review Questions

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 $\kappa(A)$.

2. The result of a roundoff error analysis of Gaussian elimination can be stated in the form of a backward error analysis. Formulate this result. (You don't need to know the precise expression of the constants involved.)

Problems

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 right hand side 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).

2. Use the result in Theorem 7.7.5 to obtain the lower bound $\kappa_\infty(A) \geq 3/(2|\epsilon|) = 1.5|\epsilon|^{-1}$ 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 $\kappa_\infty(A) = 1.5(1 + |\epsilon|^{-1})$.)

3. Show that the matrix A in Example 7.7.3 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.

4. Let y be a vector for which $\|y\|_2 = 1$ and $\|A^{-1}y\|_2 = \|A^{-1}\|_2$. Take

$$w = A^{-1}y/\|A^{-1}y\|_2^2,$$

and put $\delta A = -yw^T$. Show that $A + \delta A$ is singular and $\kappa_2(A) = \|A\|_2/\|\delta A\|_2$.

5. Let $\bar{x}$ be a computed solution, and let δA be 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

$$\delta A = r(s_1, \ldots, s_n)/\|\bar{x}\|_1, \quad \delta A = r(0, \ldots, 0, s_m, 0, \ldots, 0)/\|\bar{x}\|_\infty,$$

where $r = b - A\bar{x}$, $\|\bar{x}\|_\infty = |x_m|$, and for $i = 1, \ldots, n$, and $s_i = 1$, if $x_i \geq 0$; $s_i = -1$, if $x_i < 0$.

7.8 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 superfast methods are generally either bad or unknown we consider only fast methods in the following.

7.8.1 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 $\leq n - 1$ was shown to lead to a linear system of equations with matrix $M = [p_j(\alpha_i)]_{i,j=1}^m$. In the case of the power basis $p_j(z) = z^{j-1}$, the matrix M equals V^T, where V is the **Vandermonde matrix**

$$V = [\alpha_j^{i-1}]_{i,j=1}^n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{pmatrix}. \tag{7.8.1}$$

Hence the unique polynomial $P(z)$ satisfying the interpolating conditions $P(\alpha_i) = f_i$, $i = 1, \ldots, n$ is given by

$$P(z) = (1, z, \ldots, z^{n-1})a,$$

where a is the solution of the dual Vandermonde system.

$$V^T a = f \tag{7.8.2}$$

One of the most efficient ways to determine $P(x)$ is by Newton's interpolation formula, which uses the basis polynomials

$$Q_1(z) = 1, \qquad Q_k(z) = (z - \alpha_1) \cdots (z - \alpha_{k-1}), \quad k = 2, \ldots, n-1.$$

We write the polynomial in the form

$$P(z) = (Q_1(z), Q_2(z), \ldots, Q_n(z))c,$$

where c are the divided differences of $f_1, \ldots, f_n$. These divided differences can be recursively computed, see Section 4.?. This leads to the algorithm below for computing the coefficient vector a in the power basis. Note that the algorithm operates directly on the α_j's and the matrix V^T is never formed,

Algorithm 7.8.1 Dual Vandermonde System

Given distinct scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$ and $f = (f_1, f_2, \ldots, f_n)^T$ the following algorithm solves the dual Vandermonde system $V^T a = f$:

```
a = dvand(α, f)
  a := f;
  for k = 1 : n − 1
    for j = n : (−1) : k + 1
      a_j := (a_j − a_{j−1})/(α_j − α_{j−k})
    end
  end
  for k = n − 1 : (−1) : 1
    for j = k : n − 1
      a_j := a_j − α_k ∗ a_{j+1}
    end
  end
```

The accuracy of this algorithm depends on the ordering of the interpolation points α_i. Often the best ordering is the monotone ordering for which

$$\alpha_1 < \alpha_2 \cdots < \alpha_n.$$

If moreover $0 \leq \alpha_1$ this algorithm often gives remarkably accurate solutions.

To interpret the Newton interpolation algorithm in matrix terms we define lower bidiagonal matrices

$$L_k(\alpha) = \begin{pmatrix} I_{k-1} & 0 \\ 0 & B_{n-k+1}(\alpha) \end{pmatrix}, \quad k = 1, \ldots, n-1,$$

where

$$B_p(\alpha) = \begin{pmatrix} 1 & & & \\ -\alpha & 1 & & \\ & \ddots & \ddots & \\ & & -\alpha & 1 \end{pmatrix} \in \mathbf{R}^{p \times p}.$$

We further let

$$D_k = \operatorname{diag}(1, \ldots, 1, (\alpha_{k+1} - \alpha_1), \ldots, (\alpha_n - \alpha_{n-k})).$$

Then we find that the dual Vandermonde algorithm can be written as

$$\begin{aligned} c &= U^T f, \qquad U^T = D_{n-1}^{-1} L_{n-1}(1) \cdots D_1^{-1} L_1(1), \\ a &= L^T c, \qquad L^T = L_1^T(\alpha_1) L_2^T(\alpha_2) \cdots L_{n-1}^T(\alpha_{n-1}). \end{aligned}$$

Systems of equations with Vandermonde matrix

$$Vx = b \tag{7.8.3}$$

are called primal Vandermonde systems and occur, e.g., in approximation of linear functionals (see Chapter 4). The matrix representation of the algorithm for the dual Vandermonde system allows us to derive an algorithm also for solving primal Vandermonde systems.

Since $a = V^{-T} f = L^T U^T f$, we have $V^{-T} = L^T U^T$. Transposing this relation and find

$$V^{-1} = UL,$$

Hence the solution to the primal system $Vx = b$ is given by $x = V^{-1} b = U(Lb)$ or

$$\begin{aligned} d &= Lb, \qquad L = L_{n-1}(\alpha_{n-1}) \cdots L_2(\alpha_2) L_1(\alpha_1) \\ x &= Uf, \qquad U = M_1^T D_1^{-1} \cdots M_{n-1}^T D_{n-1}^{-1} \end{aligned}$$

This gives rise to the following algorithm:

Algorithm 7.8.2 Primal Vandermonde System

Given distinct scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$ and $b = (b_1, b_2, \ldots, b_n)^T$ the following algorithm solves the primal Vandermonde system $Vx = b$:

$x = \operatorname{pvand}(\alpha, b)$
  $x := b;$
  for $k = 1 : n-1$
    for $j = n : (-1) : k+1$

```
      x_j := x_j − α_k * x_{j−1}
    end
  end
  for k = n − 1 : (−1) : 1
    for j = k + 1 : n
      x_j := x_j/(α_j − α_{j−k})
    end
    for j = k : n − 1
      x_j := x_j − x_{j+1}
    end
  end
```

The above algorithms solve primal and dual Vandermonde systems with only $\frac{1}{2}n(n+1)(3A+2M)$ operations, where A and M denotes one floating point addition and multiplication, respectively. Note also that no extra storage is needed since a can overwrite f.

7.8.2 Toeplitz and Hankel matrices

Note: The following subsection are not yet complete and will be amended.

A **Toeplitz matrix** T is a matrix whose entries are constant along every diagonal; $T = (t_{i-j})_{1\le i,j\le n}$,

$$T = \begin{pmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_{-1} & t_0 & \dots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{-n+1} & t_{-n+2} & \dots & t_0 \end{pmatrix} \in \mathbf{R}^{n\times n},$$

and is defined by the $2n-1$ values of $t_{-n+1},\dots,t_0,\dots,t_{n-1}$. Toeplitz matrices arising in applications are often large, and dimensions of $10,000$ not uncommon. Consequently there is a need for special fast methods for solving Toeplitz systems. In large problems also storage requirements are important. The original matrix T only requires $2n-1$ storage. However, if standard factorization methods are used, at least $n(n+1)/2$ storage is needed.

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. Hence methods developed for solving Toeplitz systems apply also to Hankel systems.

7.8.3 Cauchy-like matrices

A **Cauchy matrix** is a matrix of the following form:

$$C = \left(\frac{1}{y_i - z_j} \right)_{1 \le i,j \le n}, \qquad a_i, b_j \in \mathbf{R}^p. \tag{7.8.4}$$

where we assume that $y_i \neq z_j$ for $1 \le i, j \le n$.

Example 7.8.1. 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 (7.8.4).

Cauchy gave in 1841 the following explicit expression for the determinant

$$\det(C) = \frac{\prod_{1 \le i < j \le n} (y_j - y_i)(z_j - z_i)}{\prod_{1 \le i \le j \le n} (y_j + z_i)}.$$

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

$$C = \left(\frac{a_i^T b_j}{y_i - z_j} \right)_{1 \le i,j \le n}, \qquad a_i, b_j \in \mathbf{R}^p. \tag{7.8.5}$$

Example 7.8.2. 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 $\kappa_2(H_n) \sim e^{3.5n}$.

Notes

Although the history of Gaussian elimination goes back at least to Chinese mathematicians about 250 B.C. Before the advent of computers in the 1940s there was no practical experience of solving large linear systems. In 1946 there was a mood of pessimism about the stability of Gaussian elimination. Hotelling 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, and A. M. Turing published more relevant error bounds. The final form of stability analysis of Gaussian elimination was later given by J. H. Wilkinson.

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.

The literature on linear algebra is very extensive. For a theoretical treatise a classical source is Gantmacher [8, 1959]. Several nonstandard topics are covered in depth in two excellent volumes by Horn and Johnson [12, 1985] and [13, 1991].

An interesting survey of classical numerical methods in linear algebra can be found in Faddeev and Faddeeva [7, 1963], but many of the methods treated are now dated. A compact, lucid and modern presentation is given in Householder [14, 1964]. Bellman [2, 1970] 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 [10, 1996]. The book by Higham [11, 1995] 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 [17, 1998]. For results on on perturbation theory and related topics a very complete reference book is Stewart and Sun [18, 1990]. In particular, an elegant treatise on norms and metrics is found in [18, Chapter II].

The programs and discussions in Wilkinson and Reinsch [21, 1971] are very instructive. For an introduction to the implementation of algorithms for vector and parallel computers, see also Dongarra et al. [5, 1991]. Many important practical details on implementation of algorithms can be found in the documentation of LINPACK in Dongarra et al. [4, 1979] and the more recent LAPACK Guide [1, 1995]. Direct methods for sparse symmetric positive definite systems are covered in George and Liu [9, 1981], while a more general treatise is given by Duff et al. [6, 1986].

LAPACK was designed to supersede and integrate the algorithms in LINPACK and EISPACK. The software was restructured to achieve much greater efficiency, where possible, on modern high-performance computers. LAPACK routines are written so that as much as possible of the computations is performed by calls to the Basic Linear Algebra Subprograms (BLAS). This enables the LAPACK routines to achieve its high performance with portable code. The LAPACK routines were initially released in 1992 (see [1]) and later adapted to be the backbone of Matlab.

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. For more information on LAPACK95 go to http://www.netlib.org/lapack95/.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, editors. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999.

[2] R. Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, New York, 1970.

[3] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson. An estimate for the condition number of a matrix. *SIAM J. Numer. Anal.*, 16:368–375, 1979.

[4] J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. SIAM, Philadelphia, PA, 1979.

[5] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst. *Numerical Linear Algebra for High Performance Computers*. SIAM, Philadelphia, PA, 1991.

[6] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, London, 1986.

[7] D. K. Faddeev and V. N. Faddeeva. *Computational Methods of Linear Algebra*. W. H. Freeman, San Francisco, CA, 1963.

[8] F. R. Gantmacher. *The Theory of Matrices. Vols. I and II*. Chelsea Publishing Co, New York, 1959.

[9] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[10] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[11] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, 1996.

[12] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[13] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, Cambridge, UK, 1991.

[14] A. S. Householder. *The Theory of Matrices in Numerical Analysis.* Dover, New York, 1964.

[15] R. D. Skeel. Scaling for stability in Gaussian elimination. *J. Assoc. Comput. Mach.*, 26:494–526, 1979.

[16] R. D. Skeel. Iterative refinement implies numerical stability for Gaussian elimination. *Math. Comput.*, 35:817–832, 1980.

[17] G. W. Stewart. *Matrix Algorithms Volume I: Basic Decompositions.* SIAM, Philadelphia, PA, 1998.

[18] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, Boston, MA, 1990.

[19] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. ACM*, 8:281–330, 1961.

[20] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[21] J. H. Wilkinson and C. Reinsch, editors. *Handbook for Automatic Computation. Vol. II, Linear Algebra.* Springer-Verlag, New York, 1971.

Chapter 8

Linear Least Squares Problems

8.1 Introduction

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 **overdetermined** 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. A choice which can often be motivated for statistical reasons (see Theorem 8.1.4) 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), $i = 1, \ldots, m$, $m > n$. 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 $y_i = \sum_{j=1}^n x_j \phi_j(t_i)$, $i = 1, \ldots, m$ form an overdetermined sys-

tem, which can be written in matrix form $Ax = b$, where $a_{ij} = \phi_j(t_i)$, and $b_i = y_i$.

8.1.2 Linear Models and the Gauss–Markoff Theorem

We first need to introduce some concepts from statistics. Let y be a random variable having the distribution function $F(y)$,[1] where $F(y)$ is nondecreasing, right continuous, and

$$0 \le F(y) \le 1, \quad F(-\infty) = 0, \quad F(\infty) = 1.$$

Then the expected value and the variance of y are defined as 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 $y = (y_1, \ldots, y_n)^T$ is a vector of random variables and $\mu = (\mu_1, \ldots, \mu_n)^T$, $\mu_i = \mathcal{E}(y_i)$, 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 variance-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 8.1.1.

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

$$\mathcal{E}(By) = B\mu, \qquad \mathcal{V}(By) = BVB^T.$$

In the special case that $B = b^T$ is a row vector, $r = 1$, then $\mathcal{V}(b^T y) = \mu \|b\|_2^2$.

Proof. The first property follows directly from the definition of expected value. The second follows from the relation

$$\begin{aligned}\mathcal{V}(By) &= \mathcal{E}[(B(y-\mu)(y-\mu)^T B^T] \\ &= B\mathcal{E}[(y-\mu)(y-\mu)^T]B^T = BVB^T.\end{aligned}$$

□

[1] $F(x)$ is the probability that $y \le x$.

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

$$Ax = b + \epsilon, \tag{8.1.2}$$

where $A \in \mathbf{R}^{m\times n}$ is a known matrix, and, ϵ is a vector of random errors. In the **standard case** ϵ has zero mean and covariance matrix $\sigma^2 I$, i.e.,

$$\mathcal{E}(\epsilon) = 0, \qquad \mathcal{V}(\epsilon) = \sigma^2 I.$$

We also assume that rank $(A) = n$, and make the following definitions:

Definition 8.1.2.
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 8.1.3.
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 gave the method of least squares a sound theoretical basis in [9, 1821], without any assumptions that the random variables follow a normal distribution. This contribution of Gauss was somewhat neglected until rediscovered by Markoff 1912. We state the relevant theorem without proof.

Theorem 8.1.4. The Gauss–Markoff theorem.
Consider the linear model (8.1.2), *where $A \in \mathbf{R}^{m\times n}$ is a known matrix, and ϵ is a random vector with zero mean and covariance matrix $\mathcal{V}(\epsilon) = \sigma^2 I$. Let $\hat{x}$ be the least square estimator, obtained by minimizing over x the sum of squares $\|Ax - b\|_2^2$. Then the best linear unbiased estimator of any linear function $g = c^T x$ is $c^T \hat{x}$. Furthermore, the covariance matrix of the estimate $\hat{x}$ equals*

$$\mathcal{V}(\hat{x}) = V = \sigma^2 (A^T A)^{-1} \tag{8.1.3}$$

and $\mathcal{E}(s^2) = \sigma^2$, where s^2 is the quadratic form

$$s^2 = \frac{1}{m-n} \|b - A\hat{x}\|_2^2.$$

Proof. See Zelen [26]. □

In the next subsection we show that the residual vector $\hat{r} = \hat{b} - Ax$ satisfies $A^T \hat{r} = 0$. Hence there are 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.,

$$\mathcal{V}(\hat{r}, \hat{x}) = 0, \qquad \mathcal{V}(s^2, \hat{x}) = 0.$$

In the **general univariate linear model** the covariance matrix is $\mathcal{V}(\epsilon) = \sigma^2 W$, where $W \in \mathbf{R}^{m\times m}$ is a positive semidefinite symmetric matrix. For full column rank A and positive definite W the best unbiased linear estimate is the solution of

$$\min_x (Ax-b)^T W^{-1}(Ax-b).$$

In particular, if the errors are uncorrelated with variances $w_{ii} > 0$, $i = 1, \ldots, m$, then W is diagonal and this problem is equivalent to the **weighted least squares** problem

$$\min_x \|D^{-1}(Ax-b)\|_2, \quad D = \operatorname{diag}(\sqrt{w_{11}}, \ldots, \sqrt{w_{mm}}). \tag{8.1.4}$$

Hence if the ith equation is scaled by $1/\sqrt{w_{ii}}$ we get the standard case. This is consistent with the obvious observation that 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 linear system is *consistent*, i.e., unless $b \in \mathcal{R}(A)$. For a fuller treatment of weighted and the general linear model we refer to Björck [3, Chap.3].

8.1.3 Characterization of Least Squares Solutions

We begin by characterizing the set of all solutions to the problem (8.1.1).

Theorem 8.1.5.

The vector x minimizes $\|b-Ax\|_2$ if and only if the residual vector $r = b-Ax$ is orthogonal to $\mathcal{R}(A)$, or equivalently

$$A^T(b-Ax) = 0. \tag{8.1.5}$$

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 (8.1.5) 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 $\|b-Ax\|_2$. ∎

Theorem 8.1.5 shows that any least squares solution x decomposes the right hand side b into two orthogonal components

$$b = Ax + r, \qquad r \perp Ax. \tag{8.1.6}$$

Here Ax is the orthogonal projection onto $\mathcal{R}(A)$ and $r \in \mathcal{N}(A^T)$; see Fig. 8.2.1. Note that although the least squares solution x may not be unique the decomposition in (8.1.6) always is unique,

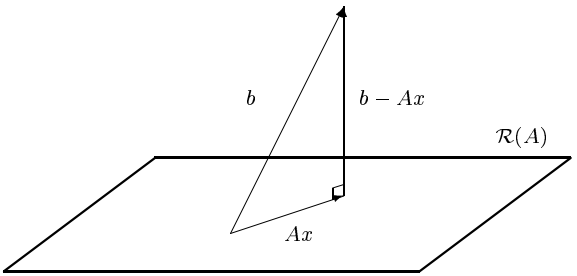


Figure 8.1.1. *Geometric characterization of the least squares solution.*

The above characterization of a least squares solution immediately leads to a classical method for solving the least squares problem (8.1.1). Multiplying in the factor A^T in (8.1.5) it follows that a least squares solution always satisfies the **normal equations**

$$A^T Ax = A^T b. \tag{8.1.7}$$

Here $A^T A \in \mathbf{R}^{n\times n}$ is a symmetric, positive semidefinite matrix. The normal equations are always *consistent* since

$$A^T b \in \mathcal{R}(A^T) = \mathcal{R}(A^T A),$$

and therefore a least squares solution always exists.

We now give a condition for the least squares solution to be unique.

Theorem 8.1.6.

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 x is unique and given by*

$$x = (A^T A)^{-1} A^T b. \tag{8.1.8}$$

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 (8.1.8) follows. □

In the full rank case, $\operatorname{rank}(A) = n$, the residual $r = b - Ax$ can be written

$$r = b - P_{\mathcal{R}(A)} b, \qquad P_{\mathcal{R}(A)} = A(A^T A)^{-1} A^T, \tag{8.1.9}$$

which gives an expression for $P_{\mathcal{R}(A)}$, the orthogonal projector onto $\mathcal{R}(A)$, the range space of A. It follows that any solution to the consistent linear system

$$Ax = P_{\mathcal{R}(A)} b \tag{8.1.10}$$

is a least squares solution.

In more general least squares problems $Ax = b$ we can have $\operatorname{rank}(A) < n$, and then A has a nontrivial nullspace. In this case if $\hat{x}$ is any vector that minimizes $\|Ax - b\|_2$, then the set of all least squares solutions is

$$\mathcal{S} = \{x = \hat{x} + y \mid y \in \mathcal{N}(A)\}. \tag{8.1.11}$$

In this set there is a unique solution of minimum norm characterized by $x \perp \mathcal{N}(A)$; cf. Theorem 8.3.8. Such problems will be considered in Section 8.3.3

8.1.4 Generalized Least Squares

A unified exposition of least squares and minimum norm problems is obtained by considering augmented linear systems of the form

$$\begin{pmatrix} B & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix} \tag{8.1.12}$$

with $A \in \mathbf{R}^{m\times n}$, $m \geq n$, and symmetric, positive definite $B \in \mathbf{R}^{m\times m}$. Such systems occur in many application areas, since they represents the condition for equilibrium of a physical system. The system (8.1.12) is also called a saddle-point system or, in optimization, a KKT (Karush–Kuhn–Tucker) system.

Theorem 8.1.7. *Assume that x and y satisfy (8.1.12) where $\operatorname{rank}(A) = n$ and B is symmetric and positive definite. Then the linear system (8.1.12) is nonsingular and gives the condition for the solution of the following two problems:*

$$\min_x \frac{1}{2}\|Ax - b\|^2_{B^{-1}} + c^T x, \tag{8.1.13}$$

$$\min_y \frac{1}{2} y^T B y - y^T b, \quad \textit{subject to} \quad A^T y = c. \tag{8.1.14}$$

Proof. The system (8.1.12) can be obtained by differentiating (8.1.13) to give $A^T B^{-1}(Ax - b) + c = 0$, and setting y to be the residual $y = B^{-1}b - Ax$. The system can also be obtained by differentiating the Lagrangian

$$L(x, y) = \frac{1}{2} y^T B y - y^T b + x^T (A^T y - c)$$

of (8.1.14), and equating to zero. Here x is the vector of Lagrange multipliers. ∎

Eliminating y in (8.1.12) gives the **generalized normal equations**

$$A^T B^{-1} A x = A^T B^{-1} b - c. \tag{8.1.15}$$

Using the assumptions, it follows that the matrix $A^T B^{-1} A$ is symmetric, positive definite. The normal equations can be solved for x, and then y obtained by solving

$$By = b - Ax.$$

Setting $B = I$ and $b = 0$ in (8.1.14) gives the problem of finding the **minimum norm solution** of an underdetermined linear system $A^T y = c$:

$$\min_y \|y\|_2, \qquad A^T y = c, \tag{8.1.16}$$

A vector y is a solution if and only if $A^T y = c$ and $y \in \mathcal{R}(A)$. (or equivalently $y \perp \mathcal{N}(A^T)$) Hence we can write $y = Az$, where $z \in \mathbf{R}^n$ and it follows that z satisfies the **normal equations of second kind**

$$A^T A z = c. \tag{8.1.17}$$

If rank $(A) = n$ then the solution can be written

$$y = Az = A(A^T A)^{-1} c.$$

8.1.5 Projections

From the characterization given in previous sections it is clear that orthogonal projections play a key role in least squares problems and their solution. Therefore we now give a general review of orthogonal and oblique projections and their matrix representations.

Consider a matrix $P \in \mathbf{R}^{n\times n}$ with range space $S = \mathcal{R}(P) \subset \mathbf{R}^n$. P is said to be a **projector** onto the subspace S if and only if it holds:

$$(i) \quad Pb = b \quad \forall \quad b \in S, \qquad (ii) \quad P^2 = P. \tag{8.1.18}$$

Any matrix which satisfies (ii) is called **idempotent**. An arbitrary vector $b \in \mathbf{R}^n$ is then decomposed into two components by

$$b = Pb + (I - P)b = b_1 + b_2, \tag{8.1.19}$$

where $b_1 = Pb \in S$ is a projection of b onto S.

If it also holds that $P^T = P$ then

$$P^T b_2 = P^T (I - P)b = (P - P^2)b = 0,$$

and it follows that $b_2^T b = b_2^T P b = 0$ for all $b \in S$. Hence $b_2 \perp S$, i.e., b_2 lies in the orthogonal complement $S^\perp$ of S; in particular $b_2 \perp b_1$. 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 is unique, see Problem 1. Orthogonal projections will play a central role in the study of least squares problems.

Let P be any projector onto a subspace $S = \mathcal{R}(P)$ of dimension $n_1 < n$. Then there exist two matrices $U_1 \in \mathbf{R}^{n\times n_1}$ and $Y_2 \in \mathbf{R}^{n\times n_2}$, $(n_1 + n_2 = n)$ such that the compound matrix $(U_1\ Y_2)$ is nonsingular and

$$PU_1 = U_1, \qquad (I - P)Y_2 = Y_2.$$

In terms of these matrices the decomposition (8.1.19) can be written

$$b = b_1 + b_2 \equiv (U_1\ Y_2) \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = U_1 c_1 + Y_2 c_2, \tag{8.1.20}$$

We call $b_1$the **oblique projection** of b onto $\mathcal{R}(U_1)$ along $\mathcal{R}(Y_2)$, and the matrix P is the corresponding oblique projector. Similarly $I - P$ is the oblique projector onto $\mathcal{R}(Y_2)$ along $\mathcal{R}(U_1)$. Since

If P is an orthogonal projector the subspaces $\mathcal{R}(U_1)$ and $\mathcal{R}(Y_2)$ are orthogonal, i.e., $U_1^T Y_2 = 0$, and we write $Y_2 = U_2$. The matrix $(U_1\ U_2)$ can be chosen to be orthogonal and then the projectors onto S and $S^\perp$ take the simple form

$$P_S = U_1 U_1^T, \qquad P_{S^\perp} = I - P = U_2 U_2^T. \tag{8.1.21}$$

For an orthogonal projector we have

$$\|Pb\|_2 \le \|b\|_2 \quad \forall \quad b \in \mathbf{R}^m,$$

where equality holds for all vectors in $\mathcal{R}(U_1)$. From this it follows that $\|P\|_2 = 1$.

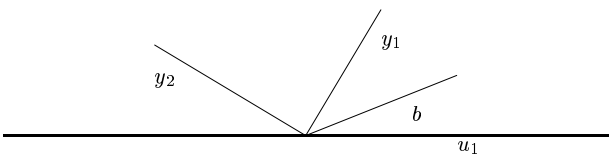


Figure 8.1.2. *The oblique projection of b on u_1 along y_2.*

In the general case, if we denote the inverse of the matrix in (8.1.21)

$$(U_1\ Y_2)^{-1} = \begin{pmatrix} \hat{Y}_1^T \\ \hat{U}_2^T \end{pmatrix},$$

then

$$b_1 = U_1 c_1 = U_1 \hat{Y}_1^T b, \qquad b_2 = Y_2 c_2 = Y_2 \hat{U}_2^T b,$$

and hence $P = U_1 \hat{Y}_1^T$, $I - P = Y_2 \hat{U}_2^T$. Note that from the equality

$$\begin{pmatrix} \hat{Y}_1^T \\ \hat{U}_2^T \end{pmatrix} (U_1\ Y_2) = \begin{pmatrix} \hat{Y}_1^T U_1 & \hat{Y}_1^T Y_2 \\ \hat{U}_2^T U_1 & \hat{U}_2^T Y_2 \end{pmatrix} = \begin{pmatrix} I_{n_1} & 0 \\ 0 & I_{n_2} \end{pmatrix}. \tag{8.1.22}$$

it follows that $\hat{Y}_1^T Y_2 = 0$ and hence the columns of $\hat{Y}_1$ form a basis of the orthogonal complement of $\mathcal{R}(Y_2)$. Similarly the columns of $\hat{U}_2$ form a basis of the orthogonal complement of $\mathcal{R}(U_1)$.

For any matrix $Y_1 \in \mathbf{R}^{n \times n_1}$ whose columns span $\mathcal{R}(\hat{Y}_1)$ we can write $\hat{Y}_1 = Y_1 G$. From (8.1.22) it follows that $G^T Y_1^T U_1 = I_{n_1}$. Hence $G^T = (Y_1^T U_1)^{-1}$ and the projector P can be written

$$P = U_1 (Y_1^T U_1)^{-1} Y_1^T. \tag{8.1.23}$$

Similarly for any matrix $U_2 \in \mathbf{R}^{n \times n_2}$ whose columns span $\mathcal{R}(\hat{U}_2)$ we have

$$I - P = Y_2 (U_2^T Y_2)^{-1} U_2^T. \tag{8.1.24}$$

In (8.1.23)–(8.1.24) it is always possible to take (U_1, U_2) and (Y_1, Y_2) to be orthogonal matrices.

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 θ be the angle between u_1 and y_1, see Fig. 8.1.1. Since $y_1^T u_1 = \cos\theta$ we have

$$P = u_1(y_1^T u_1)^{-1} y_1^T = \frac{1}{\cos\theta} u_1 y_1^T.$$

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.

Review Questions

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. State the Gauss–Markov theorem.
3. Show that the matrix $P = A(A^TA)^{-1}A^T$ satisfies the three conditions in (8.1.18) for an orthogonal projector.

Problems

1. 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 $z \in \mathbf{R}^n$
   $$\|(P_1 - P_2)z\|_2^2 = (P_1 z)^T(I - P_2)z + (P_2 z)^T(I - P_1)z = 0.$$
2. Denote by x_w the solution to the weighted least squares problem (8.1.4) and let x be the solution to the corresponding unweighted problem ($W = I$). Using the normal equations show that
   $$x_w - x = (A^TW^{-1}A)^{-1}A^T(W^{-1} - I)(b - Ax). \tag{8.1.25}$$
   Conclude that weighting the rows affects the solution if $b \notin \mathcal{R}(A)$.
3. Show the equivalence of the hyperbolic and the Givens rotations in (8.5.14).

8.2 The Method of Normal Equations

8.2.1 Forming and Solving the Normal Equations

The classical method of solving linear least squares problems by forming and solving the normal equations (8.1.7) dates back to Gauss. We now discuss the numerical

implementation of this method. We assume throughout that $\text{rank}\,(A) = n$, and defer treatment of rank deficient problems to later chapters.

The first step is to compute the elements of the symmetric matrix $C = A^TA$ and vector $d = A^Tb$. If $A = (a_1, a_2, \ldots, a_n)$ has been partitioned by columns, we can use the inner product formulation

$$c_{jk} = (A^TA)_{jk} = a_j^Ta_k, \qquad d_j = (A^Tb)_j = a_j^Tb, \quad 1 \le j \le k \le n. \tag{8.2.1}$$

Since C is symmetric it is only necessary to compute and store its lower (or upper) triangular part. This requires $\frac{1}{2}n(n+1)m+mn$ multiplications. Note that if $m \gg n$, then since $\frac{1}{2}n(n+1) \ll mn$ the number of elements in the upper triangular part of A^TA is much smaller than the number of elements in A. Hence in this case the formation of A^TA can be viewed as a data compression!

The formulas in (8.2.1) are not suitable for large problems, where the matrix A is held in secondary storage, since each column needs to be accessed many times. By sequencing the operations differently we get an alternative row oriented algorithm, which uses only *one pass* through the data (A, b), and no more storage than that needed for A^TA and A^Tb. If we denote by $\tilde{a}_i^T$, the ith row of A, $i = 1, \ldots, m$, then we obtain

$$C = A^TA = \sum_{i=1}^{m} \tilde{a}_i\tilde{a}_i^T, \qquad d = A^Tb = \sum_{i=1}^{m} b_i\tilde{a}_i. \tag{8.2.2}$$

This is an outer product form, where A^TA is expressed as the sum of m matrices of rank one and A^Tb as a linear combination of the rows of A. As remarked in Sec. 6.2.2 the outer product form is also preferable if the matrix A is sparse. Note that both formulas simplify slightly if we adjoin b to A and form $(A, b)^T(A, b)$.

We now consider the solution of the system of normal equations. The matrix $C = A^TA$ is symmetric, and if $\text{rank}\,(A) = n$ also positive definite. Then the Cholesky factorization

$$C = A^TA = R^TR,$$

exists, where R is upper triangular and nonsingular. It can be computed by one of the algorithms given in Sec. 6.4.3 in $n^3/6$ multiplications. Given R the least squares solution is obtained by solving the two triangular systems

$$R^Tz = d, \qquad Rx = z. \tag{8.2.3}$$

which requires about n^2 multiplications. Thus the total number of multiplications required to solve the least squares problem by the method of normal equations equals (neglecting lower order terms) $\frac{1}{2}mn^2 + \frac{1}{6}n^3$.

In many least squares problems the matrix A has the property that in each row all nonzero elements are contained in a narrow band. For such banded 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 i-th row of A, i.e.,

$$f_i = \min\{j \mid a_{ij} \neq 0\}, \qquad l_i = \max\{j \mid a_{ij} \neq 0\}. \tag{8.2.4}$$

Then the matrix A is said to have row bandwidth w, where

$$w = \max_{1 \le i \le m} w_i, \qquad w_i = (l_i - f_i + 1). \tag{8.2.5}$$

Alternatively w is the smallest number for which it holds that

$$a_{ij} a_{ik} = 0, \quad \text{if} \quad |j - k| \ge w. \tag{8.2.6}$$

For this structure to have practical significance we need to have $w \ll n$. Matrices of small row bandwidth often occur naturally, since they correspond to a situation where only variables "close" to each other are coupled by observations. We now prove a relation between the row bandwidth of the matrix A and the bandwidth of the corresponding matrix of normal equations $A^T A$.

Theorem 8.2.2.
Assume that the matrix $A \in \mathbf{R}^{m \times n}$ has row bandwidth w. Then the symmetric matrix $A^T A$ has bandwidth $r \le w - 1$.

Proof. From the Definition 8.2.1 it follows that $a_{ij} a_{ik} \ne 0 \Rightarrow |j - k| < w$. Hence,

$$|j - k| \ge w \Rightarrow (A^T A)_{jk} = \sum_{i=1}^{m} a_{ij} a_{ik} = 0.$$

□

If the matrix A also has full column rank it follows that we can use the band Cholesky Algorithm 6.4.6 to solve the normal equations.

8.2.2 Estimating the Covariance

The covariance matrix estimate in (8.1.3) can be expressed in terms of the Cholesky factor as

$$V = \sigma^2 (A^T A)^{-1} = \sigma^2 (R^T R)^{-1} = \sigma^2 R^{-1} R^{-T}.$$

In order to assess the accuracy of the computed least squares estimate of x it is often required to compute the matrix V, or part of it. The upper triangular matrix $S = R^{-1}$ can be obtained by back-substitution in the triangular system $RS = I$, and then $V = \sigma^2 S S^T$. Often just the variances of of the components of the least squares solution x are required, which equal $\operatorname{diag}(V) = \sigma^2 \operatorname{diag}(SS^T)$. These elements are the 2-norms squared of the rows of S and can be computed by

$$v_{ii} = \sigma^2 \sum_{j=i}^{n} s_{ij}^2, \quad i = 1, 2, \ldots, n.$$

In many situations the matrix V 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 $\sigma^2 v$, where

$$v = f^T V f = f^T R^{-1} R^{-T} f = z^T z, \quad z = R^{-T} f.$$

Thus to compute v we only need to solve the triangular system $R^T z = f$ and form $z^T z$. This is a more stable and efficient approach than using the expression $f^T V f$.

We have $r - \hat{r} = -A(A^T A)^{-1} A^T \epsilon$, where $\hat{r} = b - A\hat{x}$ is the least squares residual and ϵ the random error in the model. Hence $r - \hat{r}$ has variance-covariance matrix

$$V_r = \sigma^2 (A(A^T A)^{-1} A^T)^2 = \sigma^2 A(A^T A)^{-1} A^T = \sigma^2 P_{\mathcal{R}(A)}.$$

Note that the orthogonal projector can be computed from

$$P_{\mathcal{R}(A)} = A(R^T R)^{-1} A^T = QQ^T, \qquad Q = AR^{-1}.$$

The **normalized residuals** are defined by

$$\tilde{r} = (\operatorname{diag}(V_r))^{-1/2} \hat{r}.$$

Large components in $\tilde{r}$ can be assumed to correspond to "bad" data.

8.2.3 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 $\bar{C} = fl(A^T A)$

$$\bar{c}_{ij} = fl\left(\sum_{k=1}^{m} a_{ik} a_{jk} \right) = \sum_{k=1}^{m} a_{ik} a_{jk} (1 + \delta_k),$$

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

$$\bar{C} = A^T A + E, \qquad |e_{ij}| < 1.06um \sum_{k=1}^{m} |a_{ik}||a_{jk}|. \tag{8.2.7}$$

A similar estimate holds for the rounding errors in the computed vector $A^T b$. 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 $A^T A$ 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 $A^T A$ 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 8.2.1. *(*LÄUCHLI*)* Consider the system $Ax = b$, where

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \epsilon & & \\ & \epsilon & \\ & & \epsilon \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad |\epsilon| \ll 1.$$

This may correspond to a problem where the sum $x_1 + x_2 + x_3$ has been determined much more accurately than the individual components of x. We have, exactly

$$A^TA = \begin{pmatrix} 1+\epsilon^2 & 1 & 1 \\ 1 & 1+\epsilon^2 & 1 \\ 1 & 1 & 1+\epsilon^2 \end{pmatrix}, \qquad A^Tb = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$$x = \frac{1}{3+\epsilon^2}(1,1,1)^T, \qquad r = \frac{\epsilon}{3+\epsilon^2}(\epsilon, -1-1-1)^T.$$

Now assume that $\epsilon = 10^{-4}$, and that we use eight-digit decimal floating point arithmetic. Then $1 + \epsilon^2 = 1.00000001$ rounds to 1, and the computed matrix A^TA will be singular. Note that we have lost all information contained in the last three rows of A!

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

$$\|\bar{x} - x\|_2 \leq 2.5n^{3/2}u\kappa(A^TA)\|x\|_2. \tag{8.2.8}$$

As we shall see in Sec. 8.3.4, the condition number for the least squares problem normally is $\kappa^{1/2}(A^TA)$. Hence *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.

Example 8.2.2. Linear Regression. We want to fit the linear model $y(t) = \alpha + \beta t$ to the given data (y_i, t_i), $i = 1, \ldots, m$. This leads to a overdetermined linear system of equations $\alpha + t_i\beta = y_i$, $i = 1, \ldots, m$. The normal equations are

$$\begin{pmatrix} m & \sum_{i=1}^m t_i \\ \sum_{i=1}^m t_i & \sum_{i=1}^m t_i^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m y_i t_i \end{pmatrix}.$$

Eliminating α we obtain

$$\beta = \Big(\sum_{i=1}^m y_i t_i - m\bar{y}\bar{t}\Big) \Big/ \Big(\sum_{i=1}^m t_i^2 - m\bar{t}^2\Big),$$

where

$$\bar{y} = \frac{1}{m}\sum_{i=1}^m y_i, \qquad \bar{t} = \tfrac{1}{m}\sum_{i=1}^m t_i. \tag{8.2.9}$$

are the mean values. The first equation $\alpha + \beta\bar{t} = \bar{y}$ shows that the point $(\bar{y}, \bar{t})$ lies on the fitted line, which determines α.

A more accurate formula for β is obtained by making the change of variable $\tilde{t} = t - \bar{t}$, and writing the model as $y(t) = \tilde{\alpha} + \tilde{\beta}\tilde{t}$. Then $\sum_{i=1}^m \tilde{t}_i = 0$, i.e., the matrix of normal equation is diagonal. Using the identity $\sum_{i=1}^m (y_i - \bar{y})(t_i - \bar{t}) = \sum_{i=1}^m y_i(t_i - \bar{t})$ we find

$$\alpha = \bar{y} - \beta\bar{t}, \qquad \beta = \textstyle\sum_{i=1}^m (y_i - \bar{y})(t_i - \bar{t}) \Big/ \sum_{i=1}^m (t_i - \bar{t})^2. \tag{8.2.10}$$

8.2.4 Partitioned Least Squares Problems

Consider the least squares problem $\min_x \|Ax - b\|_2^2$, $A \in \mathbf{R}^{m\times n}$. Assume that A has full column rank and partition the unknowns into two groups

$$\min_{x_1,x_2} \left\| (A_1 \ A_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - b \right\|_2^2,$$

with n_1 and n_2 components, respectively, $n = n_1 + n_2$. Then the normal equations can be written

$$\begin{pmatrix} A_1^T A_1 & A_1^T A_2 \\ A_2^T A_1 & A_2^T A_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} A_1^T b \\ A_2^T b \end{pmatrix}. \tag{8.2.11}$$

If here the variables x_1 are eliminated we obtain for x_2 the reduced normal equations

$$(A_2^T A_2 - A_2^T A_1 (A_1^T A_1)^{-1} A_1^T A_2) x_2 = A_2^T b - A_2^T A_1 (A_1^T A_1)^{-1} A_1^T b.$$

or taking out common factors

$$A_2^T (I - A_1 (A_1^T A_1)^{-1} A_1^T) A_2 x_2 = A_2^T (I - A_1 (A_1^T A_1)^{-1} A_1^T) b,$$

Notice that by (8.1.9) these equations can be written

$$A_2^T P_{\mathcal{N}(A_1^T)} A_2 x_2 = P_{\mathcal{N}(A_1^T)} b.$$

where $P_{\mathcal{N}(A_1^T)} = (I - A_1 (A_1^T A_1)^{-1} A_1^T)$ is the orthogonal projection onto $\mathcal{N}(A_1^T)$. These equations are the normal equations for the least squares problem

$$\min_{x_2} \|P_{\mathcal{N}(A_1^T)} A_2 x_2 - P_{\mathcal{N}(A_1^T)} b\|_2. \tag{8.2.12}$$

When x_2 is known x_1 can be computed from the first block row in (8.2.11) which gives $A_1^T A_1 x_1 = A_1^T (b - A_2 x_2)$, or from the least squares problem

$$\min_{x_1} \|A_1 x_1 - (b - A_2 x_2)\|_2. \tag{8.2.13}$$

8.2.5 Scaling of Least Squares Problems

In Sec. 7.6.7 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 (8.2.8) 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

$$\kappa'(A^TA) = \min_{D>0} \kappa(DA^TAD). \tag{8.2.14}$$

The following result by van der Sluis [1969] shows the scaling where D is chosen so that in $D(A^TA)D$ all column norms are equal, i.e. $D = \text{diag}(\|a_1\|_2, \ldots, \|a_n\|_2)^{-1}$, comes within a factor of n of the minimum value.

Theorem 8.2.3. *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 8.2.3. 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 $t_i = 0, 1, \ldots, 20$. The condition numbers are

$$\kappa(A^TA) = 4.10 \cdot 10^{13}, \qquad \kappa(DA^TAD) = 4.93 \cdot 10^6.$$

Thus, the condition number of the matrix of normal equations is reduced by about seven orders of magnitude by this scaling.

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. 6.5.8. 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

$$r_s := b - Ax_s, \qquad R^TR\delta x_s = A^Tr_s,$$
$$x_{s+1} := x_s + \delta x_s.$$

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

$$\bar{\rho} = cu\kappa'(A^TA), \tag{8.2.15}$$

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 8.2.4. 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:

$$10^{5.3}, 10^{6.4}, 10^{8}, \qquad p = 1, 2, \infty.$$

Hence for moderately ill-conditioned problems the normal equations combined with iterative refinement can give very good accuracy. For more ill-conditioned problems the QR method described in Sec. 8.4 is usually to be preferred.

Review Questions

1. 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. Discuss how the accuracy of the method of normal equations can be improved by (a) scaling the columns of A, (b) iterative refinement.
3. Show that the more accurate formula in Example 8.2.2 can be interpreted as a special case of the method (8.2.12)–(8.2.13) for partitioned least squares problems.

Problems

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. (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 8.4.1, 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.

3. The recently discovered comet 1968 Tentax is supposed to move within the solar system. The following observations of its position in a certain polar coordinate system have been made

r	2.70	2.00	1.61	1.20	1.02
ϕ	48°	67°	83°	108°	126°

By Kepler's first law the comet should move in a plane orbit of elliptic or hyperbolic form, if the perturbations from planets are neglected. Then the coordinates satisfy $r = p/(1 - e\cos\phi)$, where p is a parameter and e the eccentricity. Estimate p and e by the method of least squares from the given observations.

Hint: If the relationship is rewritten as $1/p - (e/p)\cos\phi = 1/r$, it becomes linear in the parameters $1/p$ and e/p.

4. (S. M. Stiegler [23].) 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. If the earth is ellipsoidal, then to a

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″

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 $1/e$.

5. 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

$$\bar{M} = \bar{A}^T\bar{A} = \begin{pmatrix} M & d \\ d^T & b^Tb \end{pmatrix}, \qquad \bar{R} = \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}.$$

Show that the solution x and the residual norm ρ to the linear least squares problem $\min_x \|b - Ax\|_2$ is given by $Rx = z$, $\|b - Ax\|_2 = \rho$.

6. Let $A \in \mathbf{R}^{m\times n}$ and $\operatorname{rank}(A) = n$. Show that the minimum norm solution of the underdetermined system $A^Ty = c$ 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 $y = Ax$.

8.3 Least Squares and the SVD

8.3.1 The Singular Value Decomposition

The **singular value decomposition** (SVD) of a matrix $A \in \mathbf{R}^{m\times n}$ provides a diagonal form of a matrix A under an orthogonal equivalence transformation. This matrix decomposition is of great theoretical and practical importance. Although its history goes back more than a century its use in numerical computations is much more recent.

Theorem 8.3.1. (Singular Value Decomposition.) *Every matrix $A \in \mathbf{R}^{m\times n}$ of rank r can be written*

$$A = U\Sigma V^T, \qquad \Sigma = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \in \mathbf{R}^{m\times n}, \tag{8.3.1}$$

where $U \in \mathbf{R}^{m\times m}$ *and* $V \in \mathbf{R}^{n\times n}$ *are orthogonal,* $\Sigma_1 = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, *and*

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

(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*

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

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

Proof. (After Golub and Van Loan [1989, p.71].) Let $v_1 \in \mathbf{R}^n$ be a vector such that

$$\|v_1\|_2 = 1, \qquad \|Av_1\|_2 = \|A\|_2 = \sigma_1.$$

The existence of such a vector follows from the definition of an operator norm $\|A\|$, see equation (6.2.13). If $\sigma_1 = 0$, then $A = 0$, and we can take $\Sigma = 0$, and U and V arbitrary orthogonal matrices. Therefore assume that $\sigma_1 > 0$, and take $u_1 = (1/\sigma_1)Av_1 \in \mathbf{R}^m$, $\|u_1\|_2 = 1$. Let the matrices

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

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

$$A_1 \equiv U^T AV = \begin{pmatrix} \sigma_1 & w^T \\ 0 & B \end{pmatrix},$$

where $w^T = u_1^T AV_1$ and $B = U_1^T AV_1 \in \mathbf{R}^{(m-1)\times(n-1)}$. Since U and V are orthogonal it follows that $\|A_1\|_2 = \|A\|_2 = \sigma_1$. But by the definition of matrix norm,

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

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

We remark that a singular value decomposition $A = U\Sigma V^H$, with U and V unitary, and Σ real diagonal, holds for any **complex** matrix $A \in \mathbf{C}^{m\times n}$.

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

The singular values of A are uniquely determined. The singular vector v_j, $j \leq r$, is unique (up to a factor ± 1) if σ_j^2 is a *simple* eigenvalue of $A^T A$. 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, $1 \le j \le r$ have been chosen, the vectors u_j, $1 \le j \le r$ are uniquely determined, and vice versa, using

$$Av_j = \sigma_j u_j, \qquad A^T u_j = \sigma_j v_j, \qquad j = 1, \ldots, r. \tag{8.3.2}$$

If U and V are partitioned according to

$$U = (U_1,\ U_2), \quad U_1 \in \mathbf{R}^{m\times r}, \quad V = (V_1,\ V_2), \quad V_1 \in \mathbf{R}^{n\times r}. \tag{8.3.3}$$

then the SVD can be written in the compact form

$$A = U_1 \Sigma_1 V_1^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T. \tag{8.3.4}$$

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

From (6.2.20) it follows that

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

Thus σ_j^2, $j = 1, \ldots, r$ are the nonzero eigenvalues of the symmetric and positive semidefinite matrices $A^T A$ and AA^T, and v_j and u_j are the corresponding eigenvectors. Hence in principle the SVD can be reduced to the eigenvalue problem for symmetric 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 the singular values are square roots of the eigenvalues.

A very useful relationship between the SVD and a symmetric eigenvalue problem is given in the following theorem.

Theorem 8.3.2. *Let the SVD of $A \in \mathbf{R}^{m\times n}$ be $A = U\Sigma V^T$, where $U \in \mathbf{R}^{m\times m}$, $V \in \mathbf{R}^{n\times n}$ are orthogonal. Let $r = \operatorname{rank}(A)$ and let U, V be partitioned as in (8.3.3). Then*

$$C = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} = Q \begin{pmatrix} \Sigma_1 & 0 & 0 \\ 0 & -\Sigma_1 & 0 \\ 0 & 0 & 0 \end{pmatrix} Q^T, \tag{8.3.5}$$

where $\Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) > 0$ and

$$Q = \frac{1}{\sqrt{2}} \begin{pmatrix} U_1 & U_1 & \sqrt{2}U_2 & 0 \\ V_1 & -V_1 & 0 & \sqrt{2}V_2 \end{pmatrix}^T. \tag{8.3.6}$$

is orthogonal. Hence the eigenvalues of C are $\pm\sigma_1, \pm\sigma_2, \ldots, \pm\sigma_r$, and zero repeated $(m + n - 2r)$ times.

Proof. Form the product on the right hand side of (8.3.5) and note that $A = U_1 \Sigma_1 V_1^T$ and $A^T = V_1 \Sigma_1 U_1^T$. ∎

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

$$\mathcal{R}(A) = \mathcal{R}(U_1) \qquad \mathcal{N}(A^T) = \mathcal{R}(U_2) \tag{8.3.7}$$

Since $A^T = V\Sigma^T U^T$ it follows that also

$$\mathcal{R}(A^T) = \mathcal{R}(V_1) \qquad \mathcal{N}(A) = \mathcal{R}(V_2). \tag{8.3.8}$$

We immediately find the well-known relations

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

If $S = \text{span}\,(U)$ and $U = (u_1, \ldots, u_k)$ is orthogonal, $U^T U = I$, then it is easily seen that the orthogonal projector onto S can be written $P = UU^T$. It follows that the orthogonal projectors onto the four fundamental subspaces of A can be expressed in terms of the singular vectors of A as

$$\begin{aligned} P_{\mathcal{R}(A)} &= U_1 U_1^T, \qquad & P_{\mathcal{N}(A^T)} &= U_2 U_2^T, \\ P_{\mathcal{R}(A^T)} &= V_1 V_1^T, \qquad & P_{\mathcal{N}(A)} &= V_2 V_2^T. \end{aligned} \tag{8.3.9}$$

The singular values have the following important extremal property, the **minimax characterization**.

Theorem 8.3.3.
Let $A \in \mathbf{R}^{m\times n}$ have singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$, $p = \min(m,n)$, and S be a linear subspace of $\mathbf{R}^n$ of dimension $\dim(S)$. Then

$$\sigma_i = \min_{dim(S)=n-i+1} \max_{\substack{x\in S \\ x\neq 0}} \frac{\|Ax\|_2}{\|x\|_2}. \tag{8.3.10}$$

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 8.3.4.
Let $A, B \in \mathbf{R}^{m\times n}$ have singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p$ and $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_p$ respectively, where $p = \min(m,n)$. Then

$$\max_i |\sigma_i - \tau_i| \leq \|A - B\|_2, \tag{8.3.11}$$

$$\sum_{i=1}^{p} |\sigma_i - \tau_i|^2 \leq \|A - B\|_F^2. \tag{8.3.12}$$

Proof. See Stewart [1973, 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 Sec. 8.3.5.

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 (8.3.5) corresponding results can be derived for the singular values of a matrix A.

Theorem 8.3.5.

Let

$$\hat{A} = (A, u) \in \mathbf{R}^{m\times n}, \quad m \geq n, \quad u \in \mathbf{R}^m.$$

Then the ordered singular values σ_i of A interlace the ordered singular values $\hat{\sigma}_i$ of $\hat{A}$ as follows

$$\hat{\sigma}_1 \geq \sigma_1 \geq \hat{\sigma}_2 \geq \sigma_2 \ldots \geq \hat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \hat{\sigma}_n.$$

Similarly, if A is bordered by a row,

$$\hat{A} = \begin{pmatrix} A \\ v^* \end{pmatrix} \in \mathbf{R}^{m\times n}, \quad m > n, \quad v \in \mathbf{R}^n,$$

then

$$\hat{\sigma}_1 \geq \sigma_1 \geq \hat{\sigma}_2 \geq \sigma_2 \ldots \geq \hat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \hat{\sigma}_n \geq \sigma_n.$$

8.3.2 Matrix Approximation

The SVD plays an important role in a number of matrix approximation problems. In the theorem below we consider the approximation of one matrix by another of lower rank.

Theorem 8.3.6. *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. See Mirsky [1960] for the l_2, norm and Eckhard and Young [1936] for the Frobenius norm. □

According to this theorem σ_i equals the distance in l_2 norm to the nearest matrix of rank $i-1$, $i \leq \min(m, n)$. In particular $\sigma_1 = \|A\|_2$.

Let $A \in \mathbf{R}^{m\times n}$, be a matrix of rank n with the "thin" SVD $A = U_1\Sigma V^T$. Since $A = U_1\Sigma V^T = U_1\Sigma U_1^T U_1 V^T$ we have

$$A = PH, \qquad P = U_1 V^T, \qquad H = V\Sigma V^T, \tag{8.3.13}$$

where $P \in \mathbf{R}^{m\times n}$ has orthogonal columns, and $H \in \mathbf{R}^{n\times n}$ is symmetric, positive semidefinite. The decomposition (8.3.13) is called the **polar decomposition** of A, since it can be regarded as a generalization to matrices of the complex number representation $z = re^{i\theta}$, $r \geq 0$.

The significance of the factor P in the polar decomposition is the fact that it is the closest matrix with orthogonal columns to A.

Theorem 8.3.7.

Let $\mathcal{M}_{m\times n}$ denote the set of all matrices in $\mathbf{R}^{m\times n}$ with orthogonal columns. Let $A = PH$, where $P \in \mathcal{M}_{m\times n}$ and H is symmetric positive semidefinite. Then for any matrix $Q \in \mathcal{M}_{m\times n}$,

$$\|A - Q\|_F \geq \|A - P\|_F.$$

Proof. This theorem was proved for $m = n$ and general unitarily invariant norms by Fan and Hoffman [7]. The generalization to $m > n$ follows from the additive property of the Frobenius norm. □

8.3.3 The SVD and Pseudoinverse Solutions

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 of vectors, and we have the following fundamental result.

Theorem 8.3.8.

Consider the general linear least squares problem

$$\min_{x\in S} \|x\|_2, \qquad S = \{x \in \mathbf{R}^n \mid \|b - Ax\|_2 = \min\}, \tag{8.3.14}$$

where $A \in \mathbf{R}^{m\times n}$ and rank $(A) = r \leq \min(m, n)$. This problem always has a unique solution, which can be written in terms of the SVD of A as

$$x = V \begin{pmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T b, \qquad A = U \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} V^T. \tag{8.3.15}$$

Proof. Let

$$z = V^T x = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \qquad c = U^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix},$$

where $z_1, c_1 \in \mathbf{R}^r$. Then, using the orthogonal invariance of the l_2 norm we have

$$\begin{aligned}\|b - Ax\|_2 &= \|U^T(b - AVV^T x)\|_2 \\ &= \left\| \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} - \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} c_1 - \Sigma_r z_1 \\ c_2 \end{pmatrix} \right\|_2.\end{aligned}$$

The residual norm will attain its minimum value equal to $\|c_2\|_2$ for $z_1 = \Sigma_r^{-1} c_1$, z_2 arbitrary. Obviously the choice $z_2 = 0$ minimizes $\|x\|_2 = \|Vz\|_2 = \|z\|_2$. □

Note that problem (8.3.14) includes as special cases the solution of both overdetermined and underdetermined linear systems. We can write (8.3.15)

$$x = A^\dagger b, \qquad A^\dagger = V \begin{pmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T \in \mathbf{R}^{n \times m} \tag{8.3.16}$$

is the **pseudoinverse** of A, and x is called the pseudoinverse 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 $r = \text{rank}\,(A)$ columns. The pseudoinverse solution (8.3.16) can then be written

$$x = V_1 \Sigma_r^{-1} U_1^T b = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} \cdot v_i, \quad r = \text{rank}\,(A). \tag{8.3.17}$$

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**:

$$\begin{aligned} &(1) \quad AXA = A, && (2) \quad XAX = X, \\ &(3) \quad (AX)^T = AX, && (4) \quad (XA)^T = XA. \end{aligned}$$

It can be directly verified that $X = A^\dagger$ given by (8.3.16) 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 simple expressions in terms of the pseudoinverse (cf. (8.3.9))

$$\begin{aligned} P_{\mathcal{R}(A)} &= AA^\dagger, & P_{\mathcal{N}(A^T)} &= I - AA^\dagger, \\ P_{\mathcal{R}(A^T)} &= A^\dagger A, & P_{\mathcal{N}(A)} &= I - A^\dagger A. \end{aligned} \tag{8.3.18}$$

These expressions are easily verified using the definition of an orthogonal projection and the Penrose conditions.

Another very useful characterization of the pseudoinverse solution is the following:

Theorem 8.3.9. *The pseudoinverse solution $x = A^\dagger b$ is uniquely characterized by the two geometrical conditions*

$$x \perp \mathcal{N}(A), \qquad Ax = P_{\mathcal{R}(A)}\, b. \tag{8.3.19}$$

Proof. These conditions are easily verified from (8.3.17). □

In the special case that $A \in \mathbf{C}^{m\times n}$, $m \geq n$ and $\text{rank}\,(A) = n$ it holds

$$A^\dagger = (A^TA)^{-1}A^T, \qquad (A^T)^\dagger = A(A^TA)^{-1} \tag{8.3.20}$$

These expressions follow from the normal equations (8.1.7) and (8.1.17). Some properties of the usual inverse can be extended to the pseudoinverse, e.g., the relations

$$(A^\dagger)^\dagger = A, \qquad (A^T)^\dagger = (A^\dagger)^T,$$

easily follow form (8.3.16). However, in general $(AB)^\dagger \neq B^\dagger A^\dagger$, $AA^\dagger \neq A^\dagger A$. The following theorem gives a useful *sufficient* conditions for the relation $(AB)^\dagger = B^\dagger A^\dagger$ to hold.

Theorem 8.3.10.
If $A \in \mathbf{R}^{m\times r}$, $B \in \mathbf{R}^{r\times n}$, and $\text{rank}\,(A) = \text{rank}\,(B) = r$, then

$$(AB)^\dagger = B^\dagger A^\dagger = B^T(BB^T)^{-1}(A^TA)^{-1}A^T. \tag{8.3.21}$$

Proof. The last equality follows from (8.3.20). The first equality is verified by showing that the four Penrose conditions are satisfied. □

8.3.4 Perturbation Analysis

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 8.3.11.
Let $A \in \mathbf{R}^{m\times n}$ have rank $r > 0$ and singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. 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

$$A^TA = V\Sigma^T(U^TU)\Sigma V^T = V\begin{pmatrix}\Sigma_r^2 & 0\\ 0 & 0\end{pmatrix}V^T. \tag{8.3.22}$$

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 we have $\operatorname{rank}(A + \delta A) = n$. Let the perturbed solution be $x + \delta x$ and $r + \delta r$, where $r = b - Ax$ is the residual vector. Then, neglecting second order perturbations, we have

$$\delta r = \delta b - (A + \delta A)(x + \delta x) = (\delta b - \delta A\, x) - A\delta x.$$

The perturbed solution satisfies

$$(A + \delta A)^T\big((A + \delta A)(x + \delta x) - (b + \delta b)\big) = 0.$$

Subtracting $A^T(Ax - b) = 0$ and neglecting second order perturbations, we get

$$\delta x = (A^TA)^{-1}A^T(\delta b - \delta A\, x) + (A^TA)^{-1}\delta A^T r, \tag{8.3.23}$$

$$\delta r = (I - A(A^TA)^{-1}A^T)(\delta b - \delta A x) - A(A^TA)^{-1}\delta A^T\, r, \tag{8.3.24}$$

Here we can identify

$$(A^TA)^{-1}A^T = A^\dagger, \quad I - A(A^TA)^{-1}A^T = I - AA^\dagger = P_{\mathcal{N}(A^T)}, \quad A(A^TA)^{-1} = (A^\dagger)^T.$$

Using (8.3.16) and (8.3.22) 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 (8.3.24) and (8.3.24) we obtain

$$\|\delta x\|_2 \le \frac{1}{\sigma_n}(\|\delta b\|_2 + \|\delta A\|_2\|x\|_2) + \frac{1}{\sigma_n^2}\|\delta A\|_2\|r\|_2, \tag{8.3.25}$$

$$\|\delta r\|_2 \le \|\delta b\|_2 + \|\delta A\|_2\|x\|_2 + \frac{1}{\sigma_n}\|\delta A\|_2\|r\|_2. \tag{8.3.26}$$

A more refined analysis (see Wedin [1973]) gives the following result:

Theorem 8.3.12.

Let x be the least squares solution to the overdetermined linear system $Ax = b$. Assume that $\operatorname{rank}(A) = n$ *and let δA and δb be perturbations such that the condition*

$$\eta = \|A^\dagger\|_2\|\delta A\|_2 = \kappa\epsilon_A < 1, \qquad \epsilon_A = \|\delta A\|_2/\|A\|_2, \tag{8.3.27}$$

holds. Then $\operatorname{rank}(A + \delta A) = n$, *and the perturbations δx and δr satisfy*

$$\|\delta x\|_2 \le \frac{\kappa}{1 - \eta}\left(\epsilon_A\|x\|_2 + \frac{\|\delta b\|_2}{\|A\|_2} + \epsilon_A\kappa\frac{\|r\|_2}{\|A\|_2}\right), \tag{8.3.28}$$

$$\|\delta r\|_2 \le \epsilon_A\|x\|_2\|A\|_2 + \|\delta b\|_2 + \epsilon_A\kappa\|r\|_2. \tag{8.3.29}$$

It can be shown that for an arbitrary matrix A and vector b there are perturbations δA and δb such that the estimates in Theorem 8.3.12 are almost attained. Note the term proportional to κ^2 which occurs in the bound for $\|\delta x\|_2$ when $r \neq 0$. However, *for small residual problems which are only moderately ill-conditioned*, or more precisely when

$$\kappa\|r\|_2 < \|A\|_2\|x\|_2,$$

the κ^2 term does not dominate the error. Then $\kappa(A)$ is an approximate condition number of the linear least squares problem. If the system is consistent then $r = 0$ and (8.3.28) reduces to the bound (6.6.5) for the linear equation case.

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 of the form $\|\delta A\|_2 \leq \epsilon_A\|A\|_2$, $\|\delta A\|_2 \leq \epsilon_b\|b\|_2$. If the columns in $A = (a_1, a_2, \ldots, a_n)$ have widely differing norms, then a much better estimate may often be obtained by applying (8.3.28) 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.,

$$\tilde{A} = AD^{-1}, \qquad \tilde{x} = Dx, \qquad D = \operatorname{diag}(\|a_1\|_2, \ldots, \|a_n\|_2).$$

As remarked in Sec. 8.2.3, this column scaling approximately minimizes $\kappa(AD^{-1})$ over $D > 0$.

If the *rows* in A differ widely in norm, then (8.3.28) may also considerably overestimate the perturbation in x. As remarked above, we cannot scale the rows in A without changing the least squares solution. In this case it is better to consider the effect of component-wise perturbations

$$|\delta A| \leq \omega E, \qquad |\delta b| \leq \omega f. \tag{8.3.30}$$

Substituting in (8.3.24) yields the bound (terms of order $O(\omega^2)$ neglected)

$$|\delta x| \leq \omega|A^\dagger|(f + E|x|) + \omega|(A^TA)^{-1}|E^T|r|.$$

In particular, if $E = |A|$, $f = |b|$, we obtain using the norm $\|\cdot\|_\infty$ (cf. (6.6.13))

$$\|\delta x\|_\infty \leq \omega\| \, |A^\dagger|(|A||x| + |b|)\|_\infty + \omega\| \, |(A^TA)^{-1}||A|^T|r| \, \|_\infty. \tag{8.3.31}$$

8.3.5 Numerical Rank and Truncated SVD

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. In this section we discuss how to assign a **numerical rank** to a matrix and how algorithms should be modified to cope with rank deficiency and ill-conditioning.

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*.

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 8.3.13.
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, $i = 1, 2, \ldots, n$ 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 nullspace** *of A.*

It follows from Theorem 8.3.4, 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 8.3.13 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 $m > n$.

Example 8.3.1. Consider an example based on the integral equation of the first kind

$$\int_1^1 k(s,t)f(s)ds = g(t), \quad k(s,t) = e^{-(s-t)^2},$$

on $-1 \le t \le 1$. To compute $g(t)$ given $f(s)$ is well conditioned problem. However, the inverse problem of reconstructing $f(s)$ given $g(t)$ is a very ill-conditioned problem.

The equation can be discretized using a uniform mesh on [-1,1] and the trapezoidal rule, giving a finite-dimensional linear system $Kf = g$, where $K \in \mathbf{R}^{n\times n}$, and $f, g \in \mathbf{R}^n$. For $n = 100$ the singular values σ_k of the matrix K are displayed in logarithmic scale in Figure 8.3.1. Note that for $k > 30$ all σ_k are close to roundoff level, so the numerical rank of K certainly is smaller than 30. Hence the linear system $Kf = g$ is too ill-conditioned to be solved without more information about the solution f.

It follows from Theorem 8.3.4, 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

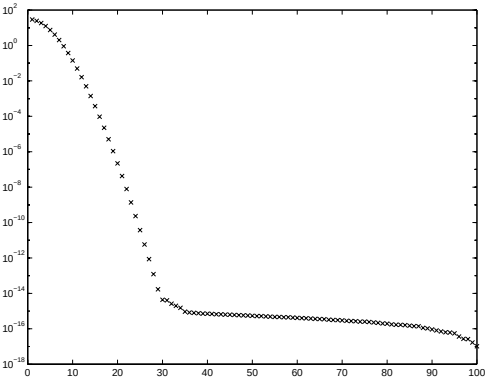


Figure 8.3.1. *Singular values of the matrix K.*

cannot *lower* the rank. Definition 8.3.13 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!

The choice of the parameter δ in Definition 8.3.13 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 8.3.13 is not appropriate. One could then scale the rows and columns of A so that the magnitude of the errors become nearly equal. (Note that any such diagonal scaling $D_r A D_c$ will induce the same scaling $D_r E D_c$ of the error matrix.)

We now consider solving the linear least squares problem

$$\min_x \|Ax - b\|_2, \tag{8.3.32}$$

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 (8.3.17) corresponding to singular values smaller or equal to δ, and take the solution as the **truncated SVD (TSVD) solution**

$$x(\delta) = \sum_{\sigma_i > \delta} \frac{c_i}{\sigma_i} v_i. \tag{8.3.33}$$

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

$$\|A - A_k\|_2 = \|AV_2\|_2 \leq \delta, \quad V_2 = (v_{k+1}, \ldots, v_n).$$

Review Questions

1. 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.
2. (a) Let $A \in \mathbf{R}^{m\times n}$ with $m < n$. Show that A^TA is singular.
   (b) Show, using the SVD, that $\operatorname{rank}(A^TA) = \operatorname{rank}(AA^T) = \operatorname{rank}(A)$.
3. 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) $\mathcal{R}(B) \subseteq \mathcal{R}(AB)$.
4. (a) What are the four Penrose conditions for X to be the pseudoinverse of A?
   (b) Give two geometric conditions which are necessary and sufficient conditions for x to be the pseudoinverse solution of $Ax = b$.
5. 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

$$\tilde{A} = (A, u), \qquad \hat{A} = \begin{pmatrix} A \\ v^T \end{pmatrix}?$$

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?
7. How is the *numerical* rank of a matrix A defined? Give an example where the numerical rank is not well determined.

Problems

1. (a) Compute the pseudoinverse $x^\dagger$ of a column vector x.
   (b) Take $A = (\,1 \quad 0\,)$, $B = (\,1 \quad 1\,)^T$, and show that $1 = (AB)^\dagger \neq B^\dagger A^\dagger = 1/2$.
2. (a) Verify that the Penrose conditions define the matrix X uniquely. Do it first for $A = \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, and then transform the result to a general matrix A.
   (b) Use the Penrose conditions to prove the formula ($\otimes$ denotes the Kronecker product)

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger.$$

3. 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 $x_1 = A_1^\dagger(b - A_2x_2)$.

4. (a) Show that the matrix $A \in \mathbf{R}^{m\times n}$ has a **left inverse** $A^L \in \mathbf{R}^{n\times m}$, i.e., $A^L A = I$, if and only if $\text{rank}(A) = n$. Although in this case $Ax = b \in \mathcal{R}(A)$ has a unique solution, the left inverse is not unique. Find the general form of Σ^L and generalize the result to A^L.

(b) Discuss the **right inverse** A^R in a similar way.

5. Show that $A^\dagger$ minimizes $\|AX - I\|_F$.

6. 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 $\|B - A\|_2 = \epsilon$, $\|B^\dagger - A^\dagger\|_2 > 1/\epsilon$.

7. Show that for any matrix A it holds

$$A^\dagger = \lim_{\mu\to 0}(A^T A + \mu^2 I)^{-1} A^T = \lim_{\mu\to 0} A^T (AA^T + \mu^2 I)^{-1}. \tag{8.3.34}$$

8. (a) Let $A = (a_1, a_2)$, where $a_1^T a_2 = \cos\gamma$, $\|a_1\|_2 = \|a_2\|_2 = 1$. Hence γ is the angle between the vectors a_1 and a_2. Determine the singular values and right singular vectors v_1, v_2 of A by solving the eigenvalue problem for

$$A^T A = \begin{pmatrix} 1 & \cos\gamma \\ \cos\gamma & 1 \end{pmatrix}.$$

Then determine the left singular vectors u_1, u_2 from (8.3.2).

(b) Show that if $\gamma \ll 1$, then $\sigma_1 \approx \sqrt{2}$ and $\sigma_2 \approx \gamma/\sqrt{2}$ and

$$u_1 \approx (a_1 + a_2)/2, \qquad u_2 \approx (a_1 - a_2)/\gamma.$$

8.4 Gram–Schmidt Orthogonalization

8.4.1 Gram–Schmidt Algorithms

We will now develop methods for solving linear least squares problems, which avoid the squaring of the condition number that results from forming the normal equations. We start by describing the Gram–Schmidt orthogonalization algorithm, one of the most fundamental procedures in linear algebra.

Algorithm 8.4.1 Classical Gram–Schmidt.

Given a sequence of linearly independent vectors $a_1, a_2, \ldots, a_n$ the following algorithm computes orthonormal vectors $q_1, q_2, \ldots, q_n$. such that span $[a_1, \ldots, a_k] =$ span $[q_1, \ldots, q_k]$, $k = 1, \ldots, n$.

for $k = 1, \ldots, n$ do

(i) If $k > 1$ then set $\hat{q}_k = a_k$ else orthogonalize a_k against $q_1, \ldots, q_{k-1}$

$$\hat{q}_k = a_k - \sum_{i=1}^{k-1} r_{ik} q_i, \qquad r_{ik} = q_i^T a_k, \quad i = 1, \ldots, k-1. \tag{8.4.1}$$

(ii) Normalize $\hat{q}_k$

$$r_{kk} = (\hat{q}_k^T \hat{q}_k)^{1/2}, \qquad q_k = \hat{q}_k / r_{kk}. \tag{8.4.2}$$

end;

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. This algorithm requires mn^2 multiplications.

The Gram–Schmidt algorithm can be interpreted in matrix terms. It computes the following factorization of the matrix $A = (a_1, a_2, \ldots, a_n)$:

Theorem 8.4.1. *The QR Factorization*

Let the matrix $A = (a_1, a_2, \ldots, a_n) \in \mathbf{R}^{m\times n}$ have linearly independent columns. Then the Gram–Schmidt algorithm computes unique matrices Q_1 and R such that

$$A = Q_1 R, \tag{8.4.3}$$

where $Q_1 = (q_1, q_2, \ldots, q_n) \in \mathbf{R}^{m\times n}$ has orthonormal columns and $R = (r_{ik})$ is upper triangular with positive diagonal elements.

Proof. Combining (8.4.1) and (8.4.2) we obtain

$$a_k = r_{kk} q_k + \sum_{i=1}^{k-1} r_{ik} q_i = \sum_{i=1}^{k} r_{ik} q_i, \quad k = 1, \ldots, n,$$

which is equivalent with (8.4.3). Since the vectors q_k are mutually orthogonal by construction the theorem follows. □

The GS algorithm computes the Cholesky factor of $A^T A$ directly from A.

Corollary 8.4.2. *The factor R in the factorization* (8.4.3) *equals the Cholesky factor of $A^T A$.*

Proof. We recall that the Cholesky factor R of $A^T A$, is unique provided we normalize R to have a positive diagonal. Now, from (8.4.3) we have $A^T A = R^T Q_1^T Q_1 R = R^T R$, and and the result follows. □

For the *numerical* GS factorization of a matrix A a small reordering of the above algorithm gives the *modified* Gram–Schmidt (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 algorithm** proceeds in n steps, $k = 1, \ldots, n$. At the beginning of step k we have computed

$$(q_1, \ldots, q_{k-1}, a_k^{(k)}, \ldots, a_n^{(k)}),$$

where we have put $a_j = a_j^{(1)}$, $j = 1, \ldots, n$. Here $a_k^{(k)}, \ldots, a_n^{(k)}$ have been made orthogonal to $q_1, \ldots, q_{k-1}$, which are final columns in Q_1. In the kth step q_k is

obtained by normalizing the vector $a_k^{(k)}$,

$$\tilde{q}_k = a_k^{(k)}, \qquad r_{kk} = (\tilde{q}_k^T \tilde{q}_k)^{1/2}, \qquad q_k = \tilde{q}_k / r_{kk}, \tag{8.4.4}$$

Then $a_{k+1}^{(k)}, \ldots, a_n^{(k)}$ are orthogonalized against q_k

$$a_j^{(k+1)} = a_j^{(k)} - r_{kj} q_k, \qquad r_{kj} = q_k^T a_j^{(k)}, \quad j = k+1, \ldots, n. \tag{8.4.5}$$

After n steps we have obtained the factorization (8.4.3). We summarize the MGS algorithm below.

Algorithm 8.4.2 Modified Gram–Schmidt.
Given $A \in R^{m \times n}$ with $\text{rank}(A) = n$ the following algorithm computes the factorization $A = Q_1 R$:

```
for k = 1 : n
    q̂_k = a_k^(k);
    r_kk = (q̂_k^T q̂_k)^{1/2};
    q_k = q̂_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 8.4.1 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 Section 8.6.3.

There is also a **square root free** version of the modified Gram–Schmidt orthogonalization method, which results if the normalization of the vectors $\tilde{q}_k$ is omitted. In this version one computes $\tilde{Q}_1 = (\tilde{q}_1, \ldots, \tilde{q}_n)$ and $\tilde{R}$ so that $A = \tilde{Q}_1 \tilde{R}$ and $\tilde{R}$ **unit** upper triangular. We take $\tilde{r}_{kk} = 1$, $d_k = \tilde{q}_k^T \tilde{q}_k$, and change (8.4.5) to

$$a_j^{(k+1)} = a_j^{(k)} - \tilde{r}_{kj} \tilde{q}_k, \qquad \tilde{r}_{kj} = \tilde{q}_k^T a_j^{(k)} / d_k, \qquad j = k+1, \ldots, n. \tag{8.4.6}$$

The unnormalized vector $\tilde{q}_k$ is just the orthogonal projection of a_k onto the complement of $\text{span}[a_1, a_2, \ldots, a_{k-1}] = \text{span}[q_1, q_2, \ldots, q_{k-1}]$.

8.4.2 Loss of Orthogonality in Gram–Schmidt

In CGS the orthogonalization of a_k in step (8.4.1) 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 $n > 2$ these two expressions are identical only if the $q_1, \ldots, q_{k-1}$ are accurately orthogonal. However, due to round-off there will be a gradual (sometimes catastrophic) loss of orthogonality. In this respect CGS and MGS behave very differently. In MGS the loss of orthogonality occurs in a predictable manner related to the conditioning of the matrix A. This is not the case for CGS.

We will show that a loss of orthogonality occurs whenever cancellation takes place in subtracting the orthogonal projection on q_i from $a_k^{(i)}$, that is when

$$a_k^{(i+1)} = a_k^{(i)} - r_{ik}q_i, \qquad \|a_k^{(i+1)}\|_2 \le \alpha \|a_k^{(i)}\|_2, \quad \alpha \ll 1. \tag{8.4.7}$$

Consider the case of orthogonalizing *two* vectors. Given vectors q_1 with $\|q_1\|_2 = 1$, and a_2, we want to compute

$$\tilde{q}_2 = a_2 - r_{12}q_1, \qquad r_{12} = q_1^T a_2. \tag{8.4.8}$$

Using the standard model for floating point computation, where u the unit roundoff, it is easily shown that the computed vector $\hat{q}_2 = \mathrm{fl}\,(\tilde{q}_2)$ satisfies

$$\|\hat{q}_2 - \tilde{q}_2\|_2 < cu\|a_2\|_2, \qquad c = 1.06(2m+3). \tag{8.4.9}$$

Since $q_1^T \tilde{q}_2 = 0$, it follows that $|q_1^T \hat{q}_2| < cu\|a_2\|_2$. Hence the loss of orthogonality is proportional to

$$\frac{\|a_2\|_2}{\|\hat{q}_2\|_2} \approx \frac{\|a_2\|_2}{\|\tilde{q}_2\|_2} = \frac{1}{\sin\phi(q_1, a_2)},$$

where $\phi(q_1, a_2)$ is the angle between q_1 and a_2.

Example 8.4.1. As an illustration consider the matrix

$$A = (a_1, a_2) = \begin{pmatrix} 8 & 21 \\ 13 & 34 \\ 21 & 55 \\ 34 & 89 \end{pmatrix}$$

with condition number $\kappa(A) = 5.8 \cdot 10^3$. Using the Gram–Schmidt algorithm and 4-digit computation we get

$$r_{11} = (a_1^T a_1)^{1/2} = 42.78, \qquad q_1 = (0.1870, 0.3039, 0.4909, 0.7948)^T,$$
$$r_{12} = a_1^T a_2 / r_{11} = 112.0, \qquad a_2^{(2)} = (0.06, -0.04, 0.02, -0.02)^T.$$

Severe cancellation has taken place since $\|a_2^{(2)}\|_2 = 0.07746 \ll \|a_2\|_2 = 112.0$. This leads to a serious loss or orthogonality between q_1 and q_2:

$$q_1^T q_2 = -0.007022/0.07746 = -0.09065.$$

For MGS the loss of orthogonality can be bounded in terms of the condition number $\kappa(A)$ also for $n > 2$. (Note that for $n = 2$ MGS and CGS are the same.) In can be 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 Problem 9.

In some applications it is important that the computed $\bar{Q}_1$ and $\bar{R}$ are such that $\bar{Q}_1\bar{R}$ accurately represents A, and $\bar{Q}_1$ is accurately orthogonal. This is the **orthogonal basis problem**. It is easy to show that

$$A + E = \bar{Q}_1\bar{R}, \qquad \|E\|_2 \le c_0 u \|A\|_2.$$

To satisfy both the second condition it is necessary to **reorthogonalize** the computed vectors in the Gram–Schmidt algorithm, whenever (8.4.7) is satisfied for some constant $\alpha < 1$.

It can be shown that, in a sense made more precise below, that *one reorthogonalization always suffices*. Hence reorthogonalization will at most double the cost of the Gram–Schmidt factorization. Consider again the case $n = 2$, and let $\hat{q}_2$ denote the computed vector before normalization. Let α be a fixed value, typically chosen in the range $[0.1, 0.5]$. Then if $\|\hat{q}_2\|_2 \ge \alpha\|a_2\|_2$, we accept $\hat{q}_2$. Otherwise we reorthogonalize $\hat{q}_2$ against q_1, i.e., compute

$$\delta r_{12} := \mathrm{fl}\,(q_1^T\hat{q}_2), \qquad \check{q}_2 := \mathrm{fl}\,(\hat{q}_2 - \delta r_{12} q_1).$$

If $\|\check{q}_2\|_2 \ge \alpha\|\hat{q}_2\|_2$, then we accept $\hat{q}_2 = \check{q}_2$. Otherwise we conclude that the given vectors (q_1, a_2) are linearly dependent, i.e., we accept $\hat{q}_2 := 0$. The computed $\hat{q}_2$ can be shown to satisfy

$$\|\check{q}_2 - \hat{q}_2\|_2 \le (1+\alpha)\epsilon\|a_2\|_2, \qquad \|q_1^T\hat{q}_2\| \le \epsilon\alpha^{-1}\|\hat{q}_2\|_2. \tag{8.4.10}$$

When α is large, say 0.5, then the bounds in (8.4.10) are very good but reorthogonalization will occur more frequently. If α is small, reorthogonalization will be rarer, but the bound on orthogonality less good. There seems to be a good case for recommending the stringent value $\alpha = 0.5$.

Example 8.4.2. We reorthogonalize the computed vector $a_2^{(2)}$ in Example 8.4.1 against q_1. Using 4-digit computation

$$\tilde{q}_2 := a_2^{(2)} - \delta r_{12} q_1, \qquad \delta r_{12} = q_1^T a_2^{(2)} = -0.007022,$$

which gives $\tilde{q}_2 = (0.06131, -0.03787, 0.02345, -0.01442)^T$. Note that the correction δr_{12} is too small to affect $r_{12} = 112.0$. The new vector $\tilde{q}_2$ is now accurately orthogonal to q_2,

$$q_1^T\tilde{q}_2/\|\tilde{q}_2\|_2 = 0.8902 \cdot 10^{-4}.$$

In the more general case we are given a matrix $Q_1 = (q_1, \ldots, q_{k-1})$ with $\|q_1\|_2 = \ldots = \|q_{k-1}\|_2 = 1$, together with a vector a_k, and want to compute a vector $\hat{q}_k \in \mathrm{span}(Q_1, a_k) \perp Q_1$. The solution equals $\hat{q}_k = a_k - Q_1 r_k$, where r_k solves the least squares problem

$$\min_{r_k} \|a_k - Q_1 r_k\|_2.$$

To solve this problem when the columns of Q_1 need not be accurately orthogonal we can use **iterated** Gram–Schmidt methods, where the CGS or MGS algorithm is repeatedly. In the iterated CGS algorithm we put $\hat{q}_k^{(0)} := a_k$, $r_k^{(0)} := 0$, and for $p = 0, 1, \ldots$ 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 column rank, then with $\alpha = 0.5$ both iterated CGS and MGS gives 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.

8.4.3 Solving Least Squares Problems by Gram–Schmidt

We now consider the use of the Modified Gram–Schmidt algorithm for solving linear least squares problem. It is important to note that because of the loss of orthogonality in Q_1 *computing* x *by forming* $c_1 = Q_1^T b$ *and then solving* $Rx = c_1$ *will not in general give an accurate solution.* Using the MGS factorization in this way seems to have contributed to an undeserved bad reputation of the method. Used correctly, as described below, the MGS factorization will give as accurate results as any competing method.

To solve a least squares problems the MGS algorithm is applied to the augmented matrix (A, b), giving a factorization

$$(A, b) = (Q_1, q_{n+1}) \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}. \tag{8.4.11}$$

Here $r = \rho q_{n+1}$, $\|r\|_2 = \rho$, and we have

$$\|Ax - b\|_2 = \left\| (A, b) \begin{pmatrix} x \\ -1 \end{pmatrix} \right\|_2 = \|Q_1(Rx - z) - \rho q_{n+1}\|_2.$$

Let us assume that q_{n+1} is orthogonal to Q_1. D Then the minimum of the last expression occurs when $Rx - z = 0$ and the least squares residual equals ρq_{n+1}. This assumptions is not strictly true, but note that it is not necessary to assume that Q_1 is orthogonal for the conclusion to hold. This heuristic argument leads to the following algorithm for solving linear least squares problems by MGS, which can be proved to be backward stable for computing the solution x:

Algorithm 8.4.3 Linear Least Squares Solution by MGS.
Carry out MGS on $A \in R^{m \times n}$, 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

$$\begin{aligned}
&\text{for } k = 1, 2, \ldots, n \\
&\quad z_k = q_k^T b^{(k)}; \quad b^{(k+1)} = b^{(k)} - z_k q_k; \\
&\text{end} \\
&\text{solve } Rx = z;
\end{aligned}$$

This algorithm was quite widely used in the 1960's, although at that time the properties of the algorithm was not fully understood.

In some applications it is important to use an algorithm which is backwards stable also for the computed residual $\bar{r}$, i.e. we want a relation

$$(A + E)^T \bar{r} = 0, \quad \|E\|_2 \leq cu\|A\|_2,$$

to hold for some constant c. This implies that $A^T \bar{r} = -E^T \bar{r}$, and

$$\|A^T \bar{r}\|_2 \leq cu\|\bar{r}\|_2. \tag{8.4.12}$$

Note that even if we take $\bar{r} = \mathrm{fl}(b - Ax)$, where x is the *exact least squares solution*, the best we can guarantee is that $\|A^T \bar{r}\|_2 \leq cu\|b\|_2$. This is a much weaker bound in case when $\|\bar{r}\|_2 \ll \|b\|_2$!

To make Algorithm 8.4.3 backward stable for r it suffices to add a loop where the vector $b^{(n+1)}$ is orthogonalized against $q_n, q_{n-1}, \ldots, q$ *in that 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. An explanation of this subtle point is given at the end of Section 8.5.6.

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 8.4.4 Minimum Norm Solution by MGS.
Carry out MGS on $A^T \in R^{m \times n}$, with $\mathrm{rank}(A) = n$ to give $Q_1 = (q_1, \ldots, q_n)$ and R. Then the minimum norm solution $y = y^{(0)}$ is obtained from

$$\begin{aligned}
&R^T(\zeta_1, \ldots, \zeta_n)^T = c; \\
&y^{(n)} = 0; \\
&\text{for } k = n, \ldots, 2, 1 \\
&\quad \omega_k = q_k^T y^{(k)}; \quad y^{(k-1)} = y^{(k)} - (\omega_k - \zeta_k) q_k; \\
&\text{end}
\end{aligned}$$

If the columns of Q_1 were orthogonal to working accuracy, then $\omega_k = 0$, $k = m, \ldots, 1$. Hence ω compensates for the lack of orthogonality to make this algorithm backwards stable!

Sometimes it may be advantageous to carry out a **partial** QR factorization, where only the first $k < n$ columns are orthogonalized. After k steps of MGS we have obtained the partial factorization

$$(A, b) = (Q_k, A^{(k+1)}, b^{(k+1)}) \begin{pmatrix} R_{11} & R_{12} & z_k \\ 0 & I & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

where R_{11} is nonsingular. Then we can decompose the residual as $r = b - Ax = r_1 + r_2$, $r_1 \perp r_2$, where

$$r_1 = Q_k(z_k - R_{11}x_1 - R_{12}x_2), \qquad r_2 = b^{(k+1)} - A^{(k+1)}x_2,$$

where $x = (x_1, x_2)^T$. For any x_2 we can make $r_1 = 0$ by taking

$$x_1 = R_{11}^{-1}(z_k - R_{12}x_2).$$

Hence it only remains to solve the reduced least squares problem

$$\min_{x_2} \|b^{(k+1)} - A^{(k+1)}x_2\|_2.$$

A simple application of this occurs in linear regression (see Example 8.2.2, where the first column of A equals the vector $(1, 1, \ldots, 1)^T$. Taking $k = 1$ in the above partial factorization is equivalent to "subtracting out the means" and leads directly to the more accurate formula given in the example (see also Problem 3).

Review Questions

1. 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?
2. 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

1. Consider the overdetermined linear system $Ax = b$ in Example 8.2.1. 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!

2. Suppose the square root free version of modified Gram–Schmidt is used to compute the factorization $A = \tilde{Q}_1\tilde{R}$. Modify Algorithm 8.4.1 for computing the least squares solution and residual from this factorization.

3. (a) Let R be the unique upper triangular Cholesky factor of A^TA ($\operatorname{diag}(R) > 0$), where $A \in \mathbf{R}^{m\times n}$ and $\operatorname{rank}(A) = n$. Partition R,

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \qquad R_{11} \in \mathbf{R}^{k\times k}.$$

Suppose that after k steps of MGS, we have computed R_{11}, R_{22}, and $A_2^{(k+1)} = (a_{k+1}^{(k+1)}, \ldots, a_n^{(k+1)})$. Show that R_{22} is the upper triangular Cholesky factor of $(A_2^{(k+1)})^TA_2^{(k+1)}$.

(b) Apply this result to the augmented matrix (A, b) to develop a hybrid MGS-Cholesky algorithm for the linear least squares problem.

(c) Use the algorithm in (b) with $k = 1$ to compute the least squares solution to the system in Problem 7.2.1.

8.5 Orthogonal Factorizations

8.5.1 Elementary Orthogonal Matrices

Let the vectors $q_1, \ldots, q_n \in \mathbf{R}^m$ be orthonormal and form the matrix $Q = (q_1, \ldots, q_n) \in \mathbf{R}^{m\times n}$, $m \geq n$. Then the matrix Q is called orthogonal and $Q^TQ = I_n$. If $m = n$ then it follows that $Q^{-1} = Q^T$, and hence also $QQ^T = I_n$. It follows that $\det(Q)^2 = 1$ and hence $|\det(Q)| = 1$.

Orthogonal matrices which are equal to the unit matrix modified by a matrix of rank one are called **elementary orthogonal matrices**. Such matrices are very useful tools for constructing algorithms for solving a variety of problems in linear algebra. They are attractive since multiplication with such an orthogonal matrix will preserve the Euclidean length and hence lead to numerically stable methods. Here we will consider two very important classes of transformations, elementary reflectors and plane rotations.

Given $a \neq 0 \in \mathbf{R}^m$, we consider the problem of constructing an orthogonal matrix $U \in \mathbf{R}^{m\times m}$ of the form

$$U = (y, U_1), \qquad y = a/\|a\|_2. \tag{8.5.1}$$

Multiplying from the left by U^T and using $U^TU = I$ it follows that $y = Ue_1$ satisfies $U^Ty = e_1$ or

$$U^Ta = \sigma e_1, \qquad \sigma = \|a\|_2.$$

Hence (8.5.1) is equivalent to finding an orthogonal matrix U such that *multiplication by U^T zeros all components except the first in a*, In order to construct a matrix U satisfying (8.5.1) we consider matrices of the form

$$H = I - uu^T/\gamma, \qquad \gamma = u^Tu/2. \tag{8.5.2}$$

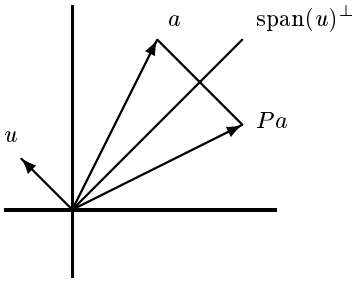


Figure 8.5.1.

By construction H is symmetric $H^T = H$, and using (8.5.2) we have

$$H^T H = H^2 = I - 2uu^T/\gamma + u(u^T u)u^T/\gamma^2 = I.$$

Hence H is orthogonal, and $H^2 = I$. The product Ha where a is a given vector can be computed without explicitly forming H itself using

$$Ha = (I - uu^T/\gamma)a = a - u(u^T a)/\gamma.$$

Note that $Ha \in \text{span}[a, u]$. We have $Hu = -u$, i.e., H reverses u, and $Ha = a$, for $a \perp u$. Hence H has $m-1$ eigenvalues equal to $+1$ and one equal to -1, and thus $\det(H) = -1$. The effect of the transformation Ha for a general vector a is to reflect a in the $(m-1)$ dimensional hyperplane characterized by the normal vector u, see Fig. 8.1.2. Therefore, H is called an **elementary reflector**.

The use of elementary reflectors in numerical linear algebra was initiated by A. S. Householder. Matrices of the form (8.5.2) are therefore often called **Householder reflectors**,, and the vector u a Householder vector. We now show how to construct a Householder reflector H such that $Ha = \mp\sigma e_1$, a slightly more general form than (8.5.1). From Fig. 8.3.1 it is easily seen that this is satisfied if we take

$$u = a \pm \sigma e_1, \qquad \sigma = \|a\|_2. \tag{8.5.3}$$

If we put $\alpha_1 = a^T e_1$, we find

$$\gamma = \tfrac{1}{2}u^T u = \tfrac{1}{2}(a \pm \sigma e_1)^T(a \pm \sigma e_1) = \tfrac{1}{2}(\sigma^2 \pm 2\sigma\alpha_1 + \sigma^2) = \sigma(\sigma \pm \alpha_1).$$

If a is close to a multiple of e_1, then $\sigma \approx |\alpha_1|$ and cancellation may occur in computing γ, leading to a large relative error in γ. To avoid this we take

$$u = a + \text{sign}\,(\alpha_1)\sigma e_1, \qquad \gamma = \sigma(\sigma + |\alpha_1|), \tag{8.5.4}$$

which gives $Ha = -\text{sign}\,(\alpha_1)\sigma e_1 = \hat{\sigma} e_1$. 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 (8.5.4) for γ so that the other choice of sign does not give rise to numerical cancellation, see Parlett [1971, pp. 91].)

The Householder transformation in (8.5.2) 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

$$u = \begin{pmatrix} 1 \\ \hat{u} \end{pmatrix}, \qquad a = \begin{pmatrix} \alpha_1 \\ a_2 \end{pmatrix},$$

then the Householder matrix becomes $H = I - \beta u u^T$, where

$$\hat{u} = \operatorname{sign}(\alpha_1) a_2/\rho, \qquad \beta = 1 + |\alpha_1|/\sigma, \qquad \rho = \sigma\beta. \tag{8.5.5}$$

This has the advantage that we can reconstruct β from $\hat{u}$ using

$$\beta = 2/u^T u = 2/(1 + \hat{u}^T \hat{u}).$$

Algorithm 8.5.1 Let a be a vector with $\|a\|_2 = \sigma$ and $a^T e_1 = \alpha_1$. The following algorithm constructs a Householder transformation $H = I - \beta u u^T$, where $u^T e_1 = 1$, such that $Ha = \hat{\sigma} e_1$, where $\hat{\sigma} = -\operatorname{sign}(\alpha_1)\sigma$.

$$\begin{aligned}
&[u, \beta, \hat{\sigma}] = \text{house}(a) \\
&\quad \alpha_1 = a(1); \\
&\quad \hat{\sigma} = -\operatorname{sign}(\alpha_1)(a^T a)^{1/2}; \\
&\quad \beta = 1 + |\alpha_1/\hat{\sigma}|; \\
&\quad \rho = \hat{\sigma}\beta; \\
&\quad u = [1; a_2/\rho];
\end{aligned}$$

If a matrix $A = (a_1, \ldots, a_n) \in \mathbf{R}^{m \times n}$ is *premultiplied* by H the product can be computed in $2mn$ multiplications as

$$HA = (Ha_1, \ldots, Ha_n), \qquad Ha_j = a_j - \beta(u^T a_j)u. \tag{8.5.6}$$

An analogous formula, exists for *postmultiplying* A with H, where H now acts on the *rows* of A. Writing the products HA and AH as

$$HA = A - \beta u(u^T A), \qquad AH = A - \beta(Au)u^T,$$

shows that in both cases is A altered by a matrix of rank one.

Another useful class of orthogonal transformations are the matrices representing **plane rotations**, which are also called **Givens rotations** after Wallace Givens, who popularized their use for numerical computations. In $\mathbf{R}^2$ the matrix representing a rotation clockwise through an angle θ is

$$G(\theta) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \qquad c = \cos\theta, \quad s = \sin\theta. \tag{8.5.7}$$

Note that $G^{-1}(\theta) = G(-\theta)$, and $\det G(\theta) = +1$.

In $\mathbf{R}^m$ 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_m

$$G_{ij}(\theta) = \begin{array}{c} \\ \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & i & & j & & \end{array} \\ \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & & s & & \\ & & & \ddots & & & \\ & & -s & & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \end{array}. \tag{8.5.8}$$

Premultiplying a vector $a = (\alpha_1, \ldots, \alpha_m)^T$ by $G_{ij}(\theta)$ we get

$$G_{ij}(\theta)a = (\tilde{\alpha}_1, \ldots, \tilde{\alpha}_m)^T, \qquad \tilde{\alpha}_k = \begin{cases} \alpha_k, & k \neq i, j; \\ c\alpha_i + s\alpha_j, & k = i; \\ -s\alpha_i + c\alpha_j, & k = j. \end{cases} \tag{8.5.9}$$

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

$$c = \alpha_i/\sigma, \qquad s = \alpha_j/\sigma, \qquad \sigma = (\alpha_i^2 + \alpha_j^2)^{1/2} \neq 0. \tag{8.5.10}$$

Note that $-G(\theta)$ also zeros $\tilde{\alpha}_j$ so c and s are only determined up to a factor ± 1.

To guard against possible overflow, the Givens rotation should be computed as in the following procedure:

Algorithm 8.5.2 Given $(\alpha, \beta)^T \neq 0$ the algorithm constructs c, s, σ such that $s^2 + c^2 = 1$ and

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix}:$$

$[c, s, \sigma] = \text{givrot}(\alpha, \beta)$
  if $|\alpha| > |\beta|$
    $t = \beta/\alpha; \quad c = 1/\sqrt{1+t^2};$
    $s = tc; \quad \sigma = \alpha/c;$
  else
    $t = \alpha/\beta; \quad s = 1/\sqrt{1+t^2};$
    $c = ts; \quad \sigma = \beta/s;$
  end

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

$$a_{ik} := ca_{ik} + sa_{jk}, \tag{8.5.11}$$
$$a_{jk} := -sa_{ik} + ca_{jk}, \quad k = 1, 2, \ldots, n. \tag{8.5.12}$$

The product requires $4n$ multiplications. 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, which satisfies (8.5.1). Let G_{1k}, $k = 2, \ldots, m$ be a sequence of Givens rotations, where G_{1k} is determined to zero the k-th 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 $G_{k-1,k}$, $k = m, m-1, \ldots, 2$, where $G_{k-1,k}$ is chosen to zero the k-th component. This demonstrates the flexibility of Givens rotations compared to reflectors.

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 G. W. Stewart one stores the number c or s of smallest magnitude. To distinguish between the two cases one stores the reciprocal of c. More precisely, if $c \neq 0$ we store

$$\rho = \begin{cases} s, & \text{if } |s| < |c|; \\ 1/c, & \text{if } |c| \leq |s| \end{cases}.$$

In case $c = 0$ we put $\rho = 1$, a value that cannot appear otherwise.

To reconstruct the Givens rotation, if $\rho = 1$, we take $s = 1$, $c = 0$, and

$$\rho = \begin{cases} s = \rho, & c = \sqrt{1-s^2}, & \text{if } |\rho| < 1; \\ c = 1/\rho, & s = \sqrt{1-c^2}, & \text{if } |\rho| > 1; \end{cases}$$

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 [13, 1996, Sec. 5.1.13].

Sometimes it is useful to consider **hyperbolic rotations** $\breve{G}$ of the form

$$\breve{G} = \begin{pmatrix} ch & -sh \\ -sh & ch \end{pmatrix}, \qquad ch^2 - sh^2 = 1. \tag{8.5.13}$$

Note that we can write $ch = \cosh\theta$, $sh = \sinh\theta$, which explains the name. The matrix $\breve{G}$ is S-orthogonal, $\breve{G}^T S \breve{G} = I$, for the **signature matrix** $S = \operatorname{diag}(1, -1)$.

A hyperbolic rotation can be used to zero a selected component in a vector. Provided that $|\alpha| > |\beta|$ and

$$s = \beta/\alpha, \quad = \sqrt{(1+s)(1-s)}, \quad \sigma = \alpha c,$$

we have $s^2 + c^2 = 1$ and

$$\breve{G}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{c}\begin{pmatrix}1 & -s\\ -s & 1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\sigma\\ 0\end{pmatrix}.$$

To form a product $\breve{G}x$ the straightforward way is not numerically stable. Instead we note the equivalence of

$$\breve{G}\begin{pmatrix}x_i\\ x_j\end{pmatrix} = \begin{pmatrix}\hat{x}_i\\ \hat{x}_j\end{pmatrix}, \qquad G\begin{pmatrix}\hat{x}_i\\ x_j\end{pmatrix} = \begin{pmatrix}c & s\\ -s & c\end{pmatrix}\begin{pmatrix}\hat{x}_i\\ x_j\end{pmatrix} = \begin{pmatrix}x_i\\ \hat{x}_j\end{pmatrix}, \tag{8.5.14}$$

where G is an orthogonal Givens rotation. The mixed method where $\hat{x}_i$ is determined from the hyperbolic rotation and then $\hat{x}_j$ from the equivalent Givens rotation

$$\hat{x}_i = (x_i - s x_j)/c, \qquad \hat{x}_j = -s\hat{x}_i + c x_j, \tag{8.5.15}$$

has been shown to be numerically more stable.

8.5.2 The Full QR Factorization

Methods for solving the linear least squares problem which, like the SVD, are based on orthogonal transformations avoid the squaring of the condition number that results from forming the normal equations. In this section we first develop algorithms using elementary orthogonal transformations to factor a matrix $A \in \mathbf{R}^{m\times n}$ $(m \geq n)$ into the product of a *square* orthogonal matrix $Q \in \mathbf{R}^{m\times m}$ and an upper triangular matrix $R \in \mathbf{R}^{m\times n}$. We then show how to use this **full QR factorization** for solving linear least squares problems.

Theorem 8.5.1. *The Full QR Factorization*

Let $A \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(A) = n$. *Then there is an orthogonal matrix $Q \in \mathbf{R}^{m\times m}$ and an upper triangular matrix R with positive diagonal elements such that*

$$A = Q\begin{pmatrix}R\\ 0\end{pmatrix}. \tag{8.5.16}$$

Proof. A constructive proof will be given in Section 8.5.3. □

Since Q is orthogonal the singular values of R equal those of A and $\kappa(R) = \kappa(A)$. Indeed, to compute the SVD of A one can first compute the QR factorization and then the SVD of R.

The QR factorization can be written

$$A = (Q_1, Q_2)\begin{pmatrix}R\\ 0\end{pmatrix} = Q_1 R. \tag{8.5.17}$$

where Q has been partitioned as $Q = (Q_1, Q_2)$, $Q_1 \in \mathbf{R}^{m\times n}$, $Q_2 \in \mathbf{R}^{m\times(m-n)}$. This is the the factorization computed by the Gram–Schmidt algorithm. From (8.5.17) it follows that the columns of Q_1 and Q_2 form orthonormal bases for the range space of A and its orthogonal complement,

$$\mathcal{R}(A) = \mathcal{R}(Q_1), \qquad \mathcal{N}(A^T) = \mathcal{R}(Q_2), \tag{8.5.18}$$

and the corresponding orthogonal projections are

$$P_{\mathcal{R}(A)} = Q_1 Q_1^T, \qquad P_{\mathcal{N}(A^T)} = Q_2 Q_2^T. \tag{8.5.19}$$

Note that although the matrix Q_1 in (8.5.17) is uniquely determined, Q_2 can be any orthogonal matrix with range $\mathcal{N}(A^T)$.

8.5.3 Householder QR Factorization

In contrast to the Gram–Schmidt algorithm for computing the QR factorization, the methods we now consider represents Q *implicitly* as a product of Householder or Givens matrices; see Section 8.5.1. This elegantly avoids the problem with loss of orthogonality in Q!

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 $A = (a_1, a_2, \ldots, a_n)$, $\sigma_1 = \|a_1\|_2$, and choose $H_1 = I - \beta_1 u_1 u_1^T$, so that

$$H_1 a_1 = H_1 \begin{pmatrix} \alpha_1 \\ \hat{a}_1 \end{pmatrix} = \begin{pmatrix} r_{11} \\ 0 \end{pmatrix}, \qquad r_{11} = -\text{sign}\,(\alpha_1)\sigma_1.$$

By (8.5.4) we achieve this by choosing $\beta_1 = 1 + |\alpha_1|/\sigma_1$,

$$u_1 = \begin{pmatrix} 1 \\ \hat{u_1} \end{pmatrix}, \qquad \hat{u_1} = \text{sign}\,(\alpha_1)\hat{a}_1/\rho_1, \qquad \rho_1 = \sigma_1\beta_1.$$

H_1 is then applied to the remaining columns $a_2, \ldots, a_n$, giving

$$A^{(2)} = H_1 A = \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ 0 & \tilde{a}_{22} & \ldots & \tilde{a}_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{n2} & \ldots & \tilde{a}_{nn} \end{pmatrix}.$$

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, $k = 2, \ldots, n$ are similar to the first. Before the kth step we have computed a matrix of the form

$$A^{(k)} = \begin{matrix} \\ {\scriptstyle k-1} \end{matrix} \overset{\overset{k-1}{}}{\begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & \hat{A}^{(k)} \end{pmatrix}}, \tag{8.5.20}$$

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,

$$A^{(k+1)} = H_k A^{(k)}, \qquad H_k = \begin{pmatrix} I_k & 0 \\ 0 & \tilde{H}_k \end{pmatrix}. \tag{8.5.21}$$

Here $\tilde{H}_k = I - \beta_k u_k u_k^T$ is chosen to zero the elements below the main diagonal in the first column of the submatrix $\hat{A}^{(k)} = (a_k^{(k)}, \ldots, a_n^{(k)}) \in \mathbf{R}^{(m-k+1)\times(n-k+1)}$, i.e., $\tilde{H}_k a_k^{(k)} = r_{kk} e_1$. With $\sigma_k = \|a_k^{(k)}\|_2$, using (8.5.3), we get $r_{kk} = -\text{sign}\,(a_{kk}^{(k)})\sigma_k$, and

$$\hat{u}_k = \text{sign}\,(\alpha_k^{(k)})\hat{a}_k^{(k)}/\rho_k, \qquad \beta_k = 1 + |a_{kk}|/\sigma_k. \tag{8.5.22}$$

where $\rho_k = \sigma_k \beta_k$. After n steps we have obtained the QR factorization of A, where

$$R = R_{11}^{(n+1)}, \qquad Q = H_1 H_2 \cdots H_n. \tag{8.5.23}$$

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. [2]

Algorithm 8.5.3 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:

$$H_k = \text{diag}\,(I_{k-1}, \tilde{H}_k), \qquad \tilde{H}_k = I - \beta_k u_k u_k^T, \quad k = 1, 2, \ldots, n, \tag{8.5.24}$$

so that $Q = H_1 H_2 \cdots H_n$.

```
for k = 1, 2, ..., n
    [u_k, β_k, r_kk] = house(a_k^(k));
    for j = k+1, ..., n
        γ_jk = β_k u_k^T a_j^(k);
        r_kj = a_kj^(k) − γ_jk;
        a_j^(k+1) = â_j^(k) − γ_jk û_k;
    end
end
```

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 $(\beta_1, \ldots, \beta_n)$ of length n can be recomputed from

$$\beta_k = \tfrac{1}{2}(1 + \|\hat{u}_k\|_2^2)^{1/2},$$

[2] 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.

and therefore need not be saved. The algorithm requires $(mn^2 - n^3/3)$ multiplications, or $n^3/3$ less than for the MGS method. Note that in the special case that $m = n$ it would be possible to skip the last step which just computes $\tilde{H}_n = -1$ and $r_{nn} = -a_{nn}^{(n)}$.

8.5.4 The Pivoted QR Factorization

It is often advantageous to use column pivoting in the QR factorization. In this we compute a factorization

$$AP = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{8.5.25}$$

where P is a permutation matrix. The following simple pivoting strategy, first suggested by Golub, has been shown to work well in practice. Assume that after k steps in the Householder Algorithm 7.3.3 we have computed the partial QR factorization

$$A^{(k+1)} = (H_k \cdots H_1)A(\Pi_1 \cdots \Pi_k) = \begin{pmatrix} R_{11}^{(k+1)} & R_{12}^{(k+1)} \\ 0 & \tilde{A}^{(k+1)} \end{pmatrix}, \tag{8.5.26}$$

Then the pivot column in the next step is chosen as a column of largest norm in the submatrix

$$\tilde{A}^{(k+1)} = (\tilde{a}_{k+1}^{(k+1)}, \ldots, \tilde{a}_n^{(k+1)}) \in \mathbf{R}^{(m-k)\times(n-k)},$$

i.e., Π_{k+1} is chosen to interchange columns p and $k+1$, where p is the smallest index such that

$$s_p^{(k+1)} \geq s_j^{(k+1)}, \qquad s_j^{(k+1)} = \|\tilde{a}_j^{(k+1)}\|_2^2, \qquad j = k+1, \ldots, n. \tag{8.5.27}$$

If $s_p^{(k+1)} = 0$ then the algorithm terminates with $\tilde{A}^{(k+1)} = 0$ in (8.5.26). This pivoting strategy can be viewed as choosing a remaining column of largest distance to the subspace spanned by the previously chosen columns. This is equivalent to maximizing the diagonal element $r_{k+1,k+1}$.

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 it is easily shown that in QR factorization with pivoting the elements in R must satisfy the inequalities

$$r_{kk}^2 \geq \sum_{i=k}^{j} r_{ij}^2, \quad j = k+1, \ldots, n. \tag{8.5.28}$$

This implies in particular that the diagonal elements form a non-increasing sequence, $r_{11} \geq r_{22} \geq \cdots \geq r_{nn}$.

8.5.5 Solving Least Squares Problems by Householder QR

We now show how to use the QR factorization to solve the linear least squares problem (8.1.1).

Theorem 8.5.2.

Let the QR factorization of $A \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(A) = n \le m$ be given by (8.5.16). Then the unique solution x to $\min_x ||Ax-b||_2$ and for the corresponding residual vector r are given by

$$x = R^{-1}c_1, \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = Q^T b, \quad r = Q\begin{pmatrix} 0 \\ c_2 \end{pmatrix}, \tag{8.5.29}$$

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 (8.5.29) and using the orthogonality of Q we have

$$b = QQ^T b = Q_1 c_1 + Q_2 c_2 = Ax + r$$

which shows the formula for r. □

By Theorem 8.5.2, when R and $P_1, P_2, \ldots, P_n$ have been computed by Algorithm 8.5.3 the least squares solution x and residual r can be computed from

$$\begin{aligned} n\{ \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} &= P_n \cdots P_2 P_1 b, \qquad Rx = c_1, \\ r &= P_1 \cdots P_{n-1} P_n \begin{pmatrix} 0 \\ c_2 \end{pmatrix}, \end{aligned} \tag{8.5.30}$$

and $\|r\|_2 = \|c_2\|_2$. Note that the matrix Q should not be explicitly formed.

When $\operatorname{rank}(A) = m \le n$, i.e., the matrix A has full row rank, the QR factorization of A^T (which is equivalent to the LQ factorization of A) can be used to solve the minimum norm problem (8.1.16).

Theorem 8.5.3.

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

$$x = Q_1 y_1 + Q_2 y_2, \qquad y_1 = L^{-1}b \tag{8.5.31}$$

where y_2 is arbitrary. The minimum norm solution is obtained by taking $y_2 = 0$,

$$x = Q_1 R^{-T} b. \tag{8.5.32}$$

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 $y_2 = 0$. □

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 (8.5.30) 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 8.5.4.

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

$$A + E = \tilde{Q} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \qquad \|E\|_F \le cu\|A\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm, $c = 6n(m - n/2 + 7)$, and u is the machine precision. 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

$$\|\delta A\|_F \le cun^{1/2}\|A\|_F, \qquad \|\delta b\|_2 \le cu\|b\|_2, \tag{8.5.33}$$

Proof. See Lawson and Hanson [16, pp.83–99]. □

An algorithm similar to Algorithm 8.5.3, 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; see, e.g., Problem 3, where the QR factorization of a Hessenberg matrix is considered.

8.5.6 Solving Augmented Systems

The Householder QR factorization can also be used to solve the generalized least squares problem associated to the

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}, \tag{8.5.34}$$

which represents necessary and sufficient conditions for the solution of the two following problems:

$$\min_r \{\|y\|_2^2 - 2b^T y\} \quad \text{subject to} \quad A^T y = c,$$

$$\min_x \{\|b - Ax\|_2^2 + 2c^T x\}.$$

When $c = 0$ this is equivalent to the least squares problem $\min_x \|b - Ax\|_2^2$ and y is the corresponding residual. When $b = 0$ this is equivalent to the computing the minimum norm solution of $A^T y = c$. Hence, these two problems discussed before are special cases of problem (8.5.34).

Methods based both on Householder and MGS factorization of A for solving problem (8.5.34) have been given and analyzed in the literature. When a Householder QR factorization is available the algorithm is as follows:

$$z = R^{-T} c, \qquad \begin{pmatrix} d \\ f \end{pmatrix} = Q^T b, \qquad r = Q \begin{pmatrix} z \\ f \end{pmatrix}, \qquad x = R^{-1}(d - z).$$

For the MGS factorization $A = Q_1 R$, where $Q_1 = (q_1, \ldots, q_n)$, the corresponding algorithm is as follows:

1. Solve $R^T z = c$ for $z = (\zeta_1, \ldots, \zeta_n)^T$,
2. **for** $k = 1, \ldots, n$ **do** $\{\delta_k := q_k^T b; \quad b := b - q_k \delta_k\}$;
   to give $h := b$, and $d = (\delta_1, \ldots, \delta_n)^T$.
3. **for** $k = n, \ldots, 1$ **do** $\{\omega_k := q_k^T h; \quad h := h - q_k(\omega_k - \zeta_k)\}$
   to give $r := h$,
4. Solve $Rx = d - z$.

Step 3 can be interpreted as a reorthogonalization. Note that this should be done in backward order starting with q_n! For this algorithm an equally satisfactory error analysis has been given.

A key observation for understanding the good numerical properties of the modified Gram–Schmidt algorithm is that it can be interpreted as Householder QR factorization applied to the matrix A *augmented with a square matrix of zero elements on top.* These two algorithms are not only mathematically but also *numerically* equivalent. In the MGS method the columns are transformed by

$$a_j^{(k+1)} = M_k a_j^{(k)}, \qquad M_k = I - q_k q_k^T,$$

where M_k is the orthogonal projection onto the complement of q_k. In the Householder method one computes the factorization

$$P^T \begin{pmatrix} 0 \\ A \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad P^T = P_n \cdots P_2 P_1,$$
$$P_k = I - v_k v_k^T, \qquad v_k = \begin{pmatrix} -e_k \\ q_k \end{pmatrix}.$$

Here $\|v_k\|_2^2 = 2$, and hence P_k is a Householder reflection. Because of the special structure of the augmented matrix the vectors v_k have a special form. Since the first n rows are initially zero, the scalar products of the vector v_k with later columns will only involve q_k, and it can be verified that the quantities r_{kj} and q_k are *numerically* equivalent to the quantities computed in the modified Gram–Schmidt method.

8.5.7 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 the matrix $A^T A$ will also be a banded matrix with only the first $r = w - 1$ superdiagonals nonzero. Since the factor R in the QR factorization equals the unique Cholesky factor of $A^T A$ it will have only w nonzeros in each row.

Even though the final factor R is independent of the row ordering in A, the intermediate fill-in will vary. In order to save storage and operations it is therefore important to first sort the rows of A so that the column indices $f_i, i = 1, 2, \ldots, m$ of the first nonzero element in each row form a nondecreasing sequence, i.e.,

$$i \le k \Rightarrow f_i \le f_k.$$

Such a band matrix is said to be in **standard form**.

For the case when $m \gg n$ an efficient scheme based on Householder transformations can be developed. We first block the rows so that the kth block A_k consists of all rows for which $f_i = k$, $k = 1, \ldots, p \le n$. The algorithm proceeds in steps $k = 1, \ldots, p$. After the first $k - 1$ steps we have reduced the first $k - 1$ blocks by a sequence of Householder transformations to an upper trapezoidal matrix R_{k-1}. In step k we treat the kth block and compute

$$Q_k^T \begin{pmatrix} R_{k-1} \\ A_k \end{pmatrix} = \begin{pmatrix} R_k \\ 0 \end{pmatrix}.$$

where Q_k is a product of Householder transformations and R_k again upper trapezoidal. Note that because of the structure of the block A_k this (and later) steps will not involve the first $k - 1$ rows and columns of R_{k-1}. This algorithm uses about $w(w-1)(m + \frac{3}{2}n)$ multiplications, which is approximately twice as much as for the method of normal equations. It is essential that the Householder transformations are subdivided as outlined above, otherwise intermediate fill-in will occur and the operation count will increase greatly. For a detailed description of this algorithm, see Lawson and Hanson [16, Ch. 11].

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

$$\min \sum_{j=1}^{n} e_j^2, \qquad e_j = w_i \left(f_j - \sum_{i=-k}^{m-1} c_i B_{i,k+1}(\tau_j) \right). \tag{8.5.35}$$

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

$$x_{j-k-1} < \tau_j < x_j, \qquad \forall j = 1, 2, \ldots, m+k. \tag{8.5.36}$$

Example 8.5.1.

The least squares approximation of a discrete set of data by a linear combination of cubic B-splines gives rise to a banded linear least squares problem. Let

$$s(t) = \sum_{j=1}^{n} x_j B_j(t),$$

where $B_j(t), j = 1, 2, \ldots, n$ are the normalized cubic B-splines, and let (y_i, t_i), $i = 1, \ldots, m$ 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 Fig. 8.4.2. Here A consists of blocks A_k^T, $k = 1, \ldots, 7$. In the Fig. 8.4.2 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.

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 $\|r\|_2^2 = \|c_2\|_2^2$.

$$\begin{pmatrix}
\times & \times & \times & \times & & & & \\
1 & \times & \times & \times & + & & & \\
1 & 2 & \times & \times & + & + & & \\
 & 3 & 4 & \times & \times & + & & \\
 & 3 & 4 & 5 & \times & + & & \\
 & & 6 & 7 & 8 & \times & & \\
 & & 6 & 7 & 8 & 9 & & \\
 & & 6 & 7 & 8 & 9 & & \\
 & & & \times & \times & \times & \times & \\
 & & & \times & \times & \times & \times & \\
 & & & & \times & \times & \times & \times \\
 & & & & \times & \times & \times & \times \\
 & & & & \times & \times & \times & \times
\end{pmatrix}$$

Figure 8.5.2. *The matrix A after $k = 3$ steps of reduction.*

It is also possible to perform the QR factorization by treating one row at a time using Givens' rotations. Each step then is equivalent to updating a full triangular matrix formed by columns $f_i(A)$ to $l_i(A)$. Further, if the matrix A is in standard form the first $f_i(A)$ rows of R are already finished at this stage. The reader is encouraged to work through Example 8.5.1 below in order to understand how the algorithm proceeds!

Review Questions

1. Define a Givens transformations $G_{ij}(\phi) \in \mathbf{R}^{n\times n}$. Give a geometrical interpretations for the case $n = 2$.
2. 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. 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?

Problems

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

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. Describe in detail how to compute the QR factorization of a Hessenberg matrix $H \in \mathbf{R}^{n\times n}$ using Givens transformations. For $n = 5$ such a matrix has the form

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

Approximately how many multiplications are needed for general n?

4. If the matrix Q in the QR factorization is explicitly required in the Householder algorithm it can be accumulated by taking $Q^{(1)} = I$, and computing $Q = Q^{(n+1)}$ by

$$Q^{(k+1)} = P_{n-k+1}Q^{(k)}, \qquad k = 1, 2, \ldots, n.$$

Show that if advantage is taken of the property that $P_k = \operatorname{diag}(I_{k-1}, \tilde{P}_k)$ this accumulation requires $2(m^2n - mn^2 + n^3/3)$ multiplications. Show also that similarly we can accumulate

$$Q_1 = Q\begin{pmatrix} I_n \\ 0 \end{pmatrix}, \qquad Q_2 = Q\begin{pmatrix} 0 \\ I_{m-n} \end{pmatrix}$$

separately in $mn^2 - n^3/3$ and $2m^2n - 3mn^2 + n^3$ multiplications, respectively.

5. Let $Q = Q_1 = (q_1, q_2, \ldots, q_n) \in \mathbf{R}^{n\times n}$ be a real orthogonal matrix.

(a) Determine a reflector $P_1 = I - 2v_1v_1^T$, such that $P_1q_1 = e_1 = (1, 0, \ldots, 0)^T$, and show that $P_1Q_1 = Q_2$ has the form

$$Q_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \tilde{Q}_2 & \\ 0 & & & \end{pmatrix},$$

where $\tilde{Q}_2 = (\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_n) \in \mathbf{R}^{(n-1)\times(n-1)}$ 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

$$P_{n-1} \cdots P_2P_1Q = \operatorname{diag}(1, \ldots, 1, \pm 1).$$

6. An orthogonal matrix Q such that $\det(Q) = 1$ is called a rotation matrix. Show that any rotation matrix $Q \in \mathbf{R}^{3\times 3}$ can be written as a product of three Givens rotations

$$Q = G_{23}(\phi)G_{12}(\theta)G_{23}(\psi).$$

The three angles ϕ, θ, and ψ are called the **Euler angles**.

Hint: Consider the QR factorization of Q.

8.6 Rank Deficient and Ill-Posed Problems

8.6.1 Regularization.

In general the most reliable way to determine an approximate pseudoinverse 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 (8.3.33). 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 the truncated SVD (**TSVD**) solution is to consider the **regularized** problem

$$\min_x \ \|Ax - b\|_2^2 + \mu^2 \|Dx\|_2^2, \tag{8.6.1}$$

where $D = \text{diag}\,(d_1, \ldots, d_n) > 0$ is a positive diagonal matrix. The problem (8.6.1), also called a **damped** least squares problem, is equivalent to the least squares problem

$$\min_x \left\| \begin{pmatrix} A \\ \mu D \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2, \tag{8.6.2}$$

where the matrix A has been modified by appending the matrix μD. When $\mu > 0$ this problem is always of full column rank and has a unique solution. (Often d_j is taken to be proportional to the 2-norm of the jth column in A.)

The solution to problem (8.6.1) satisfies the normal equations

$$(A^T A + \mu^2 D^2)x = A^T b.$$

However, from the formulation (8.6.2) it is seen that the solution can also be obtained from the QR factorization

$$\begin{pmatrix} A \\ \mu D \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{8.6.3}$$

which can be computed by some of the algorithms described before. The special structure can be taken advantage of. For example, in the Householder QR factorization the shape of the transformed matrix after $k = 2$ steps is as follows ($m = n = 4$):

$$\begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & + & + \\ & 0 & + & + \\ & & \times & \\ & & & \times \end{pmatrix}$$

Notice that the first two rows of D have filled in, but the remaining rows of D are still not touched. For each step $k = 1, \ldots, n$ there are m elements in the current column to be annihilated. Therefore the operation count for the Householder QR factorization will increase with $n^3/3$ to mn^2 flops. A similar increase in operations occurs in Givens or MGS QR factorizations. If $A = R$ already is in upper triangular

form then the flop count for the reduction is reduced to approximately $n^3/3$ (cf. Problem 1b).

If $D = I$ the singular values of the modified matrix in (8.6.2) are equal to $\tilde{\sigma}_i = (\sigma_i^2 + \mu^2)^{1/2}$, $i = 1, \ldots, n$. In this case the solution can be expressed in terms of the SVD as

$$x(\mu) = \sum_{i=1}^{n} f_i \frac{c_i}{\sigma_i} v_i, \quad f_i = \frac{\sigma_i^2}{\sigma_i^2 + \mu^2}. \tag{8.6.4}$$

The quantities f_i are often called **filter factors**. Notice that as long as $\mu \ll \sigma_i$ we have $f_i \approx 1$, and if $\mu \gg \sigma_i$ then $f_i \ll 1$. This establishes a relation to the truncated SVD solution (8.3.33) which corresponds to a filter factor which is a step function $f_i = 1$ if $\sigma_i > \delta$ and $f_i = 0$ otherwise.

Note that the regularized problem (8.6.1) can be used also when $m < n$ (i.e., when A has fewer rows than columns). However, in this case it may be better to consider the regularized problem

$$\min \left\| \begin{pmatrix} x \\ z \end{pmatrix} \right\|_2^2, \quad \text{subject to} \quad (A \ \mu D) \begin{pmatrix} x \\ z \end{pmatrix} = b. \tag{8.6.5}$$

The solution of this problem can be written $x = A^T y$, $z = (\mu D)^{-1}(b - Ax)$, where y satisfies the system of normal equations

$$(AA^T + \mu^2 D^2) y = b.$$

Using Theorem 8.5.3, a method for solving problem (8.6.5) is obtained which uses the QR factorization of the matrix $(\mu D \ A)^T$, which can be computed in $m^2 n$ operations. Surprisingly, when $D = I$ the two problems (8.6.5) and (8.6.1) are equivalent; see Problem 3.

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

$$\min_{\delta x} \left\| \begin{pmatrix} A \\ \mu I \end{pmatrix} \delta x - \begin{pmatrix} r^{(q)} \\ 0 \end{pmatrix} \right\|_2, \qquad r^{(q)} = b - Ax^{(q)}. \tag{8.6.6}$$

This iteration may be implemented very effectively since only the QR factorization (8.6.3) (with $D = I$) is needed. The convergence of iterated regularization can be expressed in terms of the SVD of A.

$$x^{(q)}(\mu) = \sum_{i=1}^{n} f_i^{(q)} \frac{c_i}{\sigma_i} v_i, \quad f_i^{(q)} = 1 - \left(\frac{\mu^2}{\sigma_i^2 + \mu^2} \right)^q. \tag{8.6.7}$$

Thus for $q = 1$ we have the standard regularized solution and as $q \to \infty$ $x^{(q)} \to A^\dagger b$.

8.6.2 QR Factorization and Rank Deficient Matrices

Although any matrix $A \in R^{m\times n}$ has a QR factorization, the following example shows that this may not always be useful when $\text{rank}\,(A) < n$:

Example 8.6.1.
For any c and s such that $c^2 + s^2 = 1$ we have

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} 0 & s \\ 0 & c \end{pmatrix} = QR.$$

Here $\text{rank}\,(A) = 1 < 2 = n$. Note that the columns of Q no longer provide any information about an orthogonal basis for $R(A)$ and its complement.

We now indicate how the QR factorization should be modified in the rank deficient case.

Theorem 8.6.1.
Given $A \in \mathbf{R}^{m\times n}$ with $\text{rank}\,(A) = r \le \min(m,n)$ there is a permutation matrix Π and an orthogonal matrix $Q =\in \mathbf{R}^{m\times m}$ such that

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \tag{8.6.8}$$

where $R_{11} \in \mathbf{R}^{r\times r}$ is upper triangular with positive diagonal elements.

Proof. Since $\text{rank}\,(A) = r$, we can always choose a permutation matrix Π such that $A\Pi = (A_1, A_2)$, where $A_1 \in \mathbf{R}^{m\times r}$ has linearly independent columns. Then A_1 has a QR factorization and we can write

$$Q^T A\Pi = (\, Q^T A_1 \quad Q^T A_2\,) = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

where R_{11} has positive diagonal elements. From $\text{rank}\,(Q^T A\Pi) = \text{rank}\,(A) = r$ it follows that $R_{22} = 0$, since otherwise $Q^T A\Pi$ would have more than r linearly independent rows. ∎

Note that it is not required that $m \ge n$ in Theorem 8.6.1. The factorization is not unique, since there may be several ways to choose the permutation Π. Pivoting strategies for determining a suitable Π will be discussed later in this section.

The factorization (8.6.8) can be used to solve rank deficient linear least squares problems. 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 the invariance of the l_2-norm problem (8.1.1) becomes

$$\min_x \left\| \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \right\|_2,$$

where x and c have been partitioned conformally. Since R_{11} is nonsingular the first r equations can be satisfied for any x_2 by taking x_1 to be the solution to $R_{11}x_1 = c_1 - R_{12}x_2$. Hence the general least squares solutions can be written

$$x_1 = R_{11}^{-1}(c_1 - R_{12}x_2) = x_b - C_1x_2, \tag{8.6.9}$$

where x_2 is arbitrary and

$$d = R_{11}^{-1}c_1, \qquad C = R_{11}^{-1}R_{12}. \tag{8.6.10}$$

Here C can be computed by solving $n-r$ triangular systems $R_{11}C = R_{12}$, which requires $r^2(n-r)/2$ multiplications.

Taking $x_2 = 0$ we we obtain a particular solution $x_1 = d$ with at most $r = \text{rank}(A)$ nonzero components Any solution x such that Ax only involves at most r columns of A, is called a **basic least squares solution**. Such a solution is appropriate when we want to fit a vector b of observations using *as few columns of* A as possible. It is not unique and depends on the initial column permutation.

We now show how the pseudoinverse solution can be computed using the factorization (8.6.8). Then we want to choose x_2 so that $\|x\|_2$ is minimized. From (8.6.9) it follows that this is achieved by solving the linear least squares problem for x_2

$$\min \left\| \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right\|_2 = \min_{x_2} \left\| \begin{pmatrix} d \\ 0 \end{pmatrix} - \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix} x_2 \right\|_2. \tag{8.6.11}$$

Note that this problem always has a unique solution x_2 and that the pseudoinverse solution $x = A^{\dagger}b$ equals *the residual* of the problem.

To compute x_2 we can form and solve the normal equations

$$(I + CC^T)x_2 = C^T d. \tag{8.6.12}$$

Alternatively we can use Householder QR factorization

$$Q_C^T \begin{pmatrix} C \\ I_{n-r} \end{pmatrix} = \begin{pmatrix} R_C \\ 0 \end{pmatrix}, \qquad Q_C^T \begin{pmatrix} d \\ 0 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix},$$

taking the special structure into account, to obtain x_2 from $R_C x_2 = d_1$.

We remark that since

$$ACA \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix} = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} R_{11}^{-1}R_{12} \\ -In - r \end{pmatrix} = 0,$$

the columns of C span the nullspace of A, i.e. $\mathcal{N}(A) = \mathcal{R}(C)$. By Theorem 8.3.10 the pseudoinverse solution is the unique least squares solution which satisfies $x \perp \mathcal{N}(A)$. Hence it can be obtained by Gram–Schmidt orthogonalization applied to

$$\begin{pmatrix} C & d \\ I_{n-r} & 0 \end{pmatrix}. \tag{8.6.13}$$

8.6.3 Rank Revealing QR Factorization

In the pivoted QR factorization in Section 8.5.3 the pivot column in each step of the reduction was chosen as a column of largest norm in the remaining part. Therefore we have

$$\sigma_1^2(R) = \|R\|_2^2 \le \sum_{i \le j} r_{ij}^2 \le n r_{11}^2,$$

and hence $\sigma_1(R) \le n^{1/2} r_{11}$. More generally, using the interlacing property of singular values (Theorem 8.3.5), a similar argument gives the upper bounds

$$\sigma_k(R) \le (n-k+1)^{1/2} |r_{k,k}|, \quad 1 \le k \le n. \tag{8.6.14}$$

Hence, if after k steps we have $|r_{k,k}| \le (n-k+1)^{-1/2}\delta$ then $\sigma_k(A) = \sigma_k(R) \le \delta$, 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}, $k \le n$. Let R be an upper triangular matrix, whose columns all have unit Euclidean length and whose elements satisfy (8.5.28). The best known *lower* bound for the smallest singular value is

$$\sigma_n \ge 3|r_{nn}|/\sqrt{4^n + 6n - 1} \ge 2^{1-n}|r_{nn}|. \tag{8.6.15}$$

(For a proof see Lawson and Hanson [1974], Ch. 6.)

The lower bound in (8.6.15) 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 8.6.2.

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 (8.6.15), 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 $(n, 1, 2, \ldots, n-1)$ and the rank is revealed from the pivoted QR factorization!

The above example did inspire 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 8.6.2. (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

$$Q^T A\Pi_r = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \tag{8.6.16}$$

with $R_{11} \in \mathbf{R}^{r\times r}$ upper triangular, $c = \sqrt{r(n-r) + \min(r, n-r)})$, and

$$\sigma_{\min}(R_{11}) \geq \frac{1}{c}\sigma_r(A), \qquad \sigma_{\max}(R_{22}) \leq c\sigma_{r+1}(A). \tag{8.6.17}$$

Note that the bounds in this theorem are much better than those in (8.6.15).

From the interlacing properties of singular values (Theorem 8.3.5) it follows by induction that for any factorization of the form (8.6.16) we have the inequalities

$$\sigma_{\min}(R_{11}) \leq \sigma_r(A), \qquad \sigma_{\max}(R_{22}) \geq \sigma_{r+1}(A). \tag{8.6.18}$$

Hence to achieve (8.6.17) 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.

An obvious solution is to try all possible permutations, but the cost of this is prohibitive—it is exponential in the dimension n.

Many other pivoting strategies for computing rank revealing QR factorizations have been proposed. A strategy by T. F. Chan [5] makes use of approximate right singular vectors of A, which can be determined by inverse iteration (see Section 8.6.7). In case $r = n - 1$, the column permutation Π is constructed from an approximation to the right singular vector corresponding to the smallest singular value σ_n.

8.6.4 Stewart's QLP Factorization

Suppose that we have computed the pivoted QR factorization

$$Q^T A\Pi = \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R \in \mathbf{R}^{n\times n}, \tag{8.6.19}$$

of a matrix $A \in \mathbf{R}^{m\times n}$. By postmultiplying the *upper* triangular matrix R by a sequence of Householder transformations we can transform R into a *lower* triangular matrix L,

$$RP = L, \quad L \in \mathbf{R}^{n\times n}, \tag{8.6.20}$$

No pivoting is used in this step. Transposing (8.6.20) shows that this LQ factorization of R is equivalent to a QR factorization of the lower triangular matrix R^T. Combining these two factorizations we obtain

$$Q^T A \Pi P = \begin{pmatrix} L \\ 0 \end{pmatrix}. \tag{8.6.21}$$

This factorization, introduced by G. W. Stewart, is called the QLP decomposition of A. The cost is roughly $mn^2 - n^3/3$ flops for the decomposition (8.6.19) and $2n^3/3$ flops for the decomposition (8.6.20). (Verify that relatively few operations are saved by the zeros in L!)

Despite the simplicity of the QLP factorization the diagonal elements of L usually give very good approximations to all the singular values of A. Indeed, the QLP decomposition thought of as a first step in an iterative algorithm for computing the SVD of A, which rapidly converges to a diagonal matrix containing the singular values; see Section 9.??

Example 8.6.3. Consider the ill-conditioned matrix $K \in \mathbf{R}^{n\times n}$, in Example 8.3.1. in Figure 8.6.3 the singular values σ_k of the matrix K_n, $n = 100$, are displayed in together with: the diagonal elements of R from a pivoted QR factorization (left), the diagonal elements of L in the QLP factorization(right).

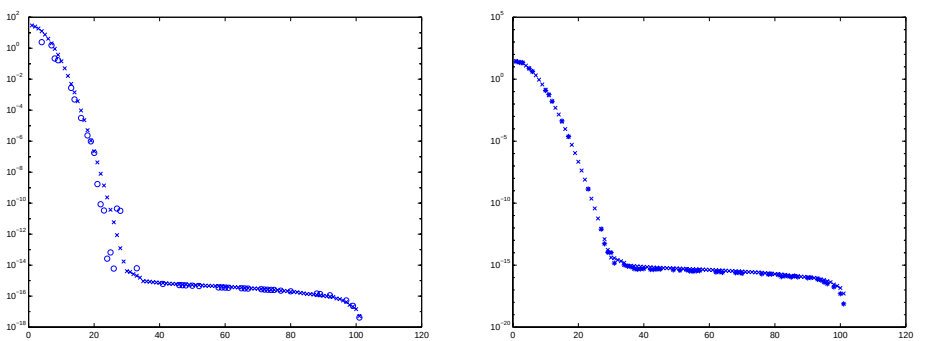


Figure 8.6.1. *Diagonal elements of R in pivoted QR and of of L in the QLP factorization compared with singular values of the matrix K.*

In this example the correct numerical rank is revealed in both factorizations, but the diagonal elements of L in the QLP factorization track the singular values much better and more smoothly.

Suppose that after k steps of the Householder Algorithm 7.3.3 we have computed the partial QR factorization

$$A^{(k+1)} = (H_k \cdots H_1) A (\Pi_1 \cdots \Pi_k) = \begin{pmatrix} R_{11} & R_{12} \\ 0 & \tilde{A}_{22} \end{pmatrix},$$

where $(R_{11} \quad R_{12})$ are the first k rows of R in the QR factorization of A. By postmultiplying with k Householder transformations we obtain

$$(R_{11} \quad R_{12})\, H_1 \cdots H_k = (L_{11} \quad 0),$$

where L_{11} is the first k rows of L in the QLP factorization. Hence, to determine the first k diagonal elements of L, which give the QLP approximations to the first k singular values of A, it is only necessary to perform k steps in each of the two factorizations.

The above observation shows that the two factorizations can be interleaved, i.e. in the kth step we compute the kth row of R and then the kth row of L. This is advantageous when, as in Example 8.6.3, the numerical rank is much less than n. In particular a good estimate of $\sigma_1 = \|A\|_2$ can be obtained in $O(n^2)$ operations by computing the first row $(r_{11} \quad r_{12})$ of R and then

$$\sigma_1 \approx l_{11} = (r_{11}^2 + \|r_{12}\|_2^2)^{1/2}.$$

8.6.5 Complete QR Factorizations

We now consider some factorizations related to the QR factorization, which are useful when A is *numerically rank deficient*.

For some applications it is useful to carry the reduction one step further and compute a so called **complete QR factorization** of A.

Theorem 8.6.3.

Given $A \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(A) = r \le \min(m,n)$. Then there are orthogonal matrices $Q = (Q_1, Q_2) \in \mathbf{R}^{m\times m}$, and $V = (V_1, V_2) \in \mathbf{R}^{n\times n}$ such that

$$A = Q \begin{pmatrix} R & 0 \\ 0 & 0 \end{pmatrix} V^T \tag{8.6.22}$$

where $R \in \mathbf{R}^{r\times r}$ is upper triangular with positive diagonal elements. The pseudoinverse of A is then given by

$$A^\dagger = V \begin{pmatrix} R^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T = V_1 R^{-1} Q_1^T. \tag{8.6.23}$$

Proof. Starting from the factorization in (8.6.8) we can determine a sequence of Householder matrices such that

$$(R_{11} \quad R_{12})\, P_r \cdots P_1 = (R \quad 0).$$

Here P_k, $k = r, r-1, \ldots, 1$, is constructed to zero elements in row k and only affect columns $k, r+1, \ldots, n$. These transformations require $r^2(n-r)$ multiplications. Then (8.6.22) holds with $V = \Pi P_1 \cdots P_r$. 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 (8.1.1). Since the pseudoinverse is uniquely defined by this property, cf. Theorem 8.3.8, the last assertion follows. □

An advantage of the complete QR factorization of A is that V_2 gives an orthogonal basis for the nullspace $\mathcal{N}(A)$. 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 (8.6.23) 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

$$A = Q \begin{pmatrix} R & F \\ 0 & G \end{pmatrix} V^T \qquad A = Q \begin{pmatrix} R^T & 0 \\ F^T & G^T \end{pmatrix} V^T \tag{8.6.24}$$

where R is upper triangular and

$$\sigma_k(R) > \frac{1}{c}, \qquad (\|F\|_F^2 + \|G\|_F^2)^{1/2} \leq c\sigma_{k+1}.$$

An advantage is that unlike the SVD it is possible to efficiently update the factorizations (8.6.24) when rows/columns are added/deleted.

8.6.6 Orthogonal Reduction to Bidiagonal Form

Another important use of orthogonal transformations is to reduce a matrix $A \in \mathbf{R}^{m\times n}$ to upper (or lower) bidiagonal form. This reduction is usually the first step when computing the SVD of A; see Section 9.7. An initial reduction to bidiagonal form is also useful when solving least squares problems with a quadratic constraint; see Section 8.8.2. In the following we assume that $m \geq n$. (This is no restriction since otherwise we can consider A^T.)

The reduction can be performed by a sequence of Householder transformations alternately from left and right. We set $A = A^{(1)}$ and compute $A^{(k+1)} = Q_k A^{(k)} P_{k+1}$, $k = 1, \ldots, n$, where Q_k and P_{k+1} are Householder matrices. After the first step we have

$$A^{(2)} = Q_1 A P_2 = \begin{pmatrix} \beta_1 & \alpha_2 & 0 & \cdots & 0 \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ 0 & \tilde{a}_{32} & \tilde{a}_{33} & \cdots & \tilde{a}_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{m2} & \tilde{a}_{m3} & \cdots & \tilde{a}_{mn} \end{pmatrix}.$$

Here Q_1 is first chosen to zero the elements in the first column of $Q_1 A$ below the main diagonal. Next P_2 is chosen to zero the last $n-2$ elements in the the first row of $Q_1 A$. The first column is obviously not touched by this transformation, which only affects the last $n-1$ columns. All later steps are similar. In the kth step Q_k is chosen to zero the last $m-k$ elements in the kth column of $\tilde{A}^{(k)}$ and then P_{k+1} to zero the last $n-1-k$ elements in the kth row. (If $k \geq n-1$ then P_{k+1} equals

the identity matrix.) After n steps we have the required form

$$Q_B^T A P_B = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad B = \begin{pmatrix} \beta_1 & \alpha_2 & & & \\ & \beta_2 & \alpha_3 & & \\ & & \ddots & \ddots & \\ & & & \beta_{n-1} & \alpha_n \\ & & & & \beta_n \end{pmatrix}, \tag{8.6.25}$$

where

$$Q_B = Q_1 \cdots Q_n \in \mathbf{R}^{m\times m}, \qquad P_B = P_2 \cdots P_{n-1} \in \mathbf{R}^{n\times n}.$$

The algorithm described above requires $2(mn^2 - \frac{1}{3}n^3)$ flops and is due to Golub and Kahan [12]. The Householder vectors associated with Q_B can be stored in the lower triangular part of A and those associated with P_B in the upper triangular part of A. Normally Q_B and P_B are not explicitly required. Note that the singular values of B equal those of A.

When $m \gg n$ it is more efficient to start by computing QR factorization of A and then bidiagonalize the upper triangular matrix R. We could even start from the complete QR factorization

$$A = Q \begin{pmatrix} R & 0 \\ 0 & 0 \end{pmatrix} V^T. \tag{8.6.26}$$

There are two different ways to reduce the upper triangular matrix R to bidiagonal form. Householder transformations can be used as described above. This has the disadvantage that it is not possible to take much advantage of the triangular structure of R. (Verify that already the postmultiplication with P_2 in general will cause the lower triangular part to fill in!) However, operations will be saved if $m \gg n$ since then R is a much smaller matrix than A.h The QR factorization and Householder reduction to bidiagonal form will require $mn^2 - \frac{1}{3}n^3$ and $\frac{4}{3}n^3$ operations, respectively. Hence the modified reduction to bidiagonal form uses $mn^2 + n^3$ flops which is less than the Golub–Kahan bidiagonalization when $m \geq \frac{5}{3}n$.

Using Givens transformation it is possible to take advantage of the triangular form when reducing R to bidiagonal form provided that the elements are zeroed in a suitable order. In the first major step we zero the elements $r_{1n}, \ldots, r_{13}$ in the first row *in this order*. To zero out the element r_{1j} we apply a Givens rotation $G_{j-1,j}$ from the right. This introduces a new non-zero element $r_{j,j-1}$ in the lower triangular part which can be annihilated by a rotation $\tilde{G}_{j-1,j}$ from the left. After this major step the first row and column have the desired form and we continue the reduction on the triangular submatrix in rows and columns $2, \ldots, n$.

Since *two* Givens rotations are needed to zero each of the $(n-1)(n-2)/2$ elements, the operation count turns out to be the same as that for the Householder reduction if standard Givens rotations are used. If the transformations need to be accumulated the Givens reduction will require more work, unless fast Givens transformations are used.

When A is a banded matrix then as seen in Sec. R will be an upper triangular banded matrix. In this case the reduction of R to bidiagonal form of a banded

upper triangular matrix can be accomplished by reducing the bandwidth by one in each major step. Because each zero element introduced generates a new nonzero element that has to be chased across the border of the matrix the reduction is much more expensive than the QR factorization.

Example 8.6.4.

Let $w = 3$ and $n = 7$. The figure below illustrates the steps in zeroing out the element r_{13} using Givens rotations applied alternately from the right and left

$$\begin{pmatrix} \times & \times & \times & & & & \\ & \times & \times & \times & & & \\ & & \times & \times & \times & & \\ & & & \times & \times & \times & \\ & & & & \times & \times & \times \\ & & & & & \times & \times \\ & & & & & & \times \end{pmatrix} \Rightarrow \begin{pmatrix} \times & \times & \otimes & & & & \\ & \times & \times & \times & \oplus & & \\ & \oplus & \times & \times & \times & & \\ & & & \times & \times & \times & \oplus \\ & & & \oplus & \times & \times & \times \\ & & & & & \times & \times \\ & & & & & \oplus & \times \end{pmatrix}.$$

corresponding to the transformations

$$G_{67}G_{45}G_{23}RG_{23}G_{45}G_{67}.$$

Then the elements $r_{13}, \ldots, r_{n-2,n}$ are eliminated in this order. Such "chasing" algorithms are also commonly used in eigenvalue algorithms. Reduction of an upper triangular matrix of bandwidth w to bidiagonal form requires $\approx 4n^2(w-2)$ multiplications.

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

$$\|E\|_F \leq cn^2u\|A\|_F, \tag{8.6.27}$$

and c is a constant of order unity. Moreover if we use the information stored to generate the products $Q_B = Q_1 \cdots Q_n$ and $P_B = P_1 \cdots P_{n-2}$ then the computed matrices are close to the matrices Q and P 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.

As an application we now consider solving a linear least squares problem using a bidiagonal reduction of the matrix $(b\ A)$:

$$Q_B^T\,(b \quad AP_B\,) = \begin{pmatrix} \tilde{B} \\ 0 \end{pmatrix}, \qquad \tilde{B} = \begin{pmatrix} \gamma & \alpha_1 & & & \\ & \beta_1 & \alpha_2 & & \\ & & \ddots & \ddots & \\ & & & \beta_{n-1} & \alpha_n \\ & & & & \beta_n \end{pmatrix}, \tag{8.6.28}$$

The reduced matrix can be written as $\tilde{B} = (\gamma e_1 \quad B\,)$, where e_1 is the first unit vector. The right hand side b is reduced to γe_1 and A is reduced to a *lower bidiagonal matrix* $B \in \mathbf{R}^{(n+1)\times n}$.

Using the invariance of the l_2-norm it follows that

$$\|Ax-b\|_2 = \left\| \begin{pmatrix} b & A \end{pmatrix} \begin{pmatrix} -1 \\ x \end{pmatrix} \right\|_2 = \left\| Q_B^T \begin{pmatrix} b & AP_B \end{pmatrix} \begin{pmatrix} -1 \\ P_B^T x \end{pmatrix} \right\|_2$$
$$= \|By - \gamma e_1\|_2, \quad x = P_B y.$$

The bidiagonal reduction (8.6.28) can terminate prematurely in two ways, namely if either $\alpha_j = 0$, $j \le n$ or if $\beta_j = 0$, $j \le n$. In both these cases the transformed the least squares problem will have the form

$$\min_y \left\| \begin{pmatrix} B_{11} & 0 \\ 0 & A_2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} \gamma e_1 \\ 0 \end{pmatrix} \right\|_2,$$

where B_{11} has full column rank. But this problem decomposes into two independent problems

$$\min_{y_1} \|B_{11}y_1 - \gamma e_1\|_2 \quad \text{and} \quad \min_{y_2} \|A_2 y_2\|_2.$$

Here the first problem has a unique solution y_1. To get a solution y of minimum norm we take $y_2 = 0$. In the two cases mentioned above we have

$$B_{11} = \begin{pmatrix} \alpha_1 & & & \\ \beta_1 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_{j-2} & \alpha_{j-1} \\ & & & \beta_{j-1} \end{pmatrix} \quad \text{and} \quad B_{11} = \begin{pmatrix} \alpha_1 & & & & \\ \beta_1 & \alpha_2 & & & \\ & \ddots & \ddots & & \\ & & \beta_{j-2} & \alpha_{j-1} & \\ & & & \beta_{j-1} & \alpha_j \end{pmatrix},$$

respectively. In the latter case, i.e. when $\beta_j = 0$, $j \le n$ the matrix B_{11} is square and lower triangular. Hence the solution can be computed from $B_{11}y_1 = \gamma e_1$ by forward substitution and has zero residual. In this case the original system $Ax = b$ is consistent.

In case $\alpha_j = 0$, $j \le n$ the matrix B_{11} has the same shape as if the bidiagonalization does not terminate prematurely. Hence it only remains to show how to solve the upper bidiagonal least squares problem $\min_y \|By - \gamma e_1\|_2$.

To do this we note that the QR factorization of B_{11} can be computed by premultiplying B_{11} by a sequence Givens transformations, which are also applied to the right hand side γe_1. In the first step we premultiply rows (1,2) with a Givens rotation G_{12} to zero out the element β_1:

$$\begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix} \begin{pmatrix} \gamma & \alpha_1 & 0 \\ 0 & \beta_1 & \alpha_2 \end{pmatrix} = \begin{pmatrix} \phi_1 & \rho_1 & \theta_2 \\ \bar{\phi}_2 & 0 & \bar{\rho}_2 \end{pmatrix}.$$

(Here and in the following only elements affected by the rotation are shown.) Continuing in this way in step j the rotation $G_{j,j+1}$ is used to zero the element β_j. In steps, $j = 2, \ldots, n-1$, the rows $(j, j+1)$ are transformed

$$\begin{pmatrix} c_j & s_j \\ -s_j & c_j \end{pmatrix} \begin{pmatrix} \bar{\phi}_j & \bar{\rho}_j & 0 \\ 0 & \beta_j & \alpha_{j+1} \end{pmatrix} = \begin{pmatrix} \phi_j & \rho_j & \theta_{j+1} \\ \bar{\phi}_{j+1} & 0 & \bar{\rho}_{j+1} \end{pmatrix}.$$

where

$$\phi_j = c_j\bar{\phi}_j, \qquad \bar{\phi}_{j+1} = -s_j\bar{\phi}_j, \qquad \rho_j = \sqrt{\bar{\rho}_j^2 + \beta_j^2},$$
$$\theta_{j+1} = s_j\alpha_{j+1}, \qquad \bar{\rho}_{n+1} = c_j\alpha_{j+1}.$$

Finally in the last step n we obtain

$$\begin{pmatrix} c_n & s_n \\ -s_n & c_n \end{pmatrix}\begin{pmatrix} \bar{\phi}_n & \bar{\rho}_n \\ 0 & \beta_n \end{pmatrix} = \begin{pmatrix} \phi_n & \rho_n \\ \bar{\phi}_{j+1} & 0 \end{pmatrix}.$$

After n steps we have obtained

$$G_{n,n+1}\cdots G_{23}G_{12}\,(\gamma e_1 \quad B\,) = \begin{pmatrix} d & R \\ \bar{\phi}_{n+1} & 0 \end{pmatrix},$$

where R is an upper bidiagonal matrix. The least squares solution y is then obtained by backsubstitution from the upper bidiagonal linear system $Rx = d$, where

$$Rx = \begin{pmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \rho_3 & \ddots & \\ & & & \ddots & \theta_n \\ & & & & \rho_n \end{pmatrix}, \qquad d = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_n \end{pmatrix},$$

and the norm of the corresponding residual vector equals $|\bar{\phi}_{n+1}|$.

8.6.7 Condition and Error Estimation

Using the above pivoting strategy, 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}, $i = 1, \ldots, n$, it follows that $r_{nn}^{-1}| \le \sigma_n^{-1} = \|R^{-1}\|_2$, provided $r_{nn} \neq 0$. Combining these estimates we obtain the *lower bound*

$$\kappa(A) = \sigma_1/\sigma_n \ge |r_{11}/r_{nn}| \tag{8.6.29}$$

Although this may considerably underestimate $\kappa(A)$, it has proved to give a fairly reliable estimate in practice. Extensive numerical testing has shown that (8.6.29) 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 $w = R^{-1}(R^{-T}u) = (A^TA)^{-1}u$ so this is equivalent to one step of inverse iteration with A^TA, 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.

In Sec. 8.3.4 we gave a componentwise estimate, (8.3.31), for the error in a least squares solution. This estimate has the form

$$\|\delta x\|_\infty \leq \omega(\| \, |B_1|g_1\|_\infty + \| \, |B_2|g_2\|_\infty), \tag{8.6.30}$$

where

$$B_1 = A^\dagger, \quad g_1 = |b| + |A||x|, \quad B_2 = (A^TA)^{-1}, \quad g_2 = |A^T||r|. \tag{8.6.31}$$

Hence it is possible to estimate the two terms in the right hand side in (8.6.30) using Hager's method. As described in Sec. 6.6.5 this only requires that matrix vector products of the form B_iG_ix and $(B_iG_i)^Tx$, $i = 1, 2$, where $G_i = \text{diag}\,(g_i)$ can be efficiently evaluated. For example, if a QR factorization of A is known, we take $A^\dagger = R^{-1}Q^T$, $(A^TA)^{-1} = R^{-1}R^{-T}$.

Review Questions

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?
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?
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. What is meant by a Rank-revealing QR factorization? Does such a factorization always exist?

Problems

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.

2. Show that the normal equations $(A^TA + \mu^2 I)x = A^Tb$ of the regularized problem (8.6.2) are equivalent to

$$(AA^T + \mu^2 I)y = b, \qquad x = A^Ty,$$

which are the normal equations for problem (8.6.5).

3. 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. 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.

6. To minimize $\|x\|_2$ is not always a good way to resolve rank deficiency, and therefore the following generalization of problem (8.6.11) 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 (8.6.10).
(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 $\mathcal{N}(B)$.

5. Let $A \in \mathbf{R}^{m\times n}$ with $\text{rank}(A) = r$. A rank revealing LU factorizations of the form

$$\Pi_1 A \Pi_2 = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \end{pmatrix},$$

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 pseudoinverse solutions $x = A^\dagger b$. Show, using Theorem 8.3.10 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 $r = m$.)

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!

8.7 Modified Least Squares Problems

8.7.1 Applications.

It is often desired to solve a sequence of least squares problems where in each step rows of data in $(A,\ b)$ are added, deleted, or both. This need arises, e.g., in various time-series problems, where data are arriving sequentially. Such modifications are usually referred to as updating when data is added and downdating when data is removed. Applications in signal processing often require near 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.

In the following we will consider the efficient solution of linear least squares problems for the following modifications:

1. Adding (deleting) a row of $(A\ b)$.
2. Deleting (adding) a column of A.

Stability will be a problem, e.g., when the unmodified problem is worse conditioned than the modified problem.

Assume that $A \in \mathbf{R}^{m\times n}$, $m > n$, is nonsingular. When a column is deleted from A, then by the interlacing property (Theorem 8.3.5) 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.

We will see that deleting a column and adding a row are "easy" operations, whereas adding a column and deleting a row are more delicate operations. This is because in the latter cases the modified matrix may become rank deficient. If we are close to a rank deficient case the problem becomes ill-conditioned.

8.7.2 Recursive least squares.

Consider the least squares problem $\min_x \|Ax - b\|_2$, where A has full column rank. The solution x can be obtained from the normal equations $A^TAx = A^Tb$.

Assume that the matrix $C = (A^TA)^{-1}$ is known as well as the solution $x = C^{-1}(A^Tb)$ are known. If the observation $w^Tx = \beta$ is added we have the related least squares problem

$$\min_x \left\| \begin{pmatrix} w^T \\ A \end{pmatrix} x - \begin{pmatrix} \beta \\ b \end{pmatrix} \right\|_2. \tag{8.7.1}$$

The updated solution $\tilde{x}$ satisfies the modified normal equations

$$(A^TA + ww^T)\tilde{x} = A^Tb + \beta w. \tag{8.7.2}$$

Adding a term ww^Tx to both sides of $A^TAx = A^Tb$ and subtracting from (8.7.2) gives

$$(A^TA + ww^T)(\tilde{x} - x) = \rho w, \quad \rho = \beta - w^Tx.$$

Note that ρ is the predicted residual of the new observation.

A straightforward method for computing $\tilde{x}$ is based on updating the matrix C^{-1}. Since the matrix C is the scaled covariance matrix such a method is called a **covariance matrix method**. Using the Sherman–Morrison formula we obtain

$$\tilde{C} = (A^TA + ww^T)^{-1} = C - \frac{1}{1 + u^Tw}uu^T, \quad u = Cw. \tag{8.7.3}$$

Hence

$$\tilde{C}w = u - \frac{1}{1 + u^Tw}u(u^Tw) = \frac{1}{1 + u^Tw}u,$$

and solving for the updated solution gives the following basic formula:

$$\tilde{x} = x + \frac{\rho}{1 + u^Tw}u. \tag{8.7.4}$$

Equations (8.7.3)–(8.7.4) define a **recursive least squares** (RLS) algorithm, which is associated with the famous Kalman filter.[3] The vector $\tilde{u} = \tilde{C}w$ is called the **Kalman gain vector** .

[3] The Kalman filter is an algorithm introduced by R. Kalman in 1960 and used in control theory to continuously update an estimate of the state of a linear system based on imprecise data.

The equations (8.7.3)–(8.7.4) can with slight modifications be used also for *deleting* old observation. To delete (8.7.1) we have the formulas

$$\tilde{C} = C + \frac{1}{1 - w^T u} u u^T, \qquad \tilde{x} = x - \frac{\rho}{1 - w^T u} u, \tag{8.7.5}$$

where it is assumed that $1 - w^T u \neq 0$. In **windowed least squares** adding a new equation is combined with deleting an old equation. In this case it is important to add the new equation *before* deleting the old.

The simplicity and recursive nature of this updating algorithm has made it very popular for many applications. The main disadvantage of the RLS algorithm is its serious sensitivity to roundoff errors. The updating algorithms based on orthogonal transformations developed in the following sections are generally to be preferred.

8.7.3 Modifying matrix factorizations.

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

$$\begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix}, \tag{8.7.6}$$

then the solution to the least squares problem $\min_x \|Ax - b\|_2$ is given by

$$Rx = z, \qquad \|Ax - b\|_2 = \rho. \tag{8.7.7}$$

The upper triangular matrix (8.7.6) 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.

All known updating algorithms for the singular value decomposition (SVD)

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T,$$

where $U \in \mathbf{R}^{m \times m}$ and $V \in \mathbf{R}^{n \times n}$, require $O(mn^2)$ flops, which is the same order as recomputing the SVD from scratch so there is little gain in using them.

8.7.4 Modifying the Full QR Decomposition

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, e.g., a row is added. Therefore we develop methods for updating the factorization

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{8.7.8}$$

where $Q \in \mathbf{R}^{m\times m}$ is stored explicitly. These updating algorithms require $O(m^2)$ multiplications, and are (almost) normwise backward stable.

Assume that we know the complete QR decomposition (8.7.8) of the matrix $A \in \mathbf{R}^{m\times n}$. In case of a general rank one change we want to compute the decomposition

$$\tilde{A} = A + uv^T = \tilde{Q}\begin{pmatrix}\tilde{R}\\ 0\end{pmatrix}, \tag{8.7.9}$$

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 apply the updating procedure to compute the QR decomposition of

$$(A + uv^T, b) = (A, b) + u(v^T, 0).$$

For simplicity we will in the following not include the right hand side in the description of the algorithms for updating and downdating the decomposition.

Deleting a column.

If the columns of A are partitioned in two blocks

$$A_1 = (a_1, \ldots, a_p), \qquad A_2 = (a_{p+1}, \ldots, a_n),$$

the QR decomposition can be written

$$A = (A_1, A_2) = Q\begin{pmatrix}R_{11} & R_{12}\\ 0 & R_{22}\\ 0 & 0\end{pmatrix}.$$

Hence the QR decomposition of the leading block A_1 is trivially obtained by deleting the $n-p$ trailing columns from the decomposition.

Suppose now that we want to compute the QR decomposition of the matrix resulting from deleting the kth column in A,

$$\tilde{A} = (a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n), \quad 1 \le k < n.$$

From the above observation it follows that this decomposition can readily be obtained from the QR decomposition of the permuted matrix

$$AP_L = (a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n, a_k) = Q\begin{pmatrix}RP_L\\ 0\end{pmatrix}, \tag{8.7.10}$$

where P_L is a permutation matrix which performs a *left circular shift* of the columns $a_k, \ldots, a_n$. The matrix RP_L will have the structure

$$RP_L = (r_1, \ldots, r_{k-1}, r_{k+1}, \ldots, r_n, r_k) = \left(\begin{array}{c|c|c} R_{11} & R_{13} & v\\ \hline 0 & w^T & r_{kk}\\ \hline 0 & R_{33} & 0\end{array}\right)$$

where $R_{33} \in \mathbf{R}^{(n-k)\times(n-k)}$ is upper triangular. In case $k = 3$, $n = 5$, we have

$$RP_L = \left(\begin{array}{cc|ccc} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ \hline 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & \times & 0 \end{array}\right).$$

Hence it is possible to determine a sequence of Givens rotations $G_k = R_{k,k+1}$ so that

$$G_{n-1}^T \cdots G_k^T \begin{pmatrix} w^T & r_{kk} \\ R_{33} & 0 \end{pmatrix} = \tilde{R}_{22}$$

is upper triangular. Note that the last column will fill in. With

$$\tilde{R} = \begin{pmatrix} R_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{pmatrix}, \quad \tilde{R}_{12} = (R_{13}, v),$$

we have the updated decomposition

$$\tilde{A} = AP_L = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \qquad \tilde{Q} = QG_k \cdots G_{n-1}.$$

The required QR decomposition of $\tilde{A}$ is then obtained by deleting the last column in $\tilde{A}$ and $\tilde{R}$.

Appending a column.

We now consider the problem of computing the QR decomposition of a matrix where the column a_{n+1} has been appended in the kth position. We write

$$(a_1, \ldots, a_{k-1}, a_{n+1}, a_k, \ldots, a_n) = (A, a_{n+1})P_R = \tilde{A}P_R, \tag{8.7.11}$$

where P_R is a permutation matrix which performs a *right circular shift* on the columns $a_k, \ldots, a_{n+1}$.

We first compute the QR decomposition of $\tilde{A} = (A, a_{n+1})$ from that of A. This is straightforward using, e.g., Householder's method, since this algorithm can be used to process A a column at a time. We form the vector w,

$$w = Q^T a_{n+1} = \begin{pmatrix} u \\ v \end{pmatrix} \begin{matrix} \} & n \\ \} & m-n \end{matrix},$$

and construct a Householder transformation P_n such that

$$P_n^T \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u \\ \gamma \\ 0 \end{pmatrix}, \qquad \gamma = \|v\|_2. \tag{8.7.12}$$

This gives the decomposition

$$(A, a_{n+1}) = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \quad \tilde{Q} = QP_n, \quad \tilde{R} = \begin{pmatrix} R & u \\ 0 & \gamma \end{pmatrix}.$$

The matrix $\tilde{R}P_R$ now has the structure

$$\tilde{R}P_R = (r_1, \ldots, r_{k-1}, r_{n+1}, r_k, \ldots, r_n) == \left(\begin{array}{c|c|c} R_{11} & u_1 & R_{12} \\ \hline 0 & u_2 & R_{22} \\ 0 & \gamma & 0 \end{array}\right),$$

where $R_{11} \in \mathbf{R}^{(k-1)\times(k-1)}$ and $R_{22} \in \mathbf{R}^{(n-k)\times(n-k)}$ are upper triangular, e.g.,

$$\tilde{R}P_R = \left(\begin{array}{cc|c|cc} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ \hline 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & 0 & \times \\ 0 & 0 & \times & 0 & 0 \end{array}\right),$$

($k = 3$, $n = 4$). We determine Givens rotations $J_i = G_{i-1,i}$, $i = n, \ldots, k$, to zero the last $n - k + 1$ elements in the kth column of $\tilde{R}P_R$. Then

$$J_k^T \cdots J_{n-1}^T \begin{pmatrix} u_2 & R_{22} \\ \gamma & 0 \end{pmatrix} = \begin{pmatrix} \tilde{R}_{22} \end{pmatrix}$$

is upper triangular and the updated R-factor is given by

$$\bar{R} = \begin{pmatrix} R_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{pmatrix}, \qquad \tilde{R}_{12} = (u_1, R_{12}). \tag{8.7.13}$$

The QR decomposition of $\tilde{A}$ in (8.7.11) becomes

$$\tilde{A} = \bar{Q} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \qquad \bar{Q} = \tilde{Q} J_{n-1} \cdots J_k.$$

It is important to note that when a column is *deleted* the new R-factor can be computed without Q being available. However, when a column is *added* it is essential that Q be known in the above algorithm.

Appending a row.

Given the QR decomposition (8.7.8) of a matrix $A \in \mathbf{R}^{m\times n}$ we first consider the problem of computing the QR decomposition of

$$\tilde{A} = \begin{pmatrix} w^T \\ A \end{pmatrix}, \quad w \in \mathbf{R}^n \tag{8.7.14}$$

where a row w^T has been appended. We have

$$\begin{pmatrix} 1 & 0 \\ 0 & Q^T \end{pmatrix} \begin{pmatrix} w^T \\ A \end{pmatrix} = \begin{pmatrix} w^T \\ R \end{pmatrix}$$

To update R we apply n Givens rotations $J_k = G_{k,n+1}$, $k = 1, \ldots, n$, the kth of which annihilates the kth element in the last row to get

$$J_n \cdots J_1 \begin{pmatrix} w^T \\ R \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}.$$

The updated Q-factor then becomes

$$\tilde{Q} = \text{diag}(Q, 1)\Pi_{n+1,m+1} J_1^T \cdots J_n^T.$$

Note that R can be updated without Q being available. Also from the interlacing property (Theorem 1.4.2) it follows that the smallest singular value of R will increase, and hence this procedure is stable. This scheme requires $2n^2 + O(n)$ multiplications if standard Givens rotations are used.

Deleting a row.

We now consider modifying the QR decomposition when a row is *deleted*, which is the **downdating** problem. This corresponds to the problem of deleting the effects of an observation in a least squares problem. There is no loss of generality in assuming that the *first* row of A is to be deleted. To delete the kth row we merely apply the algorithm below with A and Q replaced by $\Pi_{1,k}A$ and $\Pi_{1,k}Q$, where $\Pi_{1,k}$ is a permutation matrix interchanging rows 1 and k.

We wish to obtain the QR decomposition of the matrix $\tilde{A} \in \mathbf{R}^{(m-1)\times n}$ when

$$A = \begin{pmatrix} a_1^T \\ \tilde{A} \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \tag{8.7.15}$$

is known. We now show that this is equivalent to finding the QR decomposition of (e_1, A), where a dummy column $e_1 = (1, 0, \ldots, 0)^T$ 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 column* of Q^T. Now Givens rotations $J_k = G_{k,k+1}$, $k = m-1, \ldots, 1$, can be determined so that

$$J_1^T \cdots J_{m-1}^T q = \alpha e_1, \quad \alpha = \pm 1, \tag{8.7.16}$$

and then

$$J_1^T \cdots J_{m-1}^T \begin{pmatrix} q_1 & R \\ q_2 & 0 \end{pmatrix} = \begin{pmatrix} \alpha & v^T \\ 0 & \tilde{R} \\ 0 & 0 \end{pmatrix}, \tag{8.7.17}$$

where the matrix $\tilde{R}$ is upper triangular. Note that the transformations $J_{n+1}, \ldots, J_{m-1}$ will not affect R. Further, if we compute $\bar{Q} = QJ_{m-1} \cdots J_1$, it follows from (8.7.16) 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 (8.7.15),

$$\begin{pmatrix} a_1^T \\ \tilde{A} \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \tilde{Q} \end{pmatrix} \begin{pmatrix} v^T \\ \tilde{R} \\ 0 \end{pmatrix},$$

and hence the desired decomposition is

$$\tilde{A} = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}.$$

8.7.5 Modifying the Gram–Schmidt decomposition.

In many applications, especially if $m \gg n$, it is too costly to save and modify the full QR decomposition. When we use the compact QR decomposition

$$A = Q_1 R, \quad Q_1 \in \mathbf{R}^{m \times n}, \tag{8.7.18}$$

where $Q_1 \in \mathbf{R}^{m \times n}$ consists of the first n columns of Q. Stable updating algorithms for this decomposition only require $O(mn)$ storage and operations.

The algorithms given in Section 3.2.3 (and 3.2.5) for updating the full QR decomposition when deleting a column or adding a row apply with trivial modifications to the GS decomposition as well. Adding a column is also straightforward, using the columnwise GS algorithm with reorthogonalization; see Section 8.4.2.

We now consider the more difficult problem of deleting a row. Assume that we have the QR factorization

$$A = \begin{pmatrix} a_1^T \\ \hat{A} \end{pmatrix} = Q_1 R = \begin{pmatrix} q_1^T \\ \hat{Q}_1 \end{pmatrix} R, \tag{8.7.19}$$

and want to delete the first row $z^T = a_1^T$. Appending a dummy column (8.7.19) can be written

$$\begin{pmatrix} z^T \\ \tilde{A} \end{pmatrix} = \begin{pmatrix} q_1^T & 1 \\ \hat{Q}_1 & 0 \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}. \tag{8.7.20}$$

Following Daniel et al. [6, 1976] we first apply the Gram–Schmidt algorithm (with reorthogonalization) so that the appended column $e_1 = (1, 0, \ldots, 0)^T$ is orthogonalized to

$$Q_1 = \begin{pmatrix} q_1^T \\ \hat{Q}_1 \end{pmatrix} \in \mathbf{R}^{m \times n}.$$

Because of the special form of the appended column, the result has the form

$$\begin{pmatrix} q_1^T & 1 \\ \hat{Q}_1 & 0 \end{pmatrix} = \begin{pmatrix} q_1^T & \gamma \\ \hat{Q}_1 & h \end{pmatrix} \begin{pmatrix} I & q_1 \\ 0 & \gamma \end{pmatrix} \tag{8.7.21}$$

where $1 = q_1^T q_1 + \gamma^2$ and $\gamma h = -\hat{Q}_1 q_1$. From (8.7.20) we have

$$\begin{pmatrix} z^T \\ \tilde{A} \end{pmatrix} = \begin{pmatrix} q_1^T & \gamma \\ \hat{Q}_1 & h \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix},$$

and we now apply a sequence of plane rotations J_k, $k = n, n-1, \ldots, 1$, in the plane $(k, n+1)$ such that

$$\begin{pmatrix} q_1^T & \gamma \\ \hat{Q}_1 & h \end{pmatrix} U = \begin{pmatrix} 0 & \tau \\ \tilde{Q}_1 & \tilde{h} \end{pmatrix}, \qquad U = J_n \cdots J_1, \tag{8.7.22}$$

where J_k is chosen to annihilate the kth component in $(q_1^T\ \gamma)$. Since these orthogonal transformations preserve the length of the rows we can make $\tau = +1$. Then because the transformed matrix has orthonormal columns it follows that $\tilde{h} = 0$. Thus we have

$$\begin{pmatrix} z^T \\ \tilde{A} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \tilde{Q}_1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{R} \\ z^T \end{pmatrix},$$

where

$$U^T \begin{pmatrix} R \\ 0 \end{pmatrix} = J_1^T \cdots J_n^T \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} \tag{8.7.23}$$

with $\tilde{R}$ upper triangular. Hence the downdated QR decomposition becomes

$$\tilde{A} = \tilde{Q}_1 \tilde{R}. \tag{8.7.24}$$

As mentioned, it is necessary to perform reorthogonalization when the Gram–Schmidt algorithm is used to orthogonalize e_1 to Q_1 in (8.7.20). Let $v = e_1 - Q_1 q_1$. If $\|v\|_2 \geq 1/\sqrt{2}$ then take $v := v/\|v\|_2$; else reorthogonalize

$$s := Q_1^T v, \qquad v' := v - Q_1 s.$$

If $\|v'\|_2 \geq \|v\|_2/\sqrt{2}$, then take

$$v := v'/\|v'\|_2, \qquad \begin{pmatrix} \gamma \\ h \end{pmatrix} := v.$$

Otherwise e_1 is (numerically) linearly dependent on the columns of Q_1. Then we take $\gamma := 0$ and determine $h \in \mathbf{R}^{(n-1)}$ orthogonal to $\hat{Q}_1$; see Daniel et al. [6, 1976].

With one reorthogonalization, the GS downdating algorithm requires about $7mn + 2.5n^2$ flops. Note that the data matrix A is never needed; to delete the first row of A, only the R-factor and the corresponding row in Q_1 are needed. Thus, the storage requirement is about $mn + 0.5n^2$ for Q_1 and R.

8.7.6 Downdating the Cholesky Factorization

Suppose that the factor $R \in \mathbf{R}^{n\times n}$ is given from a Cholesky factorization of A^TA (or QR decomposition of A), where

$$A = \begin{pmatrix} z^T \\ \tilde{A} \end{pmatrix}.$$

In the Cholesky downdating problem we want to find the Cholesky factor of $\tilde{R}$ such that

$$\tilde{R}^T\tilde{R} = \tilde{A}^T\tilde{A} = R^TR - zz^T.$$

This problem is analytically the same as that considered before, except that Q is not used. In this formulation the downdating problem is inherently more ill-conditioned than if Q is also available since there may not be sufficient information stored in R about $\tilde{R}$. The best we can hope for is that the downdating method is backward

stable in the sense that it computes the *exact* Cholesky factor of $(R+E)^T(R+E) - (z+e)(z+e)^T$ with $\|E\|$ and $\|e\|$ small. It is important to note that this does not guarantee that we obtain an $\tilde{R}$ that is the exact Cholesky factor of $(\tilde{A}+E)^T(\tilde{A}+E)$ for some small $\|E\|$!

In the downdating algorithm the first row of Q played an essential role. If Q is not available the following algorithm due to M. Saunders can be used. (This is often referred to as the LINPACK algorithm.)

The first row of the QR decomposition (8.7.15) can be written

$$a_1^T = q^T \begin{pmatrix} R \\ 0 \end{pmatrix} = (q_1^T, q_2^T) \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where q has been partitioned conformably with the R-factor. Hence we can obtain q_1 by solving $R^T q_1 = a_1$, and then, since q is of unit length,

$$\gamma = \|q_2\|_2 = (1 - \|q_1\|_2^2)^{1/2}. \tag{8.7.25}$$

The transformations $J_{n+1}, \ldots, J_{m-1}$ in (8.7.16) will only have the effect of computing

$$J_{n+1}^T \cdots J_{m-1}^T \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} q_1 \\ \pm\gamma \\ 0 \end{pmatrix}$$

and, as remarked above, will not affect R. Thus, we may determine the Givens transformations J_k, $k = 1, \ldots, n$, by

$$J_1^T \cdots J_n^T \begin{pmatrix} q_1 \\ \gamma \end{pmatrix} = \alpha e_1, \quad \alpha = \pm 1,$$

and obtain the updated factor $\tilde{R}$ as in (8.7.17).

We note that if $\gamma \approx u^{1/2}$, where u is the unit roundoff, then γ cannot be computed stably from (8.7.25) because of severe cancellation in the subtraction. Therefore, this algorithm will not be as stable as that using information from Q. The possible failure of the Saunders algorithm can be illustrated by the following simple example.

Example 8.7.1. Consider the least squares problem $\min \|Ax - b\|_2$, where

$$A = \begin{pmatrix} \tau \\ 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \tau = 1/\sqrt{u},$$

and u is the unit roundoff. We may think of the first row of A as an outlier. The QR decomposition of A, correctly rounded to single precision, is

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

where $\epsilon = 1/\tau$. The Saunders algorithm will compute

$$q_1 = \tau/\tau = 1, \quad \gamma^2 = 1 - 1 = 0, \quad J_1 = I.$$

Hence it gives the downdated factor $\tilde{R} = 0$, and the downdated least squares solution is not defined. It is easily verified that if we downdate using Q we get the correct result $\tilde{R} = 1$ and the downdated solution $x = 1$. Since the information from the second row in A is not present in R, only in Q, and therefore *no method working only from R can hope to do better*. However, for the case when A is available a more accurate method for adding a column (or deleting a row) can be developed based on the corrected seminormal

8.8 Some Generalized Problems

8.8.1 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.

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

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad Bx = d. \tag{8.8.1}$$

A solution to problem (8.8.1) exists if and only if the linear system $Bx = d$ is consistent. If $\text{rank}\,(B) = p$ then B has linearly independent rows, and $Bx = d$ is consistent for any right hand side d. A solution to problem (8.8.1) is unique if and only if the null spaces of A and B intersect only trivially, i.e., if $\mathcal{N}(A)\cap\mathcal{N}(B) = \{0\}$, or equivalently

$$\text{rank}\begin{pmatrix} A \\ B \end{pmatrix} = n. \tag{8.8.2}$$

If (8.8.2) is not satisfied then there is a vector $z \neq 0$ such that $Az = Bz = 0$. Hence if x solves (8.8.1) then $x + z$ is a different solution. In the following we therefore assume that $\text{rank}\,(B) = p$ and that (8.8.2) 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

$$\min_{x\in S} \|Ax - b\|_2, \qquad S = \{x \mid \|Bx - d\|_2 = \min\ \}. \tag{8.8.3}$$

The most natural 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 (8.8.3) we will compute a QR factorization of B. By Theorem 8.6.1 (see next section) there is an orthogonal

matrix $Q_B \in \mathbf{R}^{p\times p}$ and a permutation matrix Π_B such that

$$Q_B^T B \Pi_B = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}, \qquad R_{11} \in \mathbf{R}^{r\times r}, \tag{8.8.4}$$

where $r = \text{rank}(B) \le p$ and R_{11} is upper triangular and nonsingular. Using this factorization, and setting $\bar{x} = \Pi_B^T x$, the constraints become

$$(R_{11}, R_{12})\bar{x} = R_{11}\bar{x}_1 + R_{12}\bar{x}_2 = \bar{d}_1, \qquad \bar{d} = Q_B^T d = \begin{pmatrix} \bar{d}_1 \\ \bar{d}_2 \end{pmatrix}, \tag{8.8.5}$$

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 (8.8.4), $\bar{A}\Pi_B = (A_1, A_2)$. then $Ax - b = A_1\bar{x}_1 + A_2\bar{x}_2 - b$. Solving (8.8.5) 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

$$\begin{aligned} &\min_{\bar{x}_2} \|\hat{A}_2\bar{x}_2 - \hat{b}\|_2, \qquad \hat{A}_2 \in \mathbf{R}^{m\times(n-r)} \\ &\hat{A}_2 = \bar{A}_2 - \bar{A}_1 R_{11}^{-1} R_{12}, \qquad \hat{b} = b - \bar{A}_1 R_{11}^{-1}\bar{d}_1. \end{aligned} \tag{8.8.6}$$

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 (8.8.2) is satisfied, then $\text{rank}\,(A_2) = r$. Hence the unconstrained problem has a unique solution, which can be computed from the QR factorization of $\hat{A}_2$.

In the null-space method, assuming that $\text{rank}\,(B) = p$, we compute the QR factorization

$$B^T = Q_B \begin{pmatrix} R_B \\ 0 \end{pmatrix}, \qquad R_B \in \mathbf{R}^{p\times p}, \tag{8.8.7}$$

where R_B is upper triangular and nonsingular. Using Theorem 8.5.3 we find that the general solution of the system $Bx = d$ can be written as

$$x = x_1 + Q_2 y_2, \qquad x_1 = B^\dagger d = Q_1 R_B^{-T} d. \tag{8.8.8}$$

where $Q_B = (Q_1, Q_2)$, $Q_1 \in \mathbf{R}^{n\times p}$, and $Q_2 \in \mathbf{R}^{n\times(n-p)}$. (Note that Q_2 gives an orthogonal basis for the null space of B.) Hence, $Ax - b = Ax_1 + AQ_2y_2 - b$, $y_2 \in \mathbf{R}^{n-p}$, and it remains to solve the unconstrained least squares problem

$$\min_{y_2} \|(AQ_2)y_2 - (b - Ax_1)\|_2. \tag{8.8.9}$$

Let $y_2 = (AQ_2)^I(b - Ax_1)$ be the minimum length solution to (8.8.9), and let x be defined by (8.8.8). Then since $x_1 \perp Q_2y_2$ it follows that

$$\|x\|_2^2 = \|x_1\|_2^2 + \|Q_2y_2\|_2^2 = \|x_1\|_2^2 + \|y_2\|_2^2$$

and x is the minimum norm solution to problem LSE.

If (8.8.2) is satisfied it follows that rank $(AQ_2) = n-p$. Then we can compute the QR factorization

$$Q_A^T(AQ_2) = \begin{pmatrix} R_A \\ 0 \end{pmatrix},$$

where R_A is upper triangular and nonsingular. The unique solution to (8.8.9) can then be computed from

$$R_A y_2 = c_1, \qquad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = Q_A^T(b - Ax_1), \tag{8.8.10}$$

and we finally obtain $x = x_1 + Q_2 y_2$, the unique solution to problem LSE.

The representation in (8.8.8) of the solution x has been used as a basis for a perturbation theory by Leringe and Wedin [1970]. The corresponding bounds for problem LSE are too complicated to be given here. However, it follows that the problem LSE is well conditioned if $\kappa(B)$ and $\kappa(AQ_2)$ are both small. It is important to note that these two condition numbers can be small even when $\kappa(A)$ is large. Any method which starts with minimizing $\|Ax - b\|_2$ will give bad results in such a case.

The method of direct elimination and the null space method both have good numerical stability. In a numerical comparison by Leringe and Wedin [1970] they gave almost identical results. The operation count for the method of direct elimination is slightly lower because Gaussian elimination is used to derive the reduced unconstrained problem.

8.8.2 Quadratic Inequality Constraints

Least squares problems with quadratic constraints arise in a variety of applications, such as smoothing of noisy data, and in trust region methods for nonlinear least squares problems; see Chapter 11.3.4. Another important source is the solution of discretized ill-posed problems. Such problems are often formulated as a least squares problem

$$\min_x \|Ax - b\|_2, \tag{8.8.11}$$

where the singular values of $A \in \mathbf{R}^{m\times n}$ decay exponentially to zero. Hence A will not have a well-defined numerical rank. Therefore, any attempt to solve (8.8.11) without restricting x will usually give a meaningless result.

Example 8.8.1. Consider the linear system in Example 8.8.1, $Kf = g$, where $K \in \mathbf{R}^{n\times n}$, and $f, g \in \mathbf{R}^n$. This was derived by discretizing an integral equation of the first kind. on a uniform mesh on [-1,1] using the trapezoidal rule For $n = 100$ the singular values σ_k of K are are close to roundoff level for $k > 30$. The linear system $Kf = g$ is too ill-conditioned to be solved without more information about the solution f.

Suppose we know that the solution to be $f(s)$ has Euclidian norm equal to 1. Then it is better to formulate the problem as the constrained least squares problem

$$\min_x \|Kf - g\|_2, \quad \text{subject to} \quad \|f\|_2 = 1,$$

which can be solved using techniques now to be described.

One of the most successful methods for solving ill-conditioned problems of this type is **Tikhonov regularization**; see Section 8.6.1. In this method the solution space is restricted by imposing an a priori bound on $\|Lx\|_2$ for a suitably chosen matrix $L \in \mathbf{R}^{p\times n}$. A particularly simple but important case is when $L = I_n$, which is the **standard form** of LSQI. More generally L is often taken to be a discrete approximation to some derivative operator.

The above approach leads us to take f as the solution to a least squares problem with a quadratic inequality constraint (LSQI problem)

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad \|Lx\|_2 \le \gamma. \tag{8.8.12}$$

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.

To find conditions for the existence and uniqueness of solutions to the problem LSQI, first notice that the constraint in problem LSQI is binding only if

$$\min_{x\in S} \|Lx\|_2 > \gamma, \qquad S = \{x \in \mathbf{R}^n \| \ \|Ax - b\|_2 = \min\}. \tag{8.8.13}$$

This observation gives rise to the following theorem.

Theorem 8.8.1. *Assume that problem LSQI has a solution. Then either the unconstrained least squares solution is a solution or or the solution occurs on the boundary of the constraint region. In the latter case the solution $x = x(\lambda)$ satisfies the generalized normal equations*

$$(A^TA + \lambda L^TL)x(\lambda) = A^Tb, \tag{8.8.14}$$

where λ is determined by the secular equation

$$f(\lambda) = \|Lx(\lambda)\|_2 = \gamma. \tag{8.8.15}$$

Proof. Using the method of Lagrange multipliers we minimize

$$\psi(x, \lambda) = \|Ax - b\|_2^2 + \lambda\big(\|Lx\|_2^2 - \gamma^2\big).$$

A necessary condition for a minimum is that the gradient of $\psi(x)$ equals zero, which gives (8.8.14). □

In the following we assume that the constraint is binding. We also assume that the nullspaces of A and L intersect only trivially, i.e., $\mathcal{N}(A) \cap \mathcal{N}(L) = \{0\}$. This condition is equivalent to

$$\operatorname{rank}\begin{pmatrix} A \\ L \end{pmatrix} = n. \tag{8.8.16}$$

It then follows that $f(\lambda)$ is convex and strictly decreasing, and hence, there is a unique positive solution $\lambda^* > 0$ to the secular equation.

Note that (8.8.14) are the normal equations for the least squares problem

$$\min_x \left\| \begin{pmatrix} A \\ \mu L \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2, \quad \mu = \lambda^{1/2}. \tag{8.8.17}$$

Thus, $x(\lambda)$ can be computed for a given value of λ from the QR factorization

$$\begin{pmatrix} A \\ \mu L \end{pmatrix} = Q_\mu \begin{pmatrix} R_\mu \\ 0 \end{pmatrix}, \tag{8.8.18}$$

and it is not necessary to form the cross-product matrices A^TA and L^TL.

A numerical method for solving problem LSQI can be based on applying Newton's method to the secular equation (8.8.15). Differentiating the function

$$f^2(\lambda) = x(\lambda)^T L^T L x(\lambda)$$

we obtain $2f(\lambda)f'(\lambda) = 2x(\lambda)^T L^T L x'(\lambda)$, i.e.,

$$f'(\lambda) = \frac{x(\lambda)^T L^T L x'(\lambda)}{f(\lambda)}.$$

Here $x'(\lambda)$ is obtained by differentiating (8.8.14) giving

$$(A^TA + \lambda L^TL)x'(\lambda) = -L^TLx(\lambda). \tag{8.8.19}$$

Note that $x'(\lambda)$ can be evaluated with little extra work since it does not require any new matrix factorization. (See Problem 2.) Newton's method becomes

$$\lambda_{i+1} = \lambda_i - \frac{f(\lambda_i) - \gamma}{f'(\lambda_i)}. \tag{8.8.20}$$

It can be shown that $f(\lambda)$ is convex for $\lambda > 0$. Therefore, if started with $\lambda_0 \in [0, \lambda^*]$, the iteration (8.8.20) produces a strictly increasing sequence λ_i, $i = 1, 2, \ldots$, converging to λ^*.

Unfortunately, when A is ill-conditioned and γ is small convergence can be very slow. It is preferable to apply Newton's method the equation

$$\frac{1}{f(\lambda)} = \frac{1}{\gamma}, \tag{8.8.21}$$

which gives the iteration

$$\lambda_{i+1} = \lambda_i - \frac{f(\lambda_i)}{\gamma} \frac{f(\lambda_i) - \gamma}{f'(\lambda_i)}, \tag{8.8.22}$$

which is often known as **Hebden's method**. Note that in this iteration the correction to λ_i is larger by a factor of $f(\lambda_i)/\gamma$ than for the previous Newton iteration. It can be shown that the function $1/f(\lambda)$ is concave and hence also the iteration

(8.8.22) is monotonically convergent to the solution λ^* if started within $[0, \lambda^*]$. (For a proof see Reinsch [20, 1971].) It is also possible to use the secant method. Then, if the initial iterates are both non-negative, this method also is monotonically convergent.

Algorithm 8.8.1
Hebden's method. Assume that the function qr computes the "economy size" QR factorization, with $Q \in \mathbf{R}^{m\times n}$. Then Hebden's method is as follows:

$$
\begin{aligned}
&\lambda = \lambda_0;\\
&\text{for } k = 1, 2, \ldots\\
&\qquad [Q, R] = \mathrm{qr}\big([A; \sqrt{\lambda} I]\big);\\
&\qquad c = Q(1:m, 1:n)^T b;\\
&\qquad x = R^{-1} c;\\
&\qquad n_x = \|x\|;\\
&\qquad z = R^{-T} x;\\
&\qquad n_z = \|z\|;\\
&\qquad \lambda = \lambda - (1 - n_x/\gamma) * (n_x/n_z)^2;\\
&\text{end}
\end{aligned}
$$

Consider now the computation of the QR factorization (8.8.18) when A and L both are banded matrices with bandwidth w_1 and w_2, respectively. If we first compute the QR factorizations of A and L it remains to compute the QR factorizations of a matrix of the form

$$
\begin{pmatrix} R_1 \\ \sqrt{\lambda} R_2 \end{pmatrix},
$$

with R_1 and R_2 upper triangular and banded. For this task a row-wise Givens algorithm can be constructed, which is optimal in that no unnecessary fill-in is created and requires approximately $2n(w_1^2 + w_2^2)$ multiplications.

Example 8.8.2. We illustrate the importance of using an optimal ordering of the Givens rotations in computing the QR factorization by considering the case $w_1 = 3$, $w_2 = 1$, and $n = 6$. First we show the result after three steps without reordering: It can be deduced that if we proceed in this fashion $1 + 2 + \cdots + n = \frac{1}{2}n(n+1)$ rotations are needed for the reduction, each using $4w_1$ multiplications, i.e., a total of about $2n(n+1)w_1$.

$$\begin{pmatrix} \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \\ \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{pmatrix} \Rightarrow \begin{pmatrix} \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \\ \otimes & \oplus & \oplus & + & + & \\ & \otimes & \oplus & + & + & \\ & & \otimes & + & + & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{pmatrix}.$$

It can be shown that if $w_2 \leq w_1$ then an optimal ordering is to first take the $w_1 - w_2 + 1$ first rows in R_2 and then interleave the rows of R_1 and R_2. With this ordering the result after the same number of steps in this example is

$$\begin{pmatrix} \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ \times & \times & \times & & & \\ & & & \times & & \\ & \times & \times & \times & & \\ & & & & \times & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \\ & & & & & \times \end{pmatrix} \Rightarrow \begin{pmatrix} \times & + & + & & & \\ & \times & + & + & & \\ & & \times & + & + & \\ \otimes & \otimes & \otimes & & & \\ & & & \times & & \\ & \otimes & \otimes & \times & & \\ & & & & \times & \\ & & \otimes & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \\ & & & & & \times \end{pmatrix}.$$

Now the reduction, each using $4w_1$ multiplications, i.e., a total of about $2nw(w_1+1)$ multiplications.

We now describe an algorithm for the case when $L \neq I$. The idea is to transform (8.8.17) to a regularization problem of standard form. Note that if L is nonsingular we can achieve this by the change of variables $y = Lx$. However, often $L \in \mathbf{R}^{(n-t)\times n}$ with rank $n - t < n$. For example,

$$L = \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{pmatrix} \in \mathbf{R}^{(n-1)\times n}, \tag{8.8.23}$$

which, except for a scaling factor, approximates the first derivative operator. The transformation to standard form can then be achieved using the pseudoinverse of L. Compute the QR factorization of L^T:

$$L^T = (V_1, V_2) \begin{pmatrix} R_2 \\ 0 \end{pmatrix},$$

where V_2 spans the nullspace of L. If we set $y = Lx$, then

$$x = L^I y + V_2 w, \qquad L^I = V_1 R_2^{-T}, \tag{8.8.24}$$

where L^I is the pseudoinverse of L, and $Ax - b = AL^I y - b + AV_2 w$. We form AV_2 and compute its QR factorization

$$AV_2 = (Q_1, Q_2) \begin{pmatrix} U \\ 0 \end{pmatrix}, \qquad U \in \mathbf{R}^{t \times t}.$$

Then

$$Q^T(Ax - b) = \begin{pmatrix} Q_1^T(AL^I y - b) + Uw \\ Q_2^T(AL^I y - b) \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}.$$

Now, if A and L have no nullspace in common, then AV_2 has rank t and U is nonsingular. Thus, we can always determine w so that $r_1 = 0$ and (8.8.17) is equivalent to

$$\min_y \left\| \begin{pmatrix} \tilde{A} \\ \mu I \end{pmatrix} y - \begin{pmatrix} \tilde{b} \\ 0 \end{pmatrix} \right\|_2, \quad \tilde{A} = Q_2^T A L^I, \quad \tilde{b} = Q_2^T b, \tag{8.8.25}$$

This is a problem of standard form for y which can be solved efficiently by transformation of $\tilde{A}$ to bidiagonal form. We can then retrieve x from (8.8.24).

8.8.3 Problem LSQI by Bidiagonalization.

We have seen that in order to solve the secular equation $\|x(\lambda)\|_2 - \gamma = 0$, we need to compute the solution $x = x(\lambda)$ to the least squares problem (8.8.17) for a sequence of values of λ. For each new value the QR factorizations has to be computed from anew. We now show how computations can be saved first transforming A to upper bidiagonal form.

It is no restriction to assume that the problem is in standard form with $L = I_n$. Using one of the algorithms described in Section 8.6.6 we compute the factorization

$$A = Q \begin{pmatrix} B \\ 0 \end{pmatrix} P^T, \tag{8.8.26}$$

where $B \in \mathbf{R}^{n \times n}$ is upper bidiagonal and P and Q are orthogonal matrices. If we put

$$x = Py, \qquad \tilde{b} = Q^T b = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix} \begin{matrix} \}n \\ \}m-n \end{matrix}, \tag{8.8.27}$$

problem (8.8.17) is transformed into

$$\min_x \left\| \begin{pmatrix} B \\ \mu I_n \end{pmatrix} y - \begin{pmatrix} \tilde{b}_1 \\ 0 \end{pmatrix} \right\|_2, \tag{8.8.28}$$

where $\mu = \sqrt{\lambda}$. Since P is orthogonal $\|x\|_2 = \|y\|_2$ and hence the secular equation becomes $\|y(\lambda)\|_2 = \gamma$. To compute $y(\lambda)$ for a given value of λ we determine two sequences of Givens transformations $G_1, \ldots, G_n$ and $J_1, \ldots, J_{n-1}$, so that

$$G_n J_{n-1} \cdots G_2 J_1 G_1 \begin{pmatrix} B & \tilde{b}_1 \\ \mu I_n & 0 \end{pmatrix} = \begin{pmatrix} B_\mu & g_\mu \\ 0 & f_\mu \end{pmatrix}, \tag{8.8.29}$$

where B_μ is again upper bidiagonal. The details of the construction of the Givens rotations G_k and J_k are left to Problem 3. Then $y(\lambda)$ is computed from the bidiagonal system $B_\lambda y(\lambda) = g_\mu$ by back-substitution. The derivatives needed in the Newton or Hebden iteration are computed as before.

8.8.4 Total Least Squares

In the standard linear model (8.1.2) 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–Markoff theorem (Theorem 8.1.4) the best unbiased estimate of x is obtained by solving the least squares problem

$$\min_x \|r\|_2, \qquad Ax = b + r. \tag{8.8.30}$$

The assumption that A is exactly known is frequently unrealistic, and sampling or modeling errors often affect also the matrix A. In the **errors-in-variables model** one assumes 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. The estimates of the true but unknown parameters x in this model is obtained from the solution of the total least squares (TLS) problem[4]

$$\min_{E,\,r} \|(E,\, r)\|_F, \qquad (A + E)x = b + r, \tag{8.8.31}$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm.

We note that the constraint in (8.8.31) implies that $b + r \in \mathcal{R}(A + E)$. If a minimizing $(E,\ r)$ has been found for the problem (8.8.31) then any x satisfying $(A + E)x = b + r$ is said to solve the TLS problem.

It is important to note that the TLS solution depends on the relative scaling of A and b. If we scale x and b by a factor α the TLS problem is transformed into

$$\min_{E,\,r} \|(E,\, \alpha r)\|_F \qquad (A + E)x = b + r.$$

Obviously when $1/\alpha \to 0$, perturbations in b will be favored, which leads to the ordinary least squares solution. The TLS solution will similarly depend on the scaling of the columns of A. In the following we assume that the scaling of the data $(A,\, b)$ of the TLS problem is done so that any statistical knowledge of the perturbations have been taken into account.

The constraint in (8.8.31) can also be written

$$(\tilde{A}, \tilde{b}) \begin{pmatrix} x \\ -1 \end{pmatrix} = 0, \qquad \tilde{A} = A + E, \qquad \tilde{b} = b + r,$$

[4]The term "total least squares problem" was coined by Golub and Van Loan. The concept has been independently developed in other areas and is known in statistics also as "latent root regression".

which shows that the matrix $(A+E,\, b+r)$ is rank deficient and that $(x,-1)^T$ is a corresponding right singular vector. Hence the TLS problem involves finding a perturbation matrix having minimal Frobenius norm, which lowers the rank of the matrix $(A,\, b)$.

The total least squares problem can be analyzed in terms of the singular value factorization

$$(A,\, b) = U\Sigma V^T, \qquad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_{n+1}),$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{n+1} \geq 0$ are the singular values of (A, b). By Theorem 8.3.5 the singular values of $\hat{\sigma}_i$ of A interlace those of $(A,\, b)$, i.e.,

$$\sigma_1 \geq \hat{\sigma}_1 \geq \sigma_2 > \ldots \geq \sigma_n \geq \hat{\sigma}_n \geq \sigma_{n+1}. \tag{8.8.32}$$

We assume in the following that A has full rank, that is, $\sigma'_n > 0$, and that $\sigma_n > \sigma_{n+1}$. Then the minimum is attained for the rank one perturbation

$$(E,\, r) = -(A,\, b)v_{n+1}v_{n+1}^T = -\sigma_{n+1}u_{n+1}v_{n+1}^T,$$

for which $\|(E,\, r)\|_F = \sigma_{n+1}$. A TLS solution is given by

$$v_{n+1} = \begin{pmatrix} z \\ \gamma \end{pmatrix} = -\gamma \begin{pmatrix} x_{TLS} \\ -1 \end{pmatrix}. \tag{8.8.33}$$

If $\gamma = 0$ the TLS problem is called *nongeneric*, and there is no solution. This case can only occur when either $\sigma'_n = \sigma_{n+1} = 0$, or the data is highly conflicting. Here and in the following we always assume that $\gamma \neq 0$, which is the generic case, see [24].

If σ_{n+1} is a repeated singular value, i.e., $k < n$, then the TLS problem may lack a unique solution. In this case a unique minimum norm TLS solution can be determined as follows. Let Q be an orthogonal matrix of order $n-k+1$ such that

$$[v_{k+1}, \ldots, v_{n+1}]Q = \begin{pmatrix} W & y \\ 0 & \alpha \end{pmatrix} \begin{matrix} \}n \\ \}1 \end{matrix}.$$

If we set $x = -\alpha^{-1}y$ then it is easy to show that all other solution to the TLS problem have larger norms. Note that Q can be taken as the Householder transformation which zeros all leading elements in the last row. The TLS problem has no solution if $e_{n+1} = (0, \ldots, 0, 1)^T$ is orthogonal to S_C.

There are several generalizations of the total least squares problem. Golub and Van Loan [13, Sec. 12.3] consider the TLS problem with multiple right hand sides. Given a matrix $B \in \mathbf{R}^{m,d}$, $d > 1$, this problem is

$$\min_{E,\, F} \|(E,\, F)\|_F, \qquad (A+E)X = B + F. \tag{8.8.34}$$

Hence we now seek a perturbation (E, B) that reduces the rank of the matrix (A, B) by d. The solution can now be expressed in terms of the singular value factorization

$$(A,\, B) = (U_1,\, U_2) \begin{pmatrix} \Sigma_1 & \\ & \Sigma_2 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix},$$

where

$$\Sigma_1 = \mathrm{diag}\,(\sigma_1, \ldots, \sigma_n), \qquad \Sigma_2 = \mathrm{diag}\,(\sigma_{n+1}, \ldots, \sigma_{n+d}).$$

Clearly the minimizing perturbation is $(E,\, F) = -U_2 \Sigma_2 V_2^T$ The condition $\hat{\sigma}_n > \sigma_{n+1}$ ensures that there exists a unique TLS solution, given by

$$X_{TLS} = -V_{12} V_{22}^{-1}, \qquad V = \begin{pmatrix} V_{12} \\ V_{22} \end{pmatrix}. \tag{8.8.35}$$

An excellent survey of the theory and computational aspects of the total least squares problem is given by Van Huffel and Vandewalle [24].

We now consider the conditioning of the total least squares problem and its relation to the least squares problem. To ensure unique solutions to both the TLS and the least squares problems we assume that $\hat{\sigma}_n > \sigma_{n+1}$ and we denote those solutions by x_{TLS} and x_{LS} respectively. The vector $(x_{TLS}, -1)^T$ is an eigenvector of $C^T C$ with the associated eigenvalue σ_{n+1}^2, i.e.,

$$\begin{pmatrix} A^T A & A^T b \\ b^T A & b^T b \end{pmatrix} \begin{pmatrix} x_{TLS} \\ -1 \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} x_{TLS} \\ -1 \end{pmatrix}.$$

The first block row of this equation can be written

$$(A^T A - \sigma_{n+1}^2 I) x_{TLS} = A^T b. \tag{8.8.36}$$

Hence a positive multiple of the unit matrix is **subtracted** from $A^T A$ in the TLS. Since

$$\kappa(A^T A - \sigma_{n+1}^2 I) = \frac{\hat{\sigma}_1^2 - \sigma_{n+1}^2}{\hat{\sigma}_n^2 - \sigma_{n+1}^2} > \frac{\hat{\sigma}_1^2}{\hat{\sigma}_n^2} = \kappa(A^T A)$$

it follows that the TLS problem is always *worse conditioned* than the LS problem.

The least squares solution satisfies $A^T A x_{LS} = A^T b$, and subtracting this from equation (8.8.36) we get $x_{TLS} - x_{LS} = \sigma_{n+1}^2 (A^T A - \sigma_{n+1}^2 I)^{-1} x_{LS}$. Taking norms we obtain

$$\frac{\|x_{TLS} - x_{LS}\|_2}{\|x_{LS}\|_2} \leq \frac{\sigma_{n+1}^2}{\hat{\sigma}_n^2 - \sigma_{n+1}^2}.$$

This shows that when the difference $\hat{\sigma}_n^2 - \sigma_{n+1}^2$ is small then the TLS solution can differ much from the LS solution.

8.8.5 Linear Orthogonal Regression

A special case of the total least squares problem (8.8.31) is the following **orthogonal regression** problem. Let $y_i \in \mathbf{R}^n$, $i = 1, 2, \ldots, m$, be $m > n$ given points. We want to determine a hyperplane M in $\mathbf{R}^n$ such that the sum of squares of the orthogonal distances from the given points to M is minimized. The equation for the hyperplane can be written

$$c^T z = h, \quad z, c \in \mathbf{R}^n, \quad \|c\|_2 = 1,$$

where c is the normal vector of M, and $h \in \mathbf{R}$. Then the orthogonal projections of the points y_i onto M are given by

$$z_i = y_i - (c^T y_i - h)c. \tag{8.8.37}$$

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

$$\psi(c,h) = \sum_{i=1}^{m} (c^T y_i - h)^2, \quad \text{subject to} \quad \|c\|_2 = 1.$$

If we put $Y = (y_1, \ldots, y_m)$, $e = (1, \ldots, 1)^T$, this problem can be written in matrix form

$$\min_{c,h} \left\| (Y^T, -e) \begin{pmatrix} c \\ h \end{pmatrix} \right\|_2, \qquad \|c\|_2 = 1, \tag{8.8.38}$$

For a fixed c, this expression is minimized when the residual vector $(Y^T c - he)$ is orthogonal to e, that is $e^T(Y^T c - he) = e^T Y^T c - he^T e = 0$. Since $e^T e = m$ it follows that

$$h = c^T Y e/m = c^T \bar{y}, \qquad \bar{y} = Ye/m, \tag{8.8.39}$$

where $\bar{y}$ is the mean value of the given points y_i. Hence h is determined by the condition that the mean value $\bar{y}$ lies on the optimal plane M.

We now subtract the mean value $\bar{y}$ from the each given point, and form the matrix

$$\bar{Y} = (\bar{y}_1, \ldots, \bar{y}_m), \qquad \bar{y}_i = y_i - \bar{y}, \quad i = 1, \ldots, m.$$

Since by (8.8.39)

$$(Y^T, -e) \begin{pmatrix} c \\ h \end{pmatrix} = Y^T c - e\bar{y}^T c = (Y^T - e\bar{y}^T)c = \bar{Y}^T c,$$

problem (8.8.38) is equivalent to

$$\min_c \|\bar{Y}^T c\|_2, \qquad \|c\|_2 = 1 \tag{8.8.40}$$

By the min-max characterization of the singular values (Theorem 8.3.3) a solution to (8.8.40) is $c = v_n$, where v_n is a right singular vector of $\bar{Y}^T$ corresponding to the smallest singular value σ_n. Hence, the solution to problem (8.8.38) is given by

$$c = v_n, \qquad h = v_n^T \bar{y}, \qquad \sum_{i=1}^{m} (v_n^T y_i - h)^2 = \sigma_n.$$

The fitted points $z_i \in M$ are obtained from

$$z_i = \bar{y}_i - (v_n^T y_i)v_n + \bar{y},$$

i.e., by first orthogonalizing the shifted points $\bar{y}_i$ against v_n, and then adding the mean value back.

Note that in contrast to the TLS problem the orthogonal regression problem always has a solution. The solution is unique when $\sigma_{n-1} \neq \sigma_n$, and the minimum sum of squares equals σ_n^2. We have $\sigma_n = 0$, if and only if the given points y_i, $i = 1, \ldots, m$ all lie on the hyperplane M. In the extreme case, all points coincide and then $\bar{Y} = 0$, and any plane going through $\bar{y}$ is a solution.

The above method solves the problem of fitting a $(n-1)$ dimensional linear manifold to a given set of points in R. It is readily generalized to the fitting of an $(n-p)$ dimensional manifold by orthogonalizing the shifted points y against the p left singular vectors of Y corresponding to p smallest singular values. A least squares problem that often arises is to fit to given data points a geometrical element, which may be defined in implicit form. For example, the problem of fitting circles, ellipses, spheres, and cylinders arises in applications such as computer graphics, coordinate meteorology, and statistics. Such problems are nonlinear and will be discussed in Section 11.4.7.

Example 8.8.3.

Suppose we want to fit by orthogonal regression m pair of points $(x_i, y_i) \in \mathbf{R}^2$, $i = 1, \ldots, m$, to a straight line

$$cx + sy = h, \qquad c^2 + s^2 = 1.$$

First compute the mean values of x_i and y_i and the QR factorization of the matrix of shifted points

$$\bar{Y}^T = \begin{pmatrix} \bar{x}_1 & \bar{x}_2 & \cdots & \bar{x}_m \\ \bar{y}_1 & \bar{y}_2 & \cdots & \bar{y}_m \end{pmatrix}^T = Q \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where R is an upper triangular 2×2 matrix. Since the singular values and right singular vectors of $\bar{Y}^T$ and R are the same, it suffices to compute the SVD

$$R = \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix} = (u_1 \ u_2) \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix},$$

where $\sigma_1 \geq \sigma_2 \geq 0$. Then the coefficients in the equation of the straight line are given by

$$(c \ s) = v_2^T, \qquad h = v_2^T \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix}.$$

If $\sigma_2 = 0$ but $\sigma_1 > 0$ the matrix $\bar{Y}$ has rank one. In this case the given points lie on a straight line. If $\sigma_1 = \sigma_2 = 0$, then $\bar{Y} = 0$, and $x_i = \bar{x}$, $y_i = \bar{y}$ for all $i = 1, \ldots, m$. Note that u_2 is uniquely determined if and only if $\sigma_1 \neq \sigma_2$. It is left to the reader to discuss the case $\sigma_1 = \sigma_2 \neq 0$!

8.8.6 Iteratively Reweighted Least Squares.

In some applications it might be more adequate to solve the problem

$$\min \|Ax - b\|_p \tag{8.8.41}$$

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 (8.8.41) 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 8.8.4. *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 $A = e = (1, 1, \ldots, 1)^T$. It is easily verified that if $b_1 \geq b_2 \geq \ldots \geq b_m$, then the solution x_p for some different values p are*

$$x_1 = b_{\frac{m+1}{2}}, \quad (m \text{ odd})$$
$$x_2 = \frac{1}{m}(b_1 + b_2 + \ldots + b_m),$$
$$x_\infty = \frac{1}{2}(b_1 + b_m).$$

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 [18, 1985]) can be used to reduce the problem to a sequence of weighted least squares problems.

We start by noting that, provided that $|r_i(x)| = |b - Ax|_i > 0$, $i = 1, \ldots, m$, the problem (8.8.41) can be restated in the form $\min_x \psi(x)$, where

$$\psi(x) = \sum_{i=1}^{m} |r_i(x)|^p = \sum_{i=1}^{m} |r_i(x)|^{p-2} r_i(x)^2. \tag{8.8.42}$$

This can be interpreted as a weighted least squares problem

$$\min_x \|D(r)^{(p-2)/2}(b - Ax)\|_2, \qquad D(r) = \text{diag}\,(|r|), \tag{8.8.43}$$

where $\text{diag}\,(|r|)$ denotes the diagonal matrix with ith component $|r_i|$.

The diagonal weight matrix $D(r)^{(p-2)/2}$ in (8.8.43) depends on the unknown solution x, but we can attempt to use the following iterative method.

Algorithm 8.8.2
IRLS for l_p Approximation $1 < p < 2$
Let $x^{(0)}$ be an initial approximation such that $r_i^{(0)} = (b - Ax^{(0)})_i \neq 0$, $i = 1, \ldots, n$.

$$\text{for } k = 0, 1, 2, \ldots$$

$$
\begin{aligned}
&r_i^{(k)} = (b - Ax^{(k)})_i;\\
&D_k = \operatorname{diag}\left(\left(|r_i^{(k)}|\right)^{(p-2)/2}\right);\\
&\text{solve } \delta x^{(k)} \text{ from}\\
&\qquad \min_{\delta x} \left\| D_k(r^{(k)} - A\delta x)\right\|_2;\\
&x^{(k+1)} = x^{(k)} + \delta x^{(k)};\\
&\text{end}
\end{aligned}
$$

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 $r_i^{(k)} \neq 0$, $i = 1, \ldots, n$. In practice this cannot be guaranteed, and it is customary to modify the algorithm so that

$$D_k = \operatorname{diag}\left(\left(100ue + |r^{(k)}|\right)^{(p-2)/2}\right),$$

where u is the machine precision and $e^T = (1, \ldots, 1)$ is the vector of all ones. Because the weight matrix D_k is not constant, the simplest implementations of IRLS recompute, e.g., the QR factorization of $D_k A$ in each step. It should be pointed out that the iterations can be carried out entirely in the r space without the x variables. Upon convergence to a residual vector r_{opt} the corresponding solution can be found by solving the consistent linear system $Ax = b - r_{\text{opt}}$.

It can be shown that in the l_p case any fixed point of the IRLS iteration satisfies the necessary conditions for a minimum of $\psi(x)$. The IRLS method is convergent for $1 < p < 3$, and also for $p = 1$ provided that the l_1 approximation problem has a unique nondegenerate solution. However, the IRLS method can be extremely slow when p is close to unity.

Review Questions

1. What is meant by a saddle-point system? Which two optimization problems give rise to saddle-point systems?
2. Formulate the total least squares (TLS) problem. The solution of the TLS problem is related to a theorem on matrix approximation. Which?

Problems

1. 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. (6.2.12)). Conclude that M is indefinite! (M is called a saddle point matrix.)

2. Show that the derivative $x'(\lambda)$ is the solution to the least squares problem

$$\min_{x'} \left\| \begin{pmatrix} A \\ \mu L \end{pmatrix} x' + \mu^{-1} \begin{pmatrix} 0 \\ Lx \end{pmatrix} \right\|_2, \quad \mu = \lambda^{1/2}.$$

Use this result to show how to compute x' using the QR factorization (8.8.18).

3. Describe for $n = 3$ the reduction to bidiagonal form in (8.8.29). Choose the first transformation G_1 is chosen to zero the element in position (4,1). This creates a new nonzero element in position (4,2), which is then annihilated by the transformation J_1. This step reduces the dimension of the problem by one and the transformation proceeds recursively. Show that the transformation to bidiagonal form and the computation of $y(\lambda)$ takes only about $11n$ flops.

4. 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 $x = 2/\epsilon$.

Computer Exercises

1. Write a MATLAB program for fitting a straight line $c_1 x + c_2 y = h$ to given points $(x_i, y_i) \in \mathbf{R}^2$, $i = 1, 2, \ldots, m$. Follow the outline in Example 8.8.3. Use the Algorithm 10.4.2 to compute the SVD of R. The program should handle all exceptional cases, e.g., $c_1 = 0$ or and/or $c_2 = 0$.

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

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 $|r_i|$. 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 in 1805. However, Gauss claimed to have used the method since 1795 and later justified it as a statistical procedure in 1809. This led to one of the most famous priority dispute in the history of mathematics. Gauss further developed the statistical aspects in 1821–1823.

Because of its success in analyzing astronomical data the method of least squares became the method of choice. Geodetic calculations was another early area of application of the least squares principle. In the last decade applications in signal processing has been a source of inspiration for developing new methods.

The exposition of oblique projections in Section 7.1.3 is adapted from Wedin [25]. The systematic use of orthogonal transformations to reduce matrices to simpler form was initiated by Givens [10, 1958] and Householder [15, 1958]. The application of these transformations to linear least squares is due to Golub [11, 1965]. For an interesting accounts of the history of the invention of least squares, see Stiegler [23, 1981]. The early history of the SVD is documented in [21, 1993].

References

[1] Benoit. Sur la méthode de résolution des équations normales, etc. (procédés du commandant cholesky). *Bull. Géodesique*, 2:67–77, 1924.

[2] Å. Björck. Numerics of Gram–Schmidt orthogonalization. *Linear Algebra Appl.*, 197–198:297–316, 1994.

[3] Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, PA, 1996.

[4] Å. Björck and C. C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, 13:176–190, 1992.

[5] T. F. Chan. Rank revealing QR factorizations. *Linear Algebra Appl.*, 88/89:67–82, 1987.

[6] J. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comp.*, 30:772–795, 1976.

[7] K. Fan and A. Hoffman. Some metric inequalities in the space of matrices. *Proc. Amer. Math. Soc.*, 6:111–116, 1955.

[8] C. F. 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.

[9] 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.

[10] W. Givens. Computation of plane unitary rotations transforming a general matrix to triangular form. *SIAM J. Appl. Math.*, 6:26–50, 1958.

[11] G. H. Golub. Numerical methods for solving least squares problems. *Numer. Math.*, 7:206–216, 1965.

[12] G. H. Golub and W. Kahan. Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal. Ser. B*, 2:205–224, 1965.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[14] Y. T. Hong and C. T. Pan. Rank-revealing qr decompositions and the singular value decomposition. *Math. Comp.*, 58:213–232, 1992.

[15] A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.*, 5:339–342, 1958.

[16] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1974.

[17] A. A. Markoff. *Wahrscheinlichheitsrechnung.* Liebmann, Leipzig, second edition, 1912.

[18] M. R. Osborne. *Finite Algorithms in Optimization and Data Analysis.* John Wiley, New York, 1985.

[19] C. C. Paige and Z. Strakos. 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.

[20] C. H. Reinsch. Smoothing by spline functions. *Numer. Math.*, 16:451–454, 1971.

[21] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35:4:551–556, 1993.

[22] G. W. Stewart. The QLP approximation to the singular value decomposition. *SIAM J. Sci. Comput.*, 20:4:1336–1348, 1999.

[23] S. M. Stiegler. Gauss and the invention of least squares. *Ann. Statist.*, 9:465–474, 1981.

[24] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem; Computational Aspects and Analysis*. SIAM, Philadelphia, PA, 1991.

[25] P.-Å. Wedin. Perturbation theory and condition numbers for generalized and constrained linear least squares problems. Tech. Report UMINF–125.85, Institute of Information Processing, University of Umeå, 1985.

[26] M. Zelen. Linear estimation and related topics. In J. Todd, editor, *Survey of Numerical Analysis*, pages 558–584. McGraw-Hill, New York, 1962.

Chapter 9

Matrix Eigenvalue Problems

9.1 Basic Properties

9.1.1 Introduction

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 also play 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. In particular Section 9.1 gives a brief account of basic facts of the matrix eigenvalue problem, Section 9.2 treats the classical theory of canonical forms and matrix functions. Section 9.3 is devoted to the localization of eigenvalues and perturbation results for eigenvalues and eigenvectors.

Section 9.4 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 Section 9.5. Transformation to condensed form described in Section 9.5 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 Section 9.7. This method can also be adopted to compute singular values and singular vectors although the numerical implementation is often far from trivial, see Section 9.7.

In Section 9.8 we briefly discuss some methods for solving the eigenvalue prob-

lem for large sparse matrices. Finally, in Section 9.9 we consider the generalized eigenvalue problem $Ax = \lambda Bx$, and the generalized SVD.

9.1.2 Complex Matrices

In developing the theory for the matrix eigenvalue problem it often is more relevant to work with complex vectors and matrices. This is so because a real unsymmetric matrix can have complex eigenvalues and eigenvectors. We therefore introduce the vector space $\mathbf{C}^{n\times m}$ of all complex $n \times m$ matrices whose components are complex numbers.

Most concepts and operations in Section 7.2 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 Hermitian inner product of two vectors x and y in $\mathbf{C}^n$ is defined as

$$(x, y) = x^H y = \sum_{k=1}^{n} \bar{x}_k y_k, \tag{9.1.1}$$

where $x^H = (\bar{x}_1, \ldots, \bar{x}_n)$ and $\bar{x}_i$ denotes the complex conjugate of x_i. Hence $(x, y) = \overline{(y, x)}$, and $x \perp y$ if $x^H y = 0$. The Euclidean length of a vector x thus becomes

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

The set of complex $m \times n$ matrices is denoted by $\mathbf{C}^{m\times n}$. If $A = (a_{ij}) \in \mathbf{C}^{m\times n}$ then by definition its **adjoint** matrix $A^H \in \mathbf{C}^{n\times m}$ satisfies

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

By using coordinate vectors for x and y it follows that $A^H = \bar{A}^T$, that is, A^H is the conjugate transpose of A. It is easily verified that $(AB)^H = B^H A^H$. In particular, if α is a scalar $\alpha^H = \bar{\alpha}$.

A matrix $A \in \mathbf{C}^{n\times n}$ 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

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

A square matrix U is **unitary** if $U^H U = I$. From (9.1.1) we see that a unitary matrix preserves the Hermitian inner product

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

In particular the 2-norm is invariant under unitary transformations, $\|Ux\|_2^2 = \|x\|_2^2$. Hence, unitary matrices corresponds to real orthogonal matrices. Note that in every case, the new definition coincides with the old when the vectors and matrices are real.

9.1.3 Theoretical Background

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. A complex scalar λ such that

$$Ax = \lambda x, \qquad x \neq 0, \tag{9.1.2}$$

is called an **eigenvalue** of A and x is an **eigenvector** of A. Hence λ is an eigenvalue if and only if the linear homogeneous system $(A - \lambda I)x = 0$ has a nontrivial solution $x \neq 0$, or equivalently if and only if $A - \lambda I$ is singular. It follows that the eigenvalues satisfy the **characteristic equation**

$$p(\lambda) = \det(A - \lambda I) = 0. \tag{9.1.3}$$

Obviously, if x is an eigenvector so is αx for any scalar $\alpha \neq 0$.

The set $\lambda(A) = \{\lambda_i\}_{i=1}^n$ of all eigenvalues of A is called the **spectrum**[1] of A. The polynomial $p(\lambda) = \det(A - \lambda I)$ is the **characteristic polynomial** of the matrix A. Expanding the determinant in (9.1.3) it follows that $p(\lambda)$ has the form

$$p(\lambda) = (a_{11} - \lambda)(a_{22} - \lambda)\cdots(a_{nn} - \lambda) + q(\lambda), \tag{9.1.4}$$

where $q(\lambda)$ has degree at most $n-2$. Hence $p(\lambda)$ is a polynomial of degree n in λ with leading term $(-1)^n\lambda^n$. Thus, by the fundamental theorem of algebra the matrix A has exactly n eigenvalues λ_i, $i = 1, 2, \ldots n$, counting multiple roots according to their multiplicities, and we can write

$$p(\lambda) = (\lambda_1 - \lambda)(\lambda_2 - \lambda)\cdots(\lambda_n - \lambda).$$

Putting $\lambda = 0$ here and in (9.1.3) it follows that

$$p(0) = \lambda_1\lambda_2\cdots\lambda_n = \det(A), \tag{9.1.5}$$

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.

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. □

[1] From Latin verb *specere* meaning "to look".

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

$$y_i^H A = \lambda_i y_i^H, \qquad Ax_j = \lambda_j x_j.$$

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. □

If X is any square nonsingular matrix and

$$\tilde{A} = X^{-1} A X, \tag{9.1.8}$$

then $\tilde{A}$ is said to be similar to A and (9.1.8) 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 9.1.3.

If A and B are similar, then A and B have the same characteristic polynomial, and hence the same eigenvalues. Further, if $B = X^{-1}AX$ and y is an eigenvector of B corresponding to λ then Xy is an eigenvector of A corresponding to λ.

Proof. We have

$$\begin{aligned}\det(B - \lambda I) &= \det(X^{-1}AX - \lambda I) = \det(X^{-1}(A - \lambda I)X) \\ &= \det\left(X^{-1}\right) \det(A - \lambda I) \det(X) = \det(A - \lambda I).\end{aligned}$$

Further, from $AX = XB$ it follows that $AXy = XBy = \lambda Xy$. □

Let $Ax_i = \lambda_i x_i$, $i = 1, \ldots, n$. 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 $X = (x_1, \ldots, x_n)$ is a matrix of right eigenvectors of A If the eigenvectors are linearly independent then X is nonsingular and we have

$$X^{-1} A X = \Lambda. \tag{9.1.9}$$

This similarity transformation by X transforms A to diagonal form and A is said to be **diagonalizable**.

From (9.1.9) 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

$$A = \sum_{i=1}^{n} \lambda_i P_i, \quad P_i = x_i y_i^H. \tag{9.1.10}$$

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 (9.1.10) is called the **spectral decomposition** of A. The decomposition (9.1.10) 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 $A - \lambda_{i_1} I$.

9.1.4 Invariant Subspaces

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*. Note that any set of right eigenvectors spans a right invariant subspace.

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.

Definition 9.1.4.

A matrix $A \in \mathbf{R}^{n\times n}$ *is said to be* **reducible** *if there exists a permutation matrix* P *such that* $P^T AP$ *has the form*

$$P^T AP = \tilde{A} = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}, \tag{9.1.11}$$

where $A_{11} \in \mathbf{R}^{r\times r}$ *and* $A_{22} \in \mathbf{R}^{s\times s}$, $r + s = n$, *are square submatrices. Otherwise* A *is called* **irreducible**.

The concept of a reducible matrix can be illustrated using some elementary notions from the theory of graphs. The **directed graph** of a matrix A is 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 directed **edge** from node i

to node j. (Compare the definition of an undirected graph of a matrix in Def. 6.5.2.) 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 $P_i = P_{i_1}, P_{i_2}, \ldots, P_{i_p} = P_j$ 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. Then from (9.1.11) we have

$$\tilde{A}\begin{pmatrix} I_r \\ 0 \end{pmatrix} = \begin{pmatrix} I_r \\ 0 \end{pmatrix} A_{11}, \qquad (0 \quad I_s)\,\tilde{A} = A_{22}\,(0 \quad I_s),$$

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 A_{11} and A_{22}. This result generalizes directly to general block upper (lower) triangular matrices.

Theorem 9.1.5.

Assume that the matrix A can be partitioned in block upper triangular form

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ 0 & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A_{NN} \end{pmatrix},$$

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.

Many important numerical methods for computing eigenvalues and eigenvectors of a matrix A perform a *sequence of similarity transformations* to transform A into a matrix of simpler form. With $A_0 = A$ one computes

$$A_k = P_k^{-1} A_{k-1} P_k, \quad k = 1, 2, \ldots.$$

The matrix A_k is similar to A and the eigenvectors x of A and y of A_k are related by $x = P_1 P_2 \cdots P_k y$. The eigenvalues of a triangular matrix equal its diagonal elements. Hence if the matrix A can be transformed by successive similarities to triangular form, then its eigenvalues are trivial to determine.

Let $AX_1 = X_1 B$, for some $X_1 \in \mathbf{R}^{n\times p}$ of rank p, and $B \in \mathbf{R}^{p\times p}$. Then $\mathcal{R}(X_1)$ is a right invariant subspace of A. Let $X_2 \in \mathbf{R}^{n\times(n-p)}$ be such that $X = (X_1, X_2)$ is invertible. Then we have

$$X^{-1}AX = X^{-1}(AX_1, AX_2) = (X^{-1}X_1 B, X^{-1}AX_2) = \begin{pmatrix} B & T_{12} \\ 0 & T_{22} \end{pmatrix} \tag{9.1.12}$$

that is, $X^{-1}AX$ is reducible. Hence, if a set of eigenvalues of A and a basis X_1 for a corresponding right invariant are known, then we can find the remaining

eigenvalues of A from T_{22}. This process is called **deflation** and is a powerful tool for computation of eigenvalues and eigenvectors. Note that if $X_1 = Q_1$ has orthonormal columns, then $X = (Q_1, Q_2)$ in (9.1.12) can be chosen as an orthogonal matrix.

A matrix A may not have a full set of n linearly independent eigenvectors. However, it holds:

Theorem 9.1.6.

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 and hence A is diagonalizable.

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. □

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

$$\rho_i = n - \operatorname{rank}(A - \lambda_i I). \tag{9.1.13}$$

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 9.1.7.

For the geometric and algebraic multiplicity the inequality $\rho(\lambda) \leq \sigma(\lambda)$ holds.

Proof. Let $\bar{\lambda}$ be an eigenvalue with geometric multiplicity $\rho = \rho(\bar{\lambda})$ and let $x_1, \ldots, x_\rho$ be linearly independent eigenvectors associated with $\bar{\lambda}$. If we put $X_1 = (x_1, \ldots, x_\rho)$ 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. □

If $\rho(\lambda) < \sigma(\lambda)$ then λ is said to be a **defective eigenvalue**. 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 diagonalizable.

Example 9.1.1.

The matrix $\bar{\lambda}I$, where I is a unit matrix of dimension n has the characteristic polynomial $p(\lambda) = (\bar{\lambda}-\lambda)^n$ and hence $\lambda = \bar{\lambda}$ is an eigenvalue of algebraic multiplicity equal to n. Since $\operatorname{rank}(\bar{\lambda}I - \bar{\lambda}\cdot I) = 0$, there are n linearly independent eigenvectors associated with this eigenvalue. Clearly any vector $x \in \mathbf{C}^n$ is an eigenvector.

Now consider the nth order matrix

$$J_n(\bar{\lambda}) = \begin{pmatrix} \bar{\lambda} & 1 & & \\ & \bar{\lambda} & \ddots & \\ & & \ddots & 1 \\ & & & \bar{\lambda} \end{pmatrix}. \tag{9.1.14}$$

Also this matrix has the characteristic polynomial $p(\lambda) = (\bar{\lambda} - \lambda)^n$. However, since $\operatorname{rank}(J_n(\bar{\lambda}) - \bar{\lambda}\cdot I) = n - 1$, $J_n(\bar{\lambda})$ has only one right eigenvector $x = (1, 0, ..., 0)^T$. Similarly it has only one left eigenvector $y = (0, \ldots, 0, 1)^T$, and the eigenvalue $\lambda = \bar{\lambda}$ is defective. A matrix of this form is called a **Jordan block**, see Theorem 9.2.7.

For any nonzero vector $v_1 = v$ the **Krylov sequence** of vectors with respect to A is defined by

$$v_{k+1} = Av_k = A^k v_1. \tag{9.1.15}$$

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 $m \leq n$.) 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 $m \leq n$. If v is such a vector and q its minimal polynomial, then it can be shown that $q(A)x = 0$ for any vector x, 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 9.2.2.

The eigenvalues and eigenvectors of the Kronecker product $A \otimes B$ of $A \in \mathbf{R}^{n\times n}$ and $B \in \mathbf{R}^{m\times m}$ can be simply expressed in terms of the eigenvalues and eigenvectors

of A and B. Assume that $Ax_i = \lambda_i x_i$, $i = 1, \ldots, n$, and $By_j = \mu_j y_j$, $j = 1, \ldots, m$. Then, using equation (7.5.26) we obtain

$$(A \otimes B)(x_i \otimes y_j) = (Ax_i) \otimes (By_j) = \lambda_i \mu_j (x_i \otimes y_j). \tag{9.1.16}$$

This shows that the nm eigenvalues of $A \otimes B$ are $\lambda_i \mu_j$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, 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

$$(I_m \otimes A) + (B \otimes I_n) \in \mathbf{R}^{nm \times nm} \tag{9.1.17}$$

is the **Kronecker sum** of A and B. Since

$$\begin{aligned}\big[(I_m \otimes A) + (B \otimes I_n)\big](y_j \otimes x_i) &= y_j \otimes (Ax_i) + (By_j) \otimes x_i \\ &= (\lambda_i + \mu_j)(y_j \otimes x_i).\end{aligned} \tag{9.1.18}$$

the nm eigenvalues of the Kronecker sum equal the sum of all pairs of eigenvalues of A and B

Review Questions

1. How are the eigenvalues and eigenvectors of A affected by a similarity transformation?
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?
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?

Problems

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.
2. Show that if λ is an eigenvalue of a unitary matrix U then $|\lambda| = 1$.
3. Let $A \in \mathbf{R}^{m \times n}$ and $B \in \mathbf{R}^{n \times m}$. Show that

$$X^{-1} \begin{pmatrix} AB & 0 \\ B & 0 \end{pmatrix} X = \begin{pmatrix} 0 & 0 \\ B & BA \end{pmatrix}, \qquad X = \begin{pmatrix} I & A \\ 0 & I \end{pmatrix}.$$

Conclude that the nonzero eigenvalues of $AB \in \mathbf{R}^{m \times m}$ and $BA \in \mathbf{R}^{n \times n}$ are the same.

4. (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 $P = I - 2uu^T$, $\|u\|_2 = 1$?

5. 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?

6. An upper Hessenberg matrix is called unreduced if all its subdiagonal elements are nonzero. Show that 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.

7. Let $A \in \mathbf{C}^{n\times n}$ be an Hermitian matrix, λ an eigenvalue of A, and z the corresponding eigenvector. Let $A = S + iK$, $z = x + iy$, where S, K, x, y are real. Show that λ is a double eigenvalue of the real symmetric matrix

$$\begin{pmatrix} S & -K \\ K & S \end{pmatrix} \in \mathbf{R}^{2n\times 2n},$$

and determine two corresponding eigenvectors.

8. Show that the matrix

$$K_n = \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

has the characteristic polynomial

$$p(\lambda) = (-1)^n(\lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n).$$

K_n is called the **companion matrix** of $p(\lambda)$. Determine the eigenvectors of K_n corresponding to an eigenvalue λ, and show that there is only one eigenvector even when λ is a multiple eigenvalue.

Remark: The term companion matrix is sometimes used for slightly different matrices, where the coefficients of the polynomial appear, e.g., in the last row or in the last column.

9. A **shift matrix** $S \in \mathbf{R}^{n\times n}$ is a bidiagonal matrix of the form $S = J(0)$, where $J(\mu)$ is defined in (9.1.14). Show that SA and AS describes, respectively, a shift upwards and a shift to the right. What about S^T?

10. A **circulant** matrix $C \in \mathbf{R}^{n\times n}$ generated by $(a_1, a_2, \ldots, a_{n-1}, a_n)$ has the form

$$C = \begin{pmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ a_{n-1} & a_0 & \cdots & a_{n-3} & a_{n-2} \\ \vdots & \vdots & & \vdots & \vdots \\ a_2 & a_3 & \cdots & a_0 & a_1 \\ a_1 & a_2 & \cdots & a_{n-1} & a_0 \end{pmatrix}.$$

(a) Show that $C = a_0 I + a_1 S + \ldots + a_{n-1} S^{n-1}$, where S is a certain simple circulant matrix.

(b) Show that the eigenvalues and eigenvectors of C are given by

$$\lambda_j = a_0 + a_1\omega_j + \ldots + a_{n-1}\omega_j^{n-1}, \quad x_j = \frac{1}{\sqrt{n}}(1, \omega_j, \ldots, \omega_j^{n-1})^T,$$

where $\omega_j = e^{2\pi j/n}$, $j = 1, 2, \ldots, n$ are the n roots of $\omega^n = 1$.

(c) Show that the result in (b) implies that $C = F\Lambda F^H$, where $\Lambda = \text{diag}\,(\lambda_1, \ldots, \lambda_n)$, $F = (x_1, \ldots, x_n)$ is the matrix of the discrete Fourier transform, and the eigenvalues are given by the Fourier transform of its first column

$$F(a_0, a_{n-1}, \ldots, a_2, a_1)^T = (\lambda_1, \ldots, \lambda_n)^T.$$

9.2 Canonical Forms and Matrix Functions

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.

9.2.1 The Schur Canonical Form

The computationally most useful of the canonical forms is the triangular, or **Schur normal form**.

Theorem 9.2.1. 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

$$U^H A U = T = D + N, \tag{9.2.1}$$

where T is upper triangular, N strictly upper triangular, $D = \text{diag}\,(\lambda_1, \cdots, \lambda_n)$, 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

$$S_k = \text{span}[u_1, \ldots, u_k], \qquad k = 1, \ldots, n,$$

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 9.2.2. 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

$$Q^T AQ = T = D + N, \tag{9.2.2}$$

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 $A\bar{x} = \bar{\lambda}\bar{x}$, also $\bar{\lambda}$ is 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 (9.1.12) 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. □

We now introduce a class of matrices for which the Schur normal form is diagonal.

Definition 9.2.3.

A matrix $A \in \mathbf{C}^{n\times n}$ is said to be **normal** *if*

$$A^H A = AA^H. \tag{9.2.3}$$

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 in the Schur normal form is normal,

$$T^H T = TT^H, \qquad T = \begin{pmatrix} \lambda_1 & t_{12} & \dots & t_{1n} \\ & \lambda_2 & \dots & t_{2n} \\ & & \ddots & \vdots \\ & & & \lambda_n \end{pmatrix},$$

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 so $t_{1j} = 0$, $j = 2, \dots, n$. In the same way it can be shown that all the other nondiagonal elements in T vanishes, and so T is diagonal.

Important classes of normal matrices are Hermitian ($A = A^H$), skew-Hermitian ($A^H = -A$), unitary ($A^{-1} = A^H$) and circulant matrices (see Problem 9.1.10). Hermitian matrices have real eigenvalues, skew-Hermitian matrices have imaginary eigenvalues, and unitary matrices have eigenvalues on the unit circle.

Theorem 9.2.4.

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 it follows from the above that the matrix T in the Schur normal form is diagonal. If on the other hand A is unitarily diagonalizable then we immediately have that

$$A^H A = UD^H DU^H = UDD^H U^H = AA^H.$$

□

It follows in particular that any Hermitian matrix may be decomposed into

$$A = U\Lambda U^H = \sum_{i=1}^{n} \lambda_i u_i u_i^H. \tag{9.2.4}$$

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 (9.2.4) $u_i u_i^H$ 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 (9.2.1) 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.

9.2.2 The Jordan Canonical Form

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 9.2.1 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 9.2.5. Block Diagonal Decomposition.

Let the distinct eigenvalues of A be $\lambda_1, \cdots, \lambda_k$, and in the Schur normal form let $D = \operatorname{diag}(D_1, \ldots, D_k)$, $\quad D_i = \lambda_i I, \quad i = 1, \ldots, k$. Then there exists a nonsingular matrix Z such that

$$Z^{-1}U^H AUZ = Z^{-1}TZ = \operatorname{diag}(\lambda_1 I + N_1, \cdots, \lambda_k I + N_k),$$

where N_i, $i = 1, \ldots, k$ 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

$$M^{-1}TM = \begin{pmatrix} 1 & -m \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 & t \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda_1 & m(\lambda_1 - \lambda_2) + t \\ 0 & \lambda_2 \end{pmatrix}.$$

where M is an upper triangular elementary elimination matrix, see Section 7.3.5. By taking $m = t/(\lambda_2 - \lambda_1)$, we can annihilate the off-diagonal element in T.

In the general case let t_{ij} be an element in T outside the block diagonal. Let M_{ij} be a matrix which differs from the unit matrix only in the (i,j)th element, which is equal to m_{ij}. Then as above we can choose m_{ij} so that the element (i,j) is annihilated by the similarity transformation $M_{ij}^{-1}TM_{ij}$. Since T is upper triangular this transformation will not affect any already annihilated off-diagonal elements in T with indices (i',j') if $j' - i' < j - i$. Hence, we can annihilate all elements t_{ij} outside the block diagonal in this way, starting with the elements on the diagonal

closest to the main diagonal and working outwards. For example, in a case with 3 blocks of orders $2, 2, 1$ the elements are eliminated in the order

$$\begin{pmatrix} \times & \times & 2 & 3 & 4 \\ & \times & 1 & 2 & 3 \\ & & \times & \times & 2 \\ & & & \times & 1 \\ & & & & \times \end{pmatrix}.$$

Further details of the proof is left to the reader. □

A matrix which does not have n linearly independent eigenvectors is defective and cannot be similar to a diagonal matrix. We now state without proof the following fundamental Jordan Canonical Form. For a proof based on the block diagonal decomposition in Theorem 9.2.5, see Fletcher and Sorensen [8, 1983].

Theorem 9.2.6. 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

$$X^{-1}AX = J = \operatorname{diag}(J_{m_1}(\lambda_1), \cdots, J_{m_t}(\lambda_t)), \tag{9.2.5}$$

where

$$J_{m_i}(\lambda_i) = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix} = \lambda_i I + S \in \mathbf{C}^{m_i\times m_i}, \quad m_i \geq 1,$$

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 form (9.2.5) 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 9.2.1.

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, $i = 1, \ldots, m$, 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.

If $X = (x_1, x_2, \ldots, x_n)$ is the matrix in (9.2.5), then

$$Ax_1 = \lambda_1 x_1, \quad Ax_{i+1} = \lambda_1 x_{i+1} + x_i, \quad i = 1, \ldots, m_1 - 1.$$

The vectors $x_2, \ldots, x_{m_1}$ are called **principal vectors** of the matrix A. Similar relations holds for the other Jordan blocks.

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

$$q(z) = (z - \lambda_1)^{m_1} (z - \lambda_2)^{m_2} \cdots (z - \lambda_k)^{m_k}, \tag{9.2.6}$$

where m_j is the highest dimension of any Jordan box corresponding to the eigenvalue λ_j, $j = 1, \ldots, k$. As a corollary we obtain **Cayley-Hamilton theorem**, which states that the characteristic polynomial $p(z)$ of a matrix A satisfies $p(A) = 0$. The polynomials

$$\pi_i(z) = \det \left(zI - J_{m_i}(\lambda_i) \right) = (z - \lambda_i)^{m_i}$$

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 9.2.7.

Let $A \in \mathbf{C}^{n \times n}$ be a given matrix. Then for any $\epsilon > 0$ there exists a matrix B with $\|A - B\|_2 \leq \epsilon$, such that B has n distinct eigenvalues. Hence, the class of diagonalizable matrices is dense in $\mathbf{C}^{n \times n}$.

Proof. Let $X^{-1}AX = J$ be the Jordan canonical form of A. Then, by a slight extension of Example 9.2.1 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

$$\|A - B\|_2 \leq \epsilon, \quad \epsilon = \delta \|X\|_2 \|X^{-1}\|_2.$$

□

9.2.3 Sylvester's Matrix Equation

Given matrices $A \in \mathbf{R}^{n \times n}$, $B \in \mathbf{R}^{m \times m}$, and $C \in \mathbf{R}^{n \times m}$, consider the matrix equation

$$AX - XB = C, \qquad X \in \mathbf{R}^{n \times m}. \tag{9.2.7}$$

This equation, **Sylvester's equation**, is important in many areas, e.g., in control theory. We will investigate the existence and uniqueness of solutions to this equation using the Schur decomposition.

From Theorem 9.2.1 follows the existence of unitary matrices U_1 and U_2 such that

$$U_1^H A U_1 = T_1, \qquad U_2^H B U_2 = T_2,$$

where T_1 and T_2 are upper triangular. Then (9.2.7) can be reduced to

$$T_1 Y - Y T_2 = F, \qquad Y = U_1^H X U_2, \quad F = U_1^H C U_2.$$

We can now prove the following result.

Theorem 9.2.8.

The matrix equation (9.2.7) has a unique solution $X = U_1 Y U_2^H$ if and only if

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

Proof. The proof is by induction on n. For $n = 1$ the transformed equation has the form

$$\lambda_1 y_1^T - y_1^T T_2 = y_1^T(\lambda_1 I - T_2) = f_1^T,$$

where y_1^T and f_1^T are the first (and only) rows in Y and F, respectively. From the assumption it follows that $\lambda_1 I - T_2$ is nonsingular, hence this triangular system has a unique solution y_1. Now assume that the theorem is true for matrices of order less than n, and consider the following partitioned form of the equation

$$\begin{pmatrix} \lambda_1 & t_1^T \\ 0 & \hat{T}_1 \end{pmatrix} \begin{pmatrix} \sigma & y_1^T \\ z & \hat{Y} \end{pmatrix} - \begin{pmatrix} \sigma & y_1^T \\ z & \hat{Y} \end{pmatrix} \begin{pmatrix} \lambda_2 & t_2^T \\ 0 & \hat{T}_2 \end{pmatrix} = \begin{pmatrix} \beta & f_1^T \\ w & \hat{F} \end{pmatrix}.$$

Multiplying out and identifying the blocks of this matrix equation gives

$$\begin{aligned} (\hat{T}_1 - \lambda_2 I)z &= w, & (\lambda_1 - \lambda_2)\sigma + t_1^T z &= \beta, \\ \hat{T}_1 \hat{Y} - \hat{Y}\hat{T}_2 - z t_2^T &= F, & y_1^T(\lambda_1 I - \hat{T}_2) + t_1^T \hat{Y} - \sigma t_2^T &= f_1^T. \end{aligned}$$

These equations can be solved for $z, \sigma, \hat{Y}$, and y_1^T in this order, if and only if T_1 and T_2 satisfy the assumption. ∎

An important special case of (9.2.7) is the **Lyapunov equation**

$$AX + XA^H = C. \tag{9.2.8}$$

Here $B = -A^H$, and hence by Theorem 9.2.8 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. Also, 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.

9.2.4 Matrix-Valued Functions and Matrix Functions

We start with a definition of the limit of a sequence of matrices:

Definition 9.2.9.

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 9.2.10.

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.) □

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.

Example 9.2.2.

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

$$\frac{d}{dt}y(t) = Ay(t), \quad y(0) = c. \tag{9.2.9}$$

Other functions, for example, $\sin(z)$, $\cos(z)$, $\log(z)$, can similarly be defined for matrix arguments from their power series representation. In general, if $f(z) = \sum_{k=0}^{\infty} a_k z^k$ is an infinite power series we define $f(A) = \sum_{k=0}^{\infty} a_k A^k$.

We now turn to the question of how to define **analytic functions of matrices** in general. If the matrix A is diagonalizable then $A = X\Lambda X^{-1}$ and we define

$$f(A) = X\operatorname{diag}\big(f(\lambda_1), \ldots, f(\lambda_n)\big)X^{-1} = Xf(\Lambda)X^{-1}. \tag{9.2.10}$$

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 9.2.11.

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

$$f(A) = X\operatorname{diag}\Big(f\big(J_{m_1}(\lambda_1)\big), \cdots, f\big(J_{m_t}(\lambda_t)\big)\Big)X^{-1}. \tag{9.2.11}$$

the analytic function f of a Jordan block is

$$f(J_m) = f(\lambda)I + \sum_{p=1}^{m-1} \frac{1}{p!} f^{(p)}(\lambda) N^p. \tag{9.2.12}$$

One can show that for every non-singular matrix T it holds

$$f(T^{-1}AT) = T^{-1}f(A)T. \tag{9.2.13}$$

With this definition, the theory of analytic functions of *one* matrix variable closely follows the theory of *one* complex variable. If $\lim_{n\to\infty} f_n(z) = f(z)$ for $z \in D$, then $\lim_{n\to\infty} f_n(J(\lambda_i)) = f(J(\lambda_i))$, and hence $\lim_{n\to\infty} f_n(A) = f(A)$, *if the spectrum of A lies in the interior of D.* This allows us to deal with operations involving limit processes. The following important theorem can be obtained, which shows that Definition 9.2.11 is consistent with the more restricted definition (by a power series) given in Theorem 9.2.15.

Theorem 9.2.12.

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 9.2.3.

We have, for example,

$$\cos^2 A + \sin^2 A = I, \qquad \forall A;$$

$$\ln(I - A) = -\sum_{n=1}^{\infty} \frac{1}{n} A^n, \qquad \rho(A) < 1;$$

$$\int_0^{\infty} e^{-st} e^{At} dt = (sI - A)^{-1}, \qquad Re(\lambda_i) < Re(s);$$

Further, if $f(z)$ is analytic inside C, and if the whole spectrum of A is inside C, we have (cf. Problem 8)

$$\frac{1}{2\pi i} \int_C (zI - A)^{-1} f(z) dz = f(A).$$

Observe also that, for two arbitrary analytic functions f and g, which satisfy the condition of the definition, $f(A) \cdot g(A) = g(A) \cdot f(A)$. However, when several non-commutative matrices are involved, one can no longer use the usual formulas for analytic functions.

Example 9.2.4.

$e^{(A+B)t} = e^{At} e^{Bt}$ 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.

9.2.5 Convergence of Matrix Power 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 11.

We recall the definitions:

Definition 9.2.13. *The* **spectral radius** *of A is the largest modulus of an eigenvalue of a matrix A*

$$\rho(A) = \max_i |\lambda_i(A)|. \tag{9.2.14}$$

Theorem 9.2.14.

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:

(a) *If A has no defective eigenvalues with absolute value ρ then there exists a nonsingular matrix T such that*

$$\|A\|_T = \rho.$$

(b) *If A has a defective eigenvalue with absolute value ρ then for every $\epsilon > 0$ there exists a nonsingular matrix $T(\epsilon)$ such that*

$$\|A\|_{T(\epsilon)} \le \rho + \epsilon.$$

In this case, the condition number $\kappa(T(\epsilon)) \to \infty$ like ϵ^{1-m^*} as $\epsilon \to 0$, where $m^* > 1$ is the largest order of a Jordan block belonging to an eigenvalue λ with $|\lambda| = \rho$.

Proof. If A is diagonalizable, we can simply take T as the diagonalizing transformation. Then clearly $\|A\|_T = \|D\| = \rho$, where $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. In the general case, we first bring A to Jordan canonical form, $X^{-1}AX = J$, where

$$J = \operatorname{diag}\big(J_1(\lambda_1), \cdots, J_t(\lambda_t)\big), \quad J_i(\lambda_i) = \lambda_i I + N_i \in \mathbf{C}^{m_i\times m_i}, \quad m_i \ge 1,$$

and $J_i(\lambda_i)$ is a Jordan block. We shall find a diagonal matrix $D = \operatorname{diag}(D_1, \ldots, D_t)$, 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

$$K = \operatorname{diag}(K_1, K_2, \ldots, K_t), \quad K_i = D_i^{-1} J_i(\lambda_i) D_i.$$

If $m_i = 1$, we set $D_i = 1$, hence $\|K_i\| = |\lambda_i|$. Otherwise we choose

$$D_i = \operatorname{diag}\left(1, \delta_i, \delta_i^2, \ldots, \delta_i^{m_i-1}\right), \quad \delta_i > 0. \tag{9.2.15}$$

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,

$$\|K_i\| \le |\lambda_i| + \delta_i. \tag{9.2.16}$$

If $m_i > 1$ and $|\lambda_i| < \rho$, we choose $\delta_i = \rho - |\lambda_i|$, hence $\|K_i\| \le \rho$. This proves case (a).

In case (b), $m_i > 1$ for at least one eigenvalue with $|\lambda_i| = \rho$. Let $M = \{i : |\lambda_i| = \rho\}$, and choose $\delta_i = \epsilon$, for $i \in M$. Then by (9.2.16) $\|K_i\| \le \rho + \epsilon$, for $i \in M$, while $\|K_i\| \le \rho$, for $i \notin M$. Hence $\|K\| = \max_i \|K_i\| = \rho + \epsilon$, and the first part of statement (b) now follows.

With $T(\epsilon) = XD(\epsilon)$, we have that

$$\kappa\big(D(\epsilon)\big)/\kappa(X) \le \kappa\big(T(\epsilon)\big) \le \kappa\big(D(\epsilon)\big)\kappa(X).$$

When $|\lambda_i| = \rho$ we have $\delta_i = \epsilon$, and it follows from (9.2.15) that $\kappa(D_i)$ grows like ϵ^{1-m_i}. Since $\kappa(D) = \max_i \kappa(D_i)$, and for $|\lambda_i| < \rho$ the condition numbers of D_i are bounded, this proves the second part of statement (b). □

Note that $1/\kappa(T) \le \|A\|_T/\|A\| \le \kappa(T)$. For every natural number n, we have, in case (a), $\|A^n\|_T \le \|A\|_T^n = \rho(A)^n$. Hence

$$\|A^n\|_p \le \kappa(T)\|A^n\|_T \le \kappa(T)\rho^n.$$

In case (b), the same holds, if ρ, T are replaced by, respectively, $\rho+\epsilon, T(\epsilon)$. See also Problem 8.

If only statement (b) is needed, a more elementary proof can be found by a similar argument applied to the Schur canonical form instead of the Jordan canonical form. Since X is unitary in this case, one has a better control of the condition numbers, which is of particular importance in some applications to partial differential equations, where one needs to apply this kind of theorem to a *family of matrices* instead of just one individual matrix. This leads to the famous *matrix theorems of Kreiss*, see Theorems 13.8.6–13.8.7.

We now return to the question of convergence of matrix series.

Theorem 9.2.15.

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 9.2.10 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 9.2.14 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. □

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 9.2.14

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 $T = \operatorname{diag}(w_1, \ldots, w_n) > 0$, and $\kappa(T) = \max w_i / \min w_i$. We then note that if we can find a positive vector w such that $|A|w \leq \alpha w$, then $\|A\|_w \leq \alpha$.

If A is diagonalizable, $A = X^{-1}\Lambda X$, then for the exponential function we have,

$$\|e^{tA}\|_2 = \kappa(X)e^{t\alpha(A)},$$

where $\alpha(A)$ 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 $\kappa(V) = 1$.

Many physical, biological, and economic processes involve systems of linear constant coefficients ordinary differential equations $dy(t)/dt = Ay(t)$, $y(0) = c$, with solution $y(t) = e^{tA}c$. Hence the computation of the matrix exponential function, and investigation of its qualitative behavior is an important task that has been studied extensively. It is in general not an easy task; see Moler and Van Loan [24].

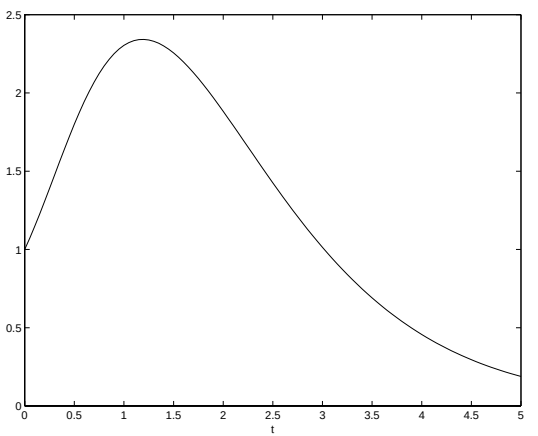


Figure 9.2.1. $\|e^{tA}\|$ *as a function of* t *for the matrix in Example 9.2.5.*

Example 9.2.5. The matrix

$$A = \begin{pmatrix} -1 & 4 \\ 0 & -2 \end{pmatrix},$$

has the spectral abscissa $\alpha(A) = -1$. It follows that $\lim_{t\to\infty} e^{tA} = 0$. In Figure 9.2.1 we have plotted $\|e^{tA}\|$ as a function of t. The curve has a *hump* illustrating that as t increases the elements in e^{tA} first increase before they start to decay. When the same type of difficulty occurs in non-tringular problems of larger size the cure is by no means easy!

9.2.6 Non-Negative Matrices

Non-negative matrices arise in many applications and play an important role in, e.g., queuing theory and analysis of iterative methods. If a matrix A is irreducible (Def. 9.1.4) then $|A|$ is a *non-negative and irreducible matrix*. For such matrices the following classical theorem of Perron and Frobenius holds.

Theorem 9.2.16. (Perron–Frobenius Theorem)
If A is an irreducible nonnegative matrix then A has a positive real eigenvalue r with the properties:

(i) *to r corresponds a positive eigenvector $x > 0$.*

(ii) *if λ is any other eigenvalue of A, then $|\lambda| \leq r$. and hence $r = \rho(A)$. The eigenvalues of modulus $\rho(A)$ are all simple roots of the characteristic equation. If there are m eigenvalues of modulus ρ, they must be of the form*

$$\lambda_k = \rho e^{\frac{2k\pi i}{m}}, \quad k = 0, 1, \ldots, m-1.$$

Proof. See, e.g., Gantmacher [11, 1959], Vol. II. A simpler proof of some of these results is found in Strang [35, 1988, [p. 271]]. □

This theorem shows that for the matrix $|A|$ *the norm $\|\cdot\|_w$ is efficient, where $w = x$ is the positive eigenvector corresponding to the eigenvalue $r = \rho(|A|)$.* (This norm may be useful also for A itself.) This result can be extended to some reducible matrices, e.g., to *any upper triangular matrix A such that $|a_{ii}| < |a_{nn}|$ for all $i < n$.* Then $\rho(A) = \rho(|A|) = |a_{nn}|$. A positive vector w such that $Aw \leq \rho(A)w$ can be found by solving the inequalities

$$(|a_{nn}| - |a_{ii}|)w_i \geq \sum_{j,\ j>i} |a_{ij}|w_j, \quad i = n-1, n-2, \ldots, 1.$$

In fact, it is in most cases possible to use equality for every i, but $\kappa(T)$ may become smaller if one chooses a larger w_i for some i. For example, consider a very strongly diagonally dominant matrix of this type. The solution obtained with equality everywhere would make T very ill-conditioned, but the unweighted max-norm ($\kappa(T) = 1$) is also efficient (and can be obtained with strict inequality everywhere). In Section 13.1 and Section 13.7 there will be applications of this to differential equations and to logarithmic norms.

Review Questions

1. How can the class of matrices which are diagonalizable by unitary transformations be characterized?
2. What is meant by a defective eigenvalue? Give a simple example of a matrix with a defective eigenvalue.

3. Define the matrix function e^A. Show how this can be used to express the solution to the initial value problem $y'(t) = Ay(t)$, $y(0) = c$?

4. 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 7.)

5. (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. Show that

$$\lim_{t\to\infty} \frac{1}{t} \ln \|e^{At}\| = \max_{\lambda\in\lambda(A)} \mathrm{Re}(\lambda), \qquad \lim_{t\to 0} \frac{1}{t} \ln \|e^{At}\| = \mu(A).$$

7. 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 9.2.8).

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?

9. Under what conditions can identities which hold for analytic functions of complex variable(s) be generalized to analytic functions of matrices?

Problems

1. 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?

2. (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 (9.2.13). Also construct an example, where $p(A)$ has other eigenvectors in addition to those of A.

3. Show that the series expansion

$$(I - A)^{-1} = I + A + A^2 + A^3 + \ldots$$

converges if $\rho(A) < 1$.

4. (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.

5. 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 $Q^TAQ = \text{diag}\,(\lambda_i) + N$ are the same.

6. Show that Sylvester's equation (9.2.7) 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 (9.1.15) to give an independent proof that Sylvester's equation has a unique solution if and only if $\lambda_i - \mu_j \neq 0$, $i = 1,\ldots,n$, $j = 1,\ldots,m$.

7. Show that

$$e^A \otimes e^B = e^{B\oplus A},$$

where $\oplus$ denotes the Kronecker sum.

8. (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 $\ln(A) = \begin{pmatrix} -0.6931 & 1.8232 \\ 0 & 0.5108 \end{pmatrix}$.

9. Show that an analytic function of the matrix A can be computed by Newton's interpolation formula, i.e.,

$$f(A) = f(\lambda_1)I + \sum_{j=1}^{n^*} f(\lambda_1,\lambda_2,\ldots,\lambda_j)(A-\lambda_1 I)\cdots(A-\lambda_j I)$$

where λ_j, $j = 1,2,\ldots,n^*$ 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.

10. We use the notation of Theorem 9.2.14. For a given n, show by an appropriate choice of ϵ that $\|A^n\|_p \leq Cn^{m^*-1}\rho^n$, where C is independent of n. Then derive the same result from the Jordan Canonical form.

Hint: See the comment after Theorem 9.2.14.

11. 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 Example 9.2.3, $\phi_C(A) = I$. 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.

9.3 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.

9.3.1 Gershgorin's Theorems

We first derive a sequence of powerful theorems, which can be used both to locate eigenvalues of a matrix and to derive perturbation results.

Theorem 9.3.1.

All the eigenvalues of the matrix $A \in \mathbf{C}^{n\times n}$ lie in the union of the **Gershgorin disks** *in the complex plane*

$$\mathcal{D}_i = \{z \mid |z - a_{ii}| \leq r_i\}, \qquad r_i = \sum_{j=1, j\neq i}^{n} |a_{ij}|, \qquad i = 1, 2, \ldots, n. \tag{9.3.1}$$

Proof. If λ is an eigenvalue there is an eigenvector $x \neq 0$ such that $Ax = \lambda x$, or

$$(\lambda - a_{ii})x_i = \sum_{j=1, j\neq i}^{n} a_{ij}x_j, \quad i = 1, \ldots, n.$$

Choose i so that $|x_i| = \|x\|_\infty$. Then

$$|\lambda - a_{ii}| \leq \sum_{j=1, j\neq i}^{n} \frac{|a_{ij}||x_j|}{|x_i|} \leq r_i. \tag{9.3.2}$$

□

The Gershgorin 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 also to A^T .

From (9.3.2) it follows that if the ith component of the eigenvector is maximal, then λ lies in the ith disk. Otherwise the Gershgorin theorem does not say in *which* disks the eigenvalues lie. Sometimes it is possible to decide this as the following theorem shows.

Theorem 9.3.2.

If the union $\mathcal{M}$ of k Gershgorin 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

$$A(t) = tA + (1 - t)D_A, \quad D_A = diag(a_{ii}).$$

The coefficients in the characteristic polynomial are continuous functions of t, and hence also the eigenvalues $\lambda(t)$ of $A(t)$ are 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 also k eigenvalues of $A = A(1)$ lie in $\mathcal{M}$. ∎

Example 9.3.1.

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 Gershgorin disks

$$\mathcal{D}_1 = \{z \mid |z - 2| \leq 0.15\}; \quad \mathcal{D}_2 = \{z \mid |z - 1| \leq 0.3\}; \quad \mathcal{D}_3 = \{z \mid |z - 1| \leq 0.15\}.$$

Since the disk $\mathcal{D}_1$ is disjoint from the rest of the disks, it must contain precisely one eigenvalue of A. The remaining two eigenvalues must lie in $\mathcal{D}_2 \cup \mathcal{D}_3 = \mathcal{D}_2$.

There is another useful sharpening of Gershgorin's Theorem in case the matrix A is irreducible, cf. Def. 9.1.4.

Theorem 9.3.3.

If A is irreducible then each eigenvalue λ lies in the **interior** *of the union of the Gershgorin disks, unless it lies on the boundary of* **all** *Gershgorin disks.*

Proof. If λ lies on the boundary of the union of the Gershgorin disks, then we have

$$|\lambda - a_{ii}| \geq r_i, \quad \forall i. \tag{9.3.3}$$

Let x be a corresponding eigenvector and assume that $|x_{i_1}| = \|x\|_\infty$. Then from the proof of Theorem 9.3.1 and (9.3.3) it follows that $|\lambda - a_{i_1 i_1}| = r_{i_1}$. But (9.3.2) implies that equality can only hold here if for any $a_{i_1 j} \neq 0$ it holds that $|x_j| = \|x\|_\infty$. If we assume that $a_{i_1,i_2} \neq 0$ then it follows that $|\lambda - a_{i_2 i_2}| = r_{i_2}$. But since A is irreducible for any $j \neq i$ there is a path $i = i_1, i_2, \ldots, i_p = j$. It follows that λ must lie on the boundary of all Gershgorin disks. □

Example 9.3.2. 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 Gershgorin disks are

$$|z-2| \leq 2, \quad i = 2, \ldots, n-1, \qquad |z-2| \leq 1, \quad i = 1, n,$$

and it follows that all eigenvalues of A satisfy $\lambda \geq 0$. Since zero is on the boundary of the union of these disks, but *not* on the boundary of all disks, zero cannot be an eigenvalue of A. Hence all eigenvalues are *strictly* positive and A is positive definite.

9.3.2 Perturbation Theorems for Eigenvalues and Eigenvectors

In the rest of this section we consider the sensitivity of eigenvalue and eigenvectors to perturbations.

Theorem 9.3.4. (Bauer–Fike.)
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

$$\min_{1\leq i\leq n} |\mu - \lambda_i| \leq \kappa_p(X)\|E\|_p. \tag{9.3.4}$$

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 (9.3.4) 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 $z \neq 0$ such that

$$(D - \mu I)z = -X^{-1}EXz.$$

Solving for z and taking norms we obtain

$$\|z\|_p \leq \kappa_p(X)\|(D - \mu I)^{-1}\|_p \|E\|_p \|z\|_p.$$

The theorem follows by dividing by $\|z\|_p$ and using the fact that for any p-norm $\|(D-\mu I)^{-1}\|_p = 1/\min_{1\le i\le n}|\lambda_i-\mu|$. ∎

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 9.2.1) 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 9.2.1, and the following example.

Example 9.3.3.

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 9.3.5.

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,

$$\lambda_j(\epsilon) = \lambda_j + \epsilon \frac{y_j^H E x_j}{y_j^H x_j} + O(\epsilon^2). \tag{9.3.5}$$

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 9.3.2 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

$$(A - \lambda_j I)x_j'(0) + Ex_j = \lambda_j'(0)x_j. \tag{9.3.6}$$

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 $|y_j^H E x_j| \leq \|x_j\|_2 \|y_j\|_2$ 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

$$s(\lambda_j) = |y_j^H x_j| \tag{9.3.7}$$

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 (9.3.7).

Example 9.3.4.

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

$$x_j(\epsilon) = x_j + \epsilon x_j'(0) + O(\epsilon^2), \qquad x_j'(0) = \sum_{k \neq j} c_{kj} x_k,$$

where we have normalized $x_j(\epsilon)$ to have unit component along x_j. Substituting the expansion of $x_j'(0)$ into (9.3.6) we get

$$\sum_{k\neq j} c_{kj}(\lambda_k - \lambda_j)x_k + Ex_j = \lambda_j'(0)x_j.$$

Multiplying by y_i^H and using $y_i^H x_j = 0$, $i \neq j$, we obtain

$$c_{ij} = \frac{y_i^H E x_j}{(\lambda_j - \lambda_i) y_i^H x_i}, \quad i \neq j. \tag{9.3.8}$$

Hence, the sensitivity of the eigenvectors also depend 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 9.3.6. *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*

$$\theta_{\max}(\mathcal{X}, \mathcal{Y}) = \max_{\substack{x\in\mathcal{X} \\ \|x\|_2=1}} \min_{\substack{y\in\mathcal{Y} \\ \|y\|_2=1}} \theta(x, y). \tag{9.3.9}$$

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 [16]) that

$$\theta(\mathcal{X}, \mathcal{Y}) = \arccos\sigma_{min}(X^H Y). \tag{9.3.10}$$

9.3.3 Hermitian Matrices

We have seen that the eigenvalues of Hermitian, and real symmetric matrices are all real, and from Theorem 9.3.5 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 (minimax) characterization also of the intermediate eigenvalues of a Hermitian matrix.

Theorem 9.3.7. Fischer's Theorem.

Let the Hermitian matrix A have eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ ordered so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Then

$$\lambda_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H A x}{x^H x} = \min_{\dim(\mathcal{S})=n-i+1} \max_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H A x}{x^H x}. \tag{9.3.11}$$

where $\mathcal{S}$ denotes a subspace of $\mathbf{C}^n$.

Proof. See Stewart [32, 1973, p. 314]. □

The above characterization of the eigenvalues may be used to establish an important relation between the eigenvalues of Hermitian matrices A, B, and $C = A + B$.

Theorem 9.3.8.

Let $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n$, $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$, and $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_n$ be the eigenvalues of the Hermitian matrices A, B, and $C = A + B$. Then

$$\alpha_i + \beta_1 \geq \gamma_i \geq \alpha_i + \beta_n, \quad i = 1, 2, \ldots, n. \tag{9.3.12}$$

Proof. Let $x_1, x_2, \ldots, x_n$ be an orthonormal system of eigenvectors of A corresponding to $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n$, and let $\mathcal{S}$ be the subspace of $\mathbf{C}^n$ spanned by $x_1, \ldots, x_i$. Then by Fischer's theorem

$$\gamma_i \geq \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H C x}{x^H x} \geq \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H A x}{x^H x} + \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H B x}{x^H x} = \alpha_i + \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H B x}{x^H x} \geq \alpha_i + \beta_n.$$

This is the last inequality of (9.3.12). 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 $\operatorname{rank}(B) < n$, the result can be sharpened, see Parlett [41, Section 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

$$\lambda_i' - \lambda_i = m_i \rho, \quad 0 \leq m_i, \quad \sum m_i = 1. \tag{9.3.13}$$

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 Hermitian matrix $A + B$, when the matrix B is a perturbation to A. Usually the eigenvalues of B are not known, but from

$$\max\{|\lambda_1(B)|, |\lambda_n(B)|\} = \rho(B) = \|B\|_2$$

it follows that

$$|\lambda_i - \lambda_i'| \le \|B\|_2. \tag{9.3.14}$$

A related result is the **Wielandt–Hoffman theorem** which states that

$$\sum_{i=1}^{n} |\lambda_i - \lambda_i'|^2 \le \|B\|_F^2. \tag{9.3.15}$$

An elementary proof of this result is given by Wilkinson [40, Section 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 9.3.9. 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

$$\lambda_i \ge \mu_i \ge \lambda_{i+1}, \quad i = 1, \ldots, n-1. \tag{9.3.16}$$

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 (9.3.11) of the eigenvalue λ_i it follows that

$$\lambda_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x\in\mathcal{S}\\ x\neq 0}} \frac{x^H A_n x}{x^H x} \ge \max_{\substack{dim(\mathcal{S})=i\\ \mathcal{S}\perp e_n}} \min_{\substack{x\in\mathcal{S}\\ x\neq 0}} \frac{x^H A_n x}{x^H x} = \mu_i.$$

The proof of the second inequality $\mu_i \ge \lambda_{i+1}$ is obtained by a similar argument applied to $-A_n$. ∎

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.

Hermitian matrices arise naturally in the study of quadratic forms $\psi(x) = x^H A x$. By the coordinate transformation $x = Ty$ this quadratic form is transformed into

$$\psi(Ty) = y^H \hat{A} y, \quad \hat{A} = T^H A T.$$

The mapping of A onto $T^H A T$ is called a **congruence transformation** of A, and we say that A and $\hat{A}$ are **congruent**. 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 9.3.10. Sylvester's Law of Inertia.

Let A be Hermitian and define the **inertia** *of A to be the number triple $in(A) = (\pi, \nu, \delta)$ of positive, negative, and zero eigenvalues of A. Then if T is nonsingular A and $\hat{A} = T^H A T$ have the same inertia.*

Proof. Since A and $\hat{A}$ are Hermitian there exist unitary matrices U and $\hat{U}$ such that

$$U^H A U = D, \qquad \hat{U}^H \hat{A} \hat{U} = \hat{D},$$

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

$$\hat{D} = S^H D S, \quad S = U^H T \hat{U}.$$

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 $\delta = \hat{\delta}$. □

9.3.4 Rayleigh quotient and residual bounds

We make the following definition.

Definition 9.3.11.

The **Rayleigh quotient** *of a nonzero vector $x \in \mathbf{C}^n$ is the (complex) scalar*

$$\rho(x) = \rho(A, x) = \frac{x^H A x}{x^H x}. \tag{9.3.17}$$

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 9.3.12.

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 $F(U^H AU) = F(A)$. From the Schur normal 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 $x^H x\mu = x^H Ax$. 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

$$\frac{1}{2}\nabla\rho(x) = \frac{Ax}{x^H x} - \frac{x^H Ax}{(x^H x)^2}x = \frac{1}{x^H x}(Ax - \rho x),$$

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 9.3.13.

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

$$E = -r\bar{x}^H, \qquad \|E\|_2 = \|r\|_2. \tag{9.3.18}$$

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 9.3.13.

By combining Theorems 9.3.4 and 9.3.13 we obtain for a Hermitian matrix A the very useful a posteriori error bound

Corollary 9.3.14. *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*

$$|\lambda - \bar{\lambda}| \le \|r\|_2, \qquad r = A\bar{x} - \bar{\lambda}\bar{x}. \tag{9.3.19}$$

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 9.3.15.

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

$$\|x - \xi_1 x_1\|_2 \le \epsilon \|x\|_2. \tag{9.3.20}$$

then

$$|\rho(x) - \lambda_1| \le 2\|A\|_2 \epsilon^2. \tag{9.3.21}$$

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 (9.3.20) we get $|\rho(x) - \lambda_1| \le \max_i |\lambda_i - \lambda_1| \epsilon^2$. Since the matrix A is Hermitian we have $|\lambda_i| \le \sigma_1(A) = \|A\|_2$, $i = 1, \ldots, n$, 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 9.3.16.

Let A be a Hermitian matrix with eigenvalues $\lambda(A) = \{\lambda_1, \ldots, \lambda_n\}$, 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

$$\text{gap}\,(\rho) = \min_{\lambda \in \lambda(A)} |\lambda - \rho|, \quad \lambda \ne \lambda_\rho. \tag{9.3.22}$$

Then it holds that

$$|\lambda_\rho - \rho(x)| \le \|Ax - x\rho\|_2^2/\text{gap}\,(\rho), \tag{9.3.23}$$

$$\sin\theta(x,z) \le \|Ax - x\rho\|_2/\text{gap}\,(\rho). \tag{9.3.24}$$

Proof. See Parlett [41, Section 11.7]. □

Example 9.3.5.

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 9.3.14 we get $|\lambda - 1| \le \epsilon$, whereas Theorem 9.3.16 gives the improved bound $|\lambda - 1| \le \epsilon^2/(1-\epsilon^2)$.

Often gap(σ) is not known and the bounds in Theorem 9.3.16 are only theoretical. In some methods, e.g., the method of spectrum slicing (see Section 9.5.4) an interval around σ can be determined which contain no eigenvalues of A.

9.3.5 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[2]

$$C = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} U & U \\ V & -V \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} U & U \\ V & -V \end{pmatrix}^T \tag{9.3.25}$$

Using these relationships the theory developed for the symmetric (Hermitian) eigenvalue problem in Secs. 9.3.3–9.3.4 applies also to the singular value decomposition. For example, Theorems 8.3.3–8.3.5 are straightforward applications of Theorems 9.3.7–9.3.9.

We now consider applications of the Rayleigh quotient and residual error bounds given in Section 9.3.4. If u, v are unit vectors the Rayleigh quotient of C is

$$\rho(u,v) = \frac{1}{\sqrt{2}}(u^T, v^T) \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} u \\ v \end{pmatrix} = u^TAv, \tag{9.3.26}$$

From Corollary 9.3.14 we obtain the following error bound.

[2] This assumption is no restriction since we can always adjoin zero rows (columns) to make A square.

Theorem 9.3.17. *For any scalar α and unit vectors u, v there is a singular value σ of A such that*

$$|\sigma - \alpha| \le \frac{1}{\sqrt{2}} \left\| \begin{pmatrix} Av - u\alpha \\ A^T u - v\alpha \end{pmatrix} \right\|_2. \tag{9.3.27}$$

For fixed u, v this error bound is minimized by taking $\alpha = u^T A v$.

The following theorem is an application to Theorem 9.3.16.

Theorem 9.3.18.

Let A have singular values σ_i, $i = 1, \ldots, n$. 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

$$|\sigma_s - \rho(x)| \le \delta^2 / \mathrm{gap}\,(\rho), \tag{9.3.28}$$

$$\max\{\sin\theta(u_s, u), \sin\theta(v_s, v)\} \le \delta / \mathrm{gap}\,(\rho). \tag{9.3.29}$$

where

$$\mathrm{gap}\,(\rho) = \min_{i \ne s} |\sigma_i - \rho|. \tag{9.3.30}$$

Review Questions

1. State Gershgorin's Theorem, and discuss how it can be sharpened.
2. Discuss the sensitivity to perturbations of eigenvalues and eigenvectors of a Hermitian matrix A.
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?
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 $\mathbf{R}^{(n-1)\times(n-1)}$.
5. Formulate and prove Sylvester's law of inertia.
6. The equation of a quadric surface in $\mathbf{R}^3$ with a center at the origin reads $x^T A x = c$. How can you tell the type of surface without computing the eigenvalues? (Notice that a congruence transformation with a nonsingular matrix means a transformation to a coordinate system which is usually not rectangular.)

Problems

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

$$\mathrm{Re}(a_{ii}) + r_i \leq 0, \quad \forall i,$$

and $\mathrm{Re}(a_{ii}) + r_i < 0$ for at least one i, then the matrix A is stable if A is irreducible.

2. Suppose that the matrix A is real, and all Gershgorin discs of A are distinct. Show that from Theorem 9.3.2 it follows that all eigenvalues of A are real.

3. Show that all eigenvalues to a matrix A lie in the union of the disks

$$|z - a_{ii}| \leq \frac{1}{d_i} \sum_{j=1, j\neq i}^{n} d_j |a_{ij}|, \qquad i = 1, 2, \ldots, n,$$

where d_i, $i = 1, 2, \ldots, n$ are given positive scale factors.

Hint: Use the fact that the eigenvalues are invariant under similarity transformations.

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 Gershgorin 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 Gershgorin disk is given by

$$|\lambda - a_{11}| \leq r_1, \quad r_1 \leq (n-1)\epsilon^2/\delta + O(\epsilon^3).$$

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.

6. Use a suitable diagonal similarity and Gershgorin'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

$$|\lambda - a| < 2\sqrt{\max_i |b_i| \max_i |c_i|}.$$

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

$$|\lambda(H) - \lambda(B)| \leq (\|C^H C\|_2)^{1/2}.$$

Hint: Use the estimate (9.3.19).

8. (a) Let $D = \operatorname{diag}(d_i)$ and $z = (z_1, \ldots, z_n)^T$. Show that if $\lambda \neq d_i$, $i = 1, \ldots, n$, then

$$\det(D + \mu z z^T - \lambda I) = \det\big((D - \lambda I)(I + (D - \lambda I)^{-1} \mu z z^T)\big).$$

Using the identity $\det(I + xy^T) = 1 + y^T x$ conclude that the eigenvalues λ of $D + \mu z z^T$ are the roots of the **secular equation**

$$f(\lambda) = 1 + \mu \sum_{i=1}^{n} \frac{z_i^2}{d_i - \lambda} = 0.$$

(b) Show by means of Fischer's Theorem 9.3.8 that the eigenvalues λ_i interlace the elements d_i so that if , for example, $\mu \geq 0$ then

$$d_1 \leq \lambda_1 \leq d_2 \leq \lambda_2 \leq \cdots \leq d_n \leq \lambda_n.$$

9.4 Jacobi Methods

The method of Jacobi (which dates back to 1846!) was before the advent of the QR algorithm the method of choice for the numerical solution of the eigenvalue problem for a real symmetric (or Hermitian) matrix. It is still an important method, partly because of its suitability for implementation on parallel computers. Moreover, it is an efficient method for computing eigenvalues of a *nearly diagonal matrix*. Both the QR algorithm (see Section 9.7) and Jacobi's algorithm are backward stable methods, and thus compute large eigenvalues (those near $\|A\|_2$ in magnitude) with high relative accuracy. However, Jacobi's method may compute *small eigenvalues with better relative accuracy*, than the algorithms which first reduce the matrix to tridiagonal form.

9.4.1 Jacobi Methods for Real Symmetric Matrices

The Jacobi method solves the eigenvalue problem for a real symmetric matrix $A \in \mathbf{R}^{n \times n}$ by solving a sequence of 2×2 subproblems. It employs a sequence of similarity transformations

$$A_0 = A, \qquad A_{k+1} = J_k^T A_k J_k \tag{9.4.1}$$

such that the sequence of matrices A_k, $k = 1, 2, \ldots$ 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. 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

$$\begin{pmatrix} l_p & 0 \\ 0 & l_q \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}. \tag{9.4.2}$$

Equating the off-diagonal elements we obtain (as $a_{pq} = a_{qp}$)

$$0 = (a_{pp} - a_{qq})cs + a_{pq}(c^2 - s^2), \tag{9.4.3}$$

which shows that the angle θ satisfies

$$\tau \equiv \cot 2\theta = (a_{qq} - a_{pp})/(2a_{pq}), \qquad a_{pq} \neq 0. \tag{9.4.4}$$

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**.

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 (9.4.2) is best computed by the following algorithm.

Algorithm 9.4.1
Jacobi transformation matrix ($a_{pq} \neq 0$):

$$\begin{aligned} &[c, s, l_p, l_q] = \text{jacobi}(a_{pp}, a_{pq}, a_{qq}) \\ &\quad \tau = (a_{qq} - a_{pp})/(2a_{pq}); \\ &\quad t = \text{sign}\,(\tau)/(|\tau| + \sqrt{1 + \tau^2}); \\ &\quad c = 1/\sqrt{1 + t^2}; \quad s = tc; \\ &\quad l_p = a_{pp} - t a_{pq}; \\ &\quad l_q = a_{qq} + t a_{pq}; \\ &\quad \text{end} \end{aligned}$$

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 [29, 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 machine unit u. From the Bauer–Fike Theorem 9.3.4 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 [9, 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

$$\begin{array}{cccc} (1,2), & (1,3), & \dots & (1,n) \\ & (2,3), & \dots & (2,n) \\ & & \dots & \dots \\ & & & (n-1,n) \end{array} \tag{9.4.5}$$

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 $\mathrm{O}(\log n)$, 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

$$X_k = X_{k-1} J_k, \quad k = 1, 2, \dots \tag{9.4.6}$$

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

$$(p,q) = \begin{array}{cccc} (1,2), & (3,4), & (5,6), & (7,8) \\ (1,4), & (2,6), & (3,8), & (5,7) \\ (1,6), & (4,8), & (2,7), & (3,5) \\ (1,8), & (6,7), & (4,5), & (2,3) \\ (1,7), & (8,5), & (6,3), & (4,2) \\ (1,5), & (7,3), & (8,2), & (6,4) \\ (1,3), & (5,2), & (7,4), & (8,6) \end{array}.$$

The rotations associated with *each row* of the above 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.

9.4.2 Jacobi Methods for the SVD.

Before the advent of the QR algorithm two different Jacobi-type methods for computing the SVD were developed. In **Kogbetliantz's method** the "norm" of the

off-diagonal elements in A is successively reduced by a sequence of two-sided Givens transformations. **Hestenes method** uses one-sided Givens transformations. These methods have several features which have made them very popular algorithms for implementation on machines with parallel architectures. These two Jacobi-type algorithms are slower than the QR algorithm, but in some cases compute the smaller singular values more accurately than any algorithm based on the bidiagonal reduction.

In Hestenes' method one-sided 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. In Hestenes' original algorithm the columns are explicitly interchanged so that the final columns of AV appear in order of decreasing norm. The basic step rotates two columns:

$$(\hat{a}_p, \hat{a}_q) = (a_p, a_q)\begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \qquad p < q. \tag{9.4.7}$$

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. However, this approach may lead to numerical problems since we are squaring part of the matrix. To determine the rotation it is better to first compute the QR factorization

$$(a_p, a_q) = (q_1,\ q_2)\begin{pmatrix} r_{pp} & r_{pq} \\ 0 & r_{qq} \end{pmatrix} \equiv QR.$$

If now the 2×2 SVD $R = U\Sigma V^T$ is computed, using one of the algorithm given below, 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 (9.4.7).

A normwise backward stable algorithm for computing the SVD of an upper triangular 2×2 matrix

$$\begin{pmatrix} c_u & s_u \\ -s_u & c_u \end{pmatrix}^T \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix} \begin{pmatrix} c_v & s_v \\ -s_v & c_v \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \tag{9.4.8}$$

is outlined in Golub and Van Loan [16, Problem 8.4.1]. In the first step a Givens rotation is determined to symmetrize the matrix R. In the second step the symmetric matrix B is diagonalized by a Jacobi transformation. The alternative algorithm below, which 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.

Algorithm 9.4.2
SVD of 2×2 upper triangular matrix (9.4.8) with $|r_{11}| \geq |r_{22}|$:

$$
\begin{aligned}
&[c_u, s_u, c_v, s_v, \sigma_1, \sigma_2] = \mathrm{svd}(r_{11}, r_{12}, r_{22}) \\
&\quad l = (|r_{11}| - |r_{22}|)/|r_{11}|; \\
&\quad m = r_{12}/r_{11}; \quad t = 2 - l; \\
&\quad s = \sqrt{t^2 + m^2}; \quad r = \sqrt{l^2 + m^2}; \\
&\quad a = 0.5(s + r); \\
&\quad \sigma_1 = |r_{11}|a; \quad \sigma_2 = |r_{22}|/a; \\
&\quad t = (1 + a)(m/(s + t) + m/(r + l)); \\
&\quad l = \sqrt{t^2 + 4}; \\
&\quad c_v = 2/l; \quad s_v = -t/l; \\
&\quad c_u = (c_v - s_v m)/a; \quad s_u = s_v(r_{22}/r_{11})/a; \\
&\quad \text{end}
\end{aligned}
$$

Note that the SVD produced by Hestenes' method will by construction have U orthogonal to working accuracy. However, loss of orthogonality in V may occur, and the columns of V should be reorthogonalized using a Gram–Schmidt process at the end.

Clearly, Hestenes' 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. There are various strategies for choosing the order in which the off-diagonal elements are annihilated. 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 rate of convergence is ultimately quadratic,

Hestenes' method works on general real (or complex) matrices $A \in \mathbf{R}^{m \times n}$, $m \geq n$. In the following we can assume without restriction that $m = n$. If $m > n$ we first compute the QR decomposition of A and apply the algorithm to the upper triangular matrix $R \in \mathbf{R}^{n \times n}$. Indeed, an initial QR decomposition of A can be recommended also when $m = n$, since it tends to speed up convergence and simplifies the transformations.

In Kogbetliantz's method applied to a square matrix A the elementary step consists of two-sided Givens transformations

$$A' = J_{pq}(\phi) A J_{pq}^T(\psi), \tag{9.4.9}$$

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

$$A = \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix}, \qquad a_{pp} \geq 0, \quad a_{qq} \geq 0.$$

The assumption of nonnegative diagonal elements is no restriction, since we can change the sign of these by premultiplication 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 Kogbetliantz's 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.

Review Questions

1. What is the asymptotic speed of convergence for the classical Jacobi method? Discuss the advantages and drawbacks of Jacobi methods compared to the QR algorithm.
2. There are two different Jacobi-type methods for computing the SVD were developed. What are they called? What 2×2 subproblems are they based on?

Problems

1. Compute with the Jacobi algorithm the eigenvalues of

$$A = \begin{pmatrix} -0.442 & -0.607 & -1.075 \\ -0.607 & 0.806 & 0.455 \\ -1.075 & 0.455 & -1.069 \end{pmatrix},$$

with an error less than 10^{-3}.

2. Suppose the matrix

$$\tilde{A} = \begin{pmatrix} 1 & 10^{-2} & 10^{-4} \\ 10^{-2} & 2 & 10^{-2} \\ 10^{-4} & 10^{-2} & 4 \end{pmatrix}.$$

has been obtained at a certain step of the Jacobi algorithm. Estimate the eigenvalues of $\tilde{A}$ as accurately as possible using the Gershgorin circles with a suitable diagonal transformation, see Problem 9.3.3.

3. Jacobi-type methods can also be constructed for Hermitian matrices using *elementary unitary rotations* of the form

$$U = \begin{pmatrix} \cos\theta & \alpha\sin\theta \\ -\bar{\alpha}\sin\theta & \cos\theta \end{pmatrix}, \qquad |\alpha| = 1.$$

Show that if we take $\alpha = a_{pq}/|a_{pq}|$ then equation (9.4.4) 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.)

4. 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. Consider the SVD of an upper triangular 2×2 matrix (9.4.8). where $\sigma_1 \geq \sigma_2$.
(a) Show that the singular values satisfy

$$\sigma_1\sigma_2 = |r_{11}r_{22}|, \quad \sigma_1^2 + \sigma_2^2 = r_{11}^2 + r_{22}^2 + r_{12}^2.$$

Deduce that

$$\sigma_{1,2} = \frac{1}{2}\left|\sqrt{(r_{11}+r_{22})^2 + r_{12}^2} \pm \sqrt{(r_{11}-r_{22})^2 + r_{12}^2}\right|, \tag{9.4.10}$$

of which the larger is σ_1 and the smaller $\sigma_2 = |r_{11}r_{22}|/\sigma_1$.
(b) Show that for the right singular vector $(s_v,\ c_v)$ is parallel to $(r_{11}^2 - \sigma_1^2, r_{11}r_{12})$. The left singular vectors then are obtained from

$$(c_u,\ s_u) = (r_{11}c_v - r_{12}s_v, r_{22}s_v)/\sigma_1.$$

6. Show that if Kogbetliantz's method is applied to a triangular matrix then after one sweep of the row cyclic algorithm (9.4.5) an upper (lower) triangular matrix becomes lower (upper) triangular.

9.5 The Power Method

9.5.1 The Simple Power Method

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.

Let $A \in \mathbf{R}^{n\times n}$ and $q_0 \neq 0$ be a given starting vector. In the power method the sequence of vectors $q_1, q_2, \ldots$ is formed, where

$$q_k = Aq_{k-1}, \qquad k = 1, 2, \ldots.$$

It follows that $q_k = A^k q_0$, which explains the name of the method. Note that in general it would be much more costly to form the matrix A^k, than to perform the above sequence of matrix vector multiplications.

We assume in the following that the eigenvalues are ordered so that

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

To simplify the analysis of the power method assume that the matrix A is diagonalizable. 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

$$\frac{1}{\lambda_1^k} q_k = \alpha_1 x_1 + O\left(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k \right), \tag{9.5.1}$$

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 Theorem 9.2.6 (9.2.1).

In practice the vectors q_k have to be normalized in order to avoid overflow or underflow. Hence we modify the initial recursion as follows. Assume that $\|q_0\|$=1, and compute

$$\hat{q}_k = Aq_{k-1}, \quad \mu_k = \|\hat{q}_k\|, \quad q_k = \hat{q}_k/\mu_k, \quad k = 1, 2, \ldots \tag{9.5.2}$$

Then we have

$$q_k = \frac{1}{\gamma_k} A^k q_0, \quad \gamma_k = \mu_1 \cdots \mu_k,$$

and under the assumptions above q_k converges to a normalized eigenvector x_1. From equations (9.5.1) and (9.5.2) it follows that

$$\hat{q}_k = \lambda_1 q_{k-1} + O(|\lambda_2/\lambda_1|^k), \qquad \lim_{k\to\infty} \mu_k = |\lambda_1|. \tag{9.5.3}$$

An approximation to λ_1 can also be obtained from the ratio of elements in the two vectors $\hat{q}_k$ and q_{k-1}. The convergence, which is slow when $|\lambda_2| \approx |\lambda_1|$, can be accelerated by Aitken extrapolation.

If the matrix A is real symmetric (or Hermitian) its eigenvalues are real and the eigenvectors can be chosen so that $X = (x_1, \ldots, x_n)$ is real and orthogonal. Using (9.5.1) one can show that the Rayleigh quotient converges twice as fast as μ_k,

$$\lambda_1 = \rho(q_{k-1}) + O\left(|\lambda_2/\lambda_1|^{2k}\right), \quad \rho(q_{k-1}) = q_{k-1}^T A q_{k-1} = q_{k-1}^T \hat{q}_k. \tag{9.5.4}$$

Example 9.5.1.

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 $e_k = \lambda_1 - \rho_k$ given in the table below:

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

The ratios of successive errors converge to $(\lambda_2/\lambda_1)^2 = 0.4019$.

The convergence of the power method depends on the assumption that $\alpha_1 \neq 0$, and hence we only can prove convergence for *almost all starting vectors*. Even when $\alpha_1 = 0$, rounding errors will tend to introduce a small component along x_1 in Aq_0, and therefore the method converges in practice 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

$$|\lambda_r| > |\lambda_{r+1}| \geq \cdots \geq |\lambda_n|.$$

However, an inherent weakness in this case is that the limit vector will depend on the expansion of q_0 along $x_1, \cdots, x_r$, and q_k will converge to *one particular vector* in the invariant subspace span$[x_1, \ldots, x_r]$. To determine the whole dominating invariant subspace we will have to perform the power method with $p \geq r$ linearly independent starting vectors, see Section 9.5.6.

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 Ay for any given vector y. If the matrix A is sparse the cost of one iteration step is proportional to the number of nonzero elements in A.

9.5.2 Deflation

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 Section 9.1, see (9.1.12). 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.

9.5.3 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 $p(x)$ and $q(x)$ be two polynomials such that $q(A)$ is nonsingular and define $r(A) = (q(A))^{-1} p(A)$. Then if A has an eigenvalue λ with corresponding eigenvector x it follows that $r(\lambda)$ is an eigenvalue of $r(A)$ with the same eigenvector x.

As a simple application of this assume that A is nonsingular and take $r(x) = 1/x$. Then the matrix $r(A) = A^{-1}$ has eigenvalues equal to $1/\lambda_i$. Hence from (9.5.3)

it follows 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}, then q_k will converge to the eigenvector x_n of A corresponding to λ_n. This is called **inverse iteration** , and was introduced by H. Wielandt in 1944.

Inverse iteration can also be applied to the matrix $A-\mu I$, where μ is a chosen **shift** of the spectrum. The eigenvalues of $(A-\mu I)^{-1}$ equal

$$\mu_j = (\lambda_j - \mu)^{-1}. \tag{9.5.5}$$

and the iteration can be written

$$(A-\mu I)\hat{q}_k = q_{k-1}, \qquad q_k = \hat{q}_k/\|\hat{q}_k\|_2, \qquad k = 1, 2, \ldots. \tag{9.5.6}$$

Note that there is no need to explicitly invert $A-\mu I$. Instead we compute a triangular factorization of $A-\mu I$, and in each step of (9.5.6) solve two triangular systems

$$L(U\hat{q}_k) = Pq_{k-1}, \qquad P(A-\mu I) = LU.$$

For a dense matrix A one step of the iteration (9.5.5) 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 $(A-\mu I)$. Hence in general partial pivoting must be employed.

If μ is chosen sufficiently close to an eigenvalue λ_i, so that $|\lambda_i - \mu| \ll |\lambda_j - \mu|$, $\lambda_i \neq \lambda_j$ then $(\lambda_i - \mu)^{-1}$ is a dominating eigenvalue of B,

$$|\lambda_i - \mu|^{-1} \gg |\lambda_j - \mu|^{-1}, \qquad \lambda_i \neq \lambda_j. \tag{9.5.7}$$

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

$$\frac{1}{\lambda_i - \mu} \approx q_{k-1}^T (A-\mu I)^{-1} q_{k-1} = q_{k-1}^T \hat{q}_k,$$

where $\hat{q}_k$ satisfies $(A-\mu I)\hat{q}_k = q_{k-1}$. Thus

$$\bar{\lambda}_i = \mu + 1/(q_{k-1}^T \hat{q}_k). \tag{9.5.8}$$

An a posteriori bound for the error in the approximate eigenvalue $\bar{\lambda}_i$ of A can be obtained from the residual corresponding to $(\bar{\lambda}_i, \hat{q}_k)$, which equals

$$r_k = A\hat{q}_k - \left(\mu + 1/(q_{k-1}^T \hat{q}_k)\right)\hat{q}_k = q_{k-1} - \hat{q}_k/(q_{k-1}^T \hat{q}_k).$$

Then, by Theorem 9.3.13, $(\bar{\lambda}_i, \hat{q}_k)$ is an exact eigenpair of a matrix $A+E$, where $\|E\|_2 \leq \|r_k\|_2/\|\hat{q}_k\|_2$. If A is real symmetric then the error in the approximative eigenvalue $\hat{\lambda}_i$ of A is bounded by $\|r_k\|_2/\|\hat{q}_k\|_2$.

9.5.4 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 (9.5.6) 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 machine unit 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 9.3.13 to inverse iteration. By (9.5.6) 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

$$\|q_{k-1}\|_2/\|\hat{q}_k\|_2 \leq u\|A\|_2. \tag{9.5.9}$$

Example 9.5.2.

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 $\|E\|_2 \leq 1/\|\hat{q}_1\|_2 = 0.73122 \cdot 10^{-4}$.

Inverse iteration is a useful algorithm for calculation of specified eigenvectors corresponding to well separated eigenvalues for dense matrices. 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 Section 9.6.

It is quite tricky to develop inverse iteration into a reliable algorithm in case 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 9.5.3.

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 (9.3.18) gives an expression for the backward error E of the computed eigenpair. An error bound can then be obtained by applying the perturbation analysis of Section 9.3. In the Hermitian case the eigenvalues are perfectly conditioned, and the error bound equals $\|E\|_2$. In general the sensitivity of an eigenvalue λ is determined by $1/s(\lambda) = 1/|y^H x|$, where x and y are right and left unit eigenvector corresponding to λ, see Section 9.3.2. 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 $s(\lambda)$.

9.5.5 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 $k = 1, 2, \ldots$ compute

$$\big(A - \rho(q_{k-1})I\big)\hat{q}_k = q_{k-1}, \qquad q_k = \hat{q}_k / \|\hat{q}_k\|_2, \tag{9.5.10}$$

where $\rho(q_{k-1}) = q_{k-1}^T A q_{k-1}$ 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 [32, 1973, Section 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 9.3.15. 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 [41, Section 4.8].

9.5.6 Subspace Iteration

A natural generalization of the power method is to iterate *simultaneously* with several vectors. Let $Z_0 = S = (s_1, \ldots, s_p) \in \mathbf{R}^{n\times p}$, be an initial matrix of rank $p > 1$. If we compute a sequence of matrices $\{Z_k\}$, from

$$Z_k = AZ_{k-1}, \qquad k = 1, 2, \ldots, \tag{9.5.11}$$

then it holds

$$Z_k = A^k S = (A^k s_1, \ldots, A^k s_p). \tag{9.5.12}$$

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 $\mathcal{S} = \text{span}\,(S)$ the iteration produces the subspaces $A^k\mathcal{S} = \text{span}\,(A^kS)$. Hence the problem is only that the basis $A^ks_1, \ldots, A^ks_p$ becomes more and more ill-conditioned. To avoid this, orthogonality between the columns can be maintained as follows. Starting with a matrix Q_0 with orthogonal columns we compute

$$Z_k = AQ_{k-1} = Q_kR_k, \qquad k = 1, 2, \ldots, \tag{9.5.13}$$

where Q_kR_k is the QR decomposition of Z_k. Here Q_k can be computed by Gram-Schmidt orthogonalization of Z_k. The iteration (9.5.13) is also called **orthogonal iteration**. Note that R_k plays the rule of a normalizing matrix. We have $Q_1 = Z_1R_1^{-1} = AQ_0R_1^{-1}$, and similarly

$$Q_k = A^kQ_0(R_k \cdots R_1)^{-1}. \tag{9.5.14}$$

It is important to note that if $Z_0 = Q_0$, then both iterations (9.5.11) and (9.5.13) will generate the same sequence of subspaces. $\mathcal{R}(A^kQ_0) = \mathcal{R}(Q_k)$. However, in orthogonal iteration an orthogonal bases for the subspace is calculated at each iteration. (Since the iteration (9.5.11) is less costly it is sometimes preferable to perform the orthogonalization in (9.5.13) 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

$$|\lambda_1| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n| \tag{9.5.15}$$

and let

$$U^HAU = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}, \qquad U = (U_1\ U_2) \tag{9.5.16}$$

be a Schur decomposition of A, partitioned correspondingly. 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 (9.5.13) converge to $\mathcal{U}_1$ when $k \to \infty$.

Theorem 9.5.1.

Let $\mathcal{U}_1$ be the dominant invariant subspace of A defined in (9.5.16). *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*

$$\theta_{\max}(A^k\mathcal{S}, \mathcal{U}_1) \leq C|\lambda_{p+1}/\lambda_p|^k.$$

where $\theta_{\max}(\mathcal{X}, \mathcal{Y})$ denotes the largest angle between the two subspaces (see Definition 9.3.6).

Proof. See Golub and Van Loan [16, pp. 333]. □

If we perform subspace iteration on p vectors, we are simultaneously performing subspace iteration on a nested sequence of subspaces

$$\text{span}\,(s_1), \quad \text{span}\,(s_1, s_2), \ldots, \quad \text{span}\,(s_1, s_2, \ldots, s_p).$$

This is also true for orthogonal iteration since this property is not changed by the orthogonalization procedure. Hence Theorem 9.5.1 shows that whenever $|\lambda_{q+1}/\lambda_q|$ is small for some $q \leq p$, the convergence to the corresponding dominant invariant subspace of dimension q will be fast.

We now show that there is a duality between direct and inverse subspace iteration.

Lemma 9.5.2. (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 $A^k x \perp (A^H)^{-k} y$. □

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 9.7.1 for the understanding of the QR algorithm.

Approximations to eigenvalues of A can be obtained from eigenvalues of the sequence of matrices

$$B_k = Q_k^T A Q_k = Q_k^T Z_{k+1} \in \mathbf{R}^{p\times p}. \tag{9.5.17}$$

Note that B_k is a generalized Rayleigh quotient, see Section 9.8.1– 9.8.2. Finally, both direct and inverse orthogonal iteration can be performed using a sequence of shifted matrices $A - \mu_k I$, $k = 0, 1, 2, \ldots$.

Review Questions

1. Describe the power method and its variants. Name at least one important application of the shifted inverse power method.
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. Describe how the power method can be generalized to simultaneously iterating with several starting vector.

Problems

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?

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. (a) Compute the residual vector corresponding to the last eigenpair obtained in Example 9.5.1, and give the corresponding backward error estimate.

(b) Perform Aitken extrapolation on the Rayleigh quotient approximations in Example 9.5.1 to compute an improved estimate of λ_1.

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 $A - 4I = LDL^T$.

5. For a symmetric matrix $A \in \mathbf{R}^{n\times n}$ it holds that $\sigma_i = |\lambda_i|$, $i = 1, \ldots, n$. 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.

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 $x = (0, 1)^T$.

9.6 Transformation to Condensed Form

9.6.1 Introduction

By Theorem 9.2.1 any matrix can be reduced to upper triangular form, the Schur canonical form, by a unitary similarity transformation. For a normal matrix this triangular form must necessarily be diagonal. In both cases we can read off the eigenvalues from the diagonal. The construction of the similarity transformation depended on the knowledge of successive eigenpairs, and this transformation can therefore in general not be realized by a finite process.

It is, however, possible to reduce a matrix to upper Hessenberg form, which is close to triangular, by a *finite* number of elementary similarity transformations. In

the symmetric case, a symmetric tridiagonal form is obtained. In several algorithms for finding the eigenvalues and eigenvectors of a matrix the work is greatly reduced if this transformation is first carried out.

9.6.2 Unitary Elementary Transformations

For transformation of complex matrices to condensed form we need to consider **unitary** Givens and Householder transformations. To generalize Givens rotations to the complex case, we consider matrices of the form

$$G = \begin{pmatrix} \bar{c} & \bar{s} \\ -s & c \end{pmatrix}, \qquad c = e^{i\gamma}\cos\theta, \qquad s = e^{i\delta}\sin\theta.$$

It is easily verified that the matrix $G^H = G$, i.e., G is unitary, and that $G^{-1} = G^H$ is itself a plane rotation Given a complex vector $(x_1 \ x_2)^T \in \mathbf{C}^2$ we now want to determine c and s so that

$$G\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix}, \qquad \sigma^2 = |x_1|^2 + |x_2|^2, \tag{9.6.1}$$

Further, (9.6.1) holds provided that

$$c = x_1/\sigma, \qquad s = x_2/\sigma.$$

The following algorithm generalizes Algorithm 7.4.2 to the complex case:

Algorithm 9.6.1

Given $x = (x_1, x_2)^T \neq 0$ construct c, s, and real σ in a complex Givens rotation such that $Gx = \sigma(1,0)^T$:

$[c, s, \sigma] = \text{givrot}(x_1, x_2)$
  if $|x_1| > |x_2|$
    $t = x_2/x_1; \quad u = \sqrt{1 + |t|^2};$
    $c = (x_1/|x_1|)/u; \quad s = tc; \quad \sigma = x_1/c;$
  else
    $t = x_1/x_2; \quad u = \sqrt{1 + |t|^2};$
    $s = (x_2/|x_2|)/u; \quad c = ts; \quad \sigma = x_2/s;$
  end

Householder transformations can also be generalized to the complex case. We consider **unitary** Householder transformations of the form

$$P = I - \frac{1}{\gamma}uu^H, \quad \gamma = \frac{1}{2}u^H u, \quad u \in \mathbf{C}^n. \tag{9.6.2}$$

It is easy to check that P is Hermitian, $P^H = P$, and unitary, $P^{-1} = P$. Given a vector $x \in \mathbf{C}^n$ we want to determine u such that $Px = ke_1$, $|k| = \sigma = \|x\|_2$. It is easily verified that if $x_1 = e^{i\alpha_1}|x_1|$ then u and γ are given by

$$u = x + ke_1, \quad k = \sigma e^{i\alpha_1}, \tag{9.6.3}$$

and

$$\gamma = \frac{1}{2}(\sigma^2 + 2|k||x_1| + |k|^2) = \sigma(\sigma + |x_1|). \tag{9.6.4}$$

Note that u differs from x only in its first component.

9.6.3 Reduction to Hessenberg Form

We now show how to reduce a matrix $A \in \mathbf{R}^{n\times n}$ to **Hessenberg** form by an orthogonal similarity,

$$Q^T A Q = H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,n-1} & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2,n-1} & h_{2n} \\ & h_{32} & \ddots & \vdots & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & h_{n,n-1} & h_{nn} \end{pmatrix}.$$

The orthogonal matrix Q will be constructed as a product of $n-2$ Householder transformations $Q = P_1 P_2 \cdots P_{n-2}$, where

$$P_k = I - \frac{1}{\gamma_k} u_k u_k^T, \qquad \gamma_k = \frac{1}{2}\|u_k\|_2^2 \tag{9.6.5}$$

(cf. the Householder QR decomposition in Section 8.4.3). Note that P_k is completely specified by u_k and γ_k, and that products of the form PA and AP, can each be computed in $2n^2$ flops by

$$PA = A - u_k(A^T u_k)^T/\gamma_k, \qquad AP = A - (Au_k)u_k^T/\gamma_k.$$

We compute $A = A^{(1)}, A^{(2)}, \ldots, A^{(n-1)} = H$, where $A^{(k+1)} = P_k A^{(k)} P_k$. In the first step, $k = 1$,

$$A^{(2)} = P_1 A P_1 = \begin{pmatrix} h_{11} & h_{12} & \tilde{a}_{13} & \ldots & \tilde{a}_{1n} \\ h_{21} & h_{22} & \tilde{a}_{23} & \ldots & \tilde{a}_{2n} \\ 0 & \tilde{a}_{32} & \tilde{a}_{33} & \ldots & \tilde{a}_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{n2} & \tilde{a}_{n3} & \ldots & \tilde{a}_{nn} \end{pmatrix},$$

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. After $(k-1)$ steps we have computed

$$A^{(k)} = \begin{pmatrix} H_{11} & h_{12} & \tilde{A}_{13} \\ 0 & a_{22} & \tilde{A}_{23} \end{pmatrix}, \tag{9.6.6}$$

where $(\,H_{11} \quad h_{12}\,) \in \mathbf{R}^{k\times k}$ is part of the final Hessenberg matrix. P_k is chosen to zero all elements but the first in a_{22}. After $n-2$ steps we have the required form

$$Q^T A Q = A^{(n-1)} = H, \qquad Q = P_1 P_2 \cdots P_{n-2}. \tag{9.6.7}$$

A simple operation count shows that this reduction requires $5n^3/3$ flops. Note that the transformation matrix Q is not explicitly computed, only the vectors defining the Householder transformations $P_1, P_2, \ldots, P_{n-2}$ are saved. These vectors can conveniently overwrite the corresponding elements in the matrix A using also two extra rows appended to A.

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

$$\|E\|_F \leq cn^2 u \|A\|_F, \tag{9.6.8}$$

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.

The reduction to Hessenberg form can also be achieved by using elementary elimination matrices as introduced in Section 7.3.5. These are lower triangular matrices of the form

$$L_j = I + m_j e_j^T, \qquad m_j = (0, \ldots, 0, m_{j+1,j}, \ldots, m_{n,j})^T.$$

Only the elements *below* the main diagonal in the jth column differ from the unit matrix. If a matrix A is premultiplied 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. We complete the similarity transformation $L_j A L_j^{-1} = \tilde{A} L_j^{-1}$ by postmultiplying

$$\tilde{A} L_j^{-1} = \tilde{A}(I - m_j e_j^T) = \tilde{A} - (\tilde{A} m_j) e_j^T.$$

In this operation a linear combination $\tilde{A} m_j$ of the last $n-j$ columns is *subtracted* from the jth column of $\tilde{A}$.

If the pivot element $a_{21} \neq 0$, then we can eliminate the last $n-2$ elements in the first column of A by the transformation $L_2 A$, where

$$m_2 = -(0, 0, a_{31}/a_{21}, \ldots, a_{n1}/a_{21})^T.$$

These zeros are not affected by the postmultiplication $(L_2 A)L_2^{-1}$, which only affects the elements in the last $n-1$ columns. Hence, if all pivot elements are nonzero we can complete the transformation to Hessenberg form. The vectors m_j, $j = 2, \ldots, n-1$ can overwrite the corresponding elements of A. The reduction may be unstable if some pivot elements are small. Therefore, in practice this algorithm has to be modified by the introduction of partial pivoting, in obvious analogy to Gaussian elimination. With this modification the stability of the reduction is usually as good as for the one using Householder reflections. The backward error bound will contain a growth ratio g_n, see Section 7.6.6, but a big growth rarely occurs in practice. The operation count for this reduction can be shown to be $n^3/3 + n^3/2 = 5n^3/6$ flops, or half that for the orthogonal reduction. Because of this reduction by elementary elimination matrices is often the preferred method.

By (9.6.8) 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. We say that a matrix $\tilde{A}$ is balanced for some norm l_p-norm if $\|\tilde{a}_i\|_p = \|\tilde{a}^i\|_p$, $i = 1, \ldots, n$ where $\tilde{a}_i$ and $\tilde{a}^i$ denote respectively the ith column and ith row of $\tilde{A}$. There are classes of matrices which do not need balancing; for example normal matrices are already balanced for $p = 2$.

An iterative algorithm has been given by Osborne that 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 [41].

9.6.4 Reduction to Symmetric Tridiagonal Form

If we carry out the orthogonal reduction to Hessenberg form 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

$$Q^T A Q = T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}. \tag{9.6.9}$$

If elementary elimination matrices are used for the reduction *symmetry is not preserved*. Hence in this case the orthogonal reduction is clearly superior. A similar remark applies to the case of the unitary reduction of a Hermitian matrix to Hermitian tridiagonal form.

In the kth step of the orthogonal reduction of a real symmetric matrix 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.

It is important to take advantage of symmetry to save storage and operations. Since the intermediate matrix $P_k A^{(k)}$ is not symmetric, this means that we must compute $P_k A^{(k)} P_k$ directly. Dropping the subscripts k we can write

$$PAP = \Big(I - \frac{1}{\gamma}uu^T\Big)A\Big(I - \frac{1}{\gamma}uu^T\Big) \tag{9.6.10}$$

$$= A - up^T - pu^T + u^T puu^T/\gamma \tag{9.6.11}$$

$$= A - uq^T - qu^T,$$

where

$$p = Au/\gamma, \quad q = p - \beta u, \quad \beta = u^T p/(2\gamma). \tag{9.6.12}$$

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 (9.6.8). 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 [3, 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.[3]

A similar algorithm can be used to transform a Hermitian matrix into a tridiagonal Hermitian matrix using the complex Householder transformation introduced in Section 9.6.2. With $U = P_1 P_2 \cdots P_{n-2}$ we obtain $T = U^H A U$, where T is Hermitian and therefore has positive real diagonal elements. By a diagonal similarity DTD^{-1}, $D = \text{diag}\,(e^{i\phi_1}, e^{i\phi_2}, \ldots, e^{i\phi_n})$ 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 [41, Section 10.5.1], see also Contribution II/8 in Wilkinson and Reinsch [41]. 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!

9.6.5 A Divide and Conquer Algorithm

The basic idea in the divide and conquer algorithm for the symmetric tridiagonal eigenproblem is to divide the tridiagonal matrix (9.7.30) into two smaller symmetric tridiagonal matrices T_1 and T_2 as follows.

$$T = \begin{pmatrix} T_1 & \beta_{k+1} e_k & 0 \\ \beta_{k+1} e_k^T & \alpha_{k+1} & \beta_{k+2} e_1^T \\ 0 & \beta_{k+2} e_1 & T_2 \end{pmatrix} = P \begin{pmatrix} \alpha_{k+1} & \beta_{k+1} e_k^T & \beta_{k+2} e_1^T \\ \beta_{k+1} e_k & T_1 & 0 \\ \beta_{k+2} e_1 & 0 & T_2 \end{pmatrix} P^T. \tag{9.6.13}$$

Here e_j is the jth unit vector of appropriate dimension and P is a permutation matrix permuting block rows and columns 1 and 2. T_1 and T_2 are $k \times k$ and $(n-k-1)\times(n-k-1)$ symmetric tridiagonal matrices and are principle submatrices of T.

Suppose now that the eigendecompositions of $T_i = Q_i D_i Q_i^T$, $i = 1, 2$ are known. Substituting into (9.6.13) we get

$$T = P \begin{pmatrix} \alpha_{k+1} & \beta_{k+1} e_k^T & \beta_{k+2} e_1^T \\ \beta_{k+1} e_k & Q_1 D_1 Q_1^T & 0 \\ \beta_{k+2} e_1 & 0 & Q_2 D_2 Q_2^T \end{pmatrix} P^T = Q H Q^T, \tag{9.6.14}$$

where

$$H = \begin{pmatrix} \alpha_{k+1} & \beta_{k+1} l_1^T & \beta_{k+2} f_2^T \\ \beta_{k+1} l_1 & D_1 & 0 \\ \beta_{k+2} f_2 & 0 & D_2 \end{pmatrix}, \qquad Q = P \begin{pmatrix} 1 & 0 & 0 \\ 0 & Q_1 & 0 \\ 0 & 0 & Q_2 \end{pmatrix},$$

and $l_1 = Q_1^T e_k$, $f_2 = Q_2^T e_1$. Hence the matrix T is reduced to H by an orthogonal

[3] Note that in the Householder tridiagonalization described in [41], Contribution II/2 the reduction is performed instead from the bottom up.

similarity transformation Q. The matrix H has the form

$$H = \begin{pmatrix} \alpha & z^T \\ z & D \end{pmatrix}, \quad D = \operatorname{diag}(d_2, \ldots, d_n).$$

where $z = (z_2, \ldots, z_n)^T$ is a vector. Such a matrix is called a **symmetric arrowhead matrix**. We assume that $d_2 \geq d_3 \geq \cdots \geq d_n$, which can be achieved by a symmetric permutation.

The eigenvalue problem for symmetric arrowhead matrices has been discussed in detail in Wilkinson [40, pp. 95–96]. In particular, if we assume that the elements d_i are distinct, $d_2 > d_3 > \cdots > d_n$, and that $z_i > 0$, $i = 2, \ldots, n$, then the eigenvalues and eigenvectors of H are characterized by the following lemma (cf. Problem 9.3.8).

Lemma 9.6.1.

The eigenvalues $\{\lambda_i\}_{i=1}^n$ of H satisfy the **secular equation**

$$f(\lambda) = \lambda - \alpha + \sum_{j=2}^{n} \frac{z_j^2}{d_j - \lambda} = 0. \tag{9.6.15}$$

and the interlacing property $\lambda_1 > d_2 > \lambda_2 > \cdots > d_n > \lambda_n$. For each eigenvalue λ_i of H, a corresponding (unnormalized) eigenvector is given by

$$u_i = \left(-1, \frac{z_2}{d_2 - \lambda_i}, \ldots, \frac{z_n}{d_n - \lambda_i}\right)^T. \tag{9.6.16}$$

Hence simple roots of the secular equation are isolated in an interval (d_i, d_{i+1}) where $f(\lambda)$ is monotonic and smooth. A zerofinder based on rational interpolation can be constructed which gets guaranteed quadratic convergence.

We make the following observations:

- If $d_i = d_{i+1}$ for some i, $2 \leq i \leq n-1$, then it can be shown that one eigenvalue of H equals d_i, and the degree of the secular equation may be reduced by one.
- If $z_i = 0$, then one eigenvalue equals d_i, and again the degree of the secular equation is decreased by one.

The splitting in (9.6.13) can be applied recursively to T_1 and T_2, i.e., we can repeat the splitting on each T_1 and T_2, etc., until the original tridiagonal matrix T has been reduced to a desired number of small subproblems. Then the relations in Lemma 9.6.1 may be applied from the bottom up to glue the eigensystems together.

In practice the formula for the eigenvectors in Lemma 9.6.1 cannot be used directly. The reason for this is that we can only compute an approximation $\hat{\lambda}_i$ to λ_i. Even if $\hat{\lambda}_i$ is very close to λ_i, the approximate ratio $z_j/(d_j - \hat{\lambda}_i)$ can be very different from the corresponding exact ratio. These errors may lead to computed eigenvectors of T which are numerically not orthogonal. Fortunately 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 [17, 1975]. The resulting algorithm seems to outperform the QR algorithm even on single processor computers.

9.6.6 Spectrum Slicing

Sylvester's law of inertia (see Theorem 9.3.10) 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 9.6.2.

Assume that symmetric Gaussian elimination can be carried through for $A-\tau I$ yielding the factorization (cf. (6.4.5))

$$A - \tau I = LDL^T, \quad D = \operatorname{diag}(d_1, \ldots, d_n), \tag{9.6.17}$$

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 $d_1, \ldots, d_n$.

Example 9.6.1.

The LDL^T factorization

$$A - 1 \cdot I = \begin{pmatrix} 1 & 2 & \\ 2 & 2 & -4 \\ & -4 & -6 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -2 & \\ & & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 & \\ & 1 & 2 \\ & & 1 \end{pmatrix}.$$

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 9.6.1. Then $a_{11} - \tau = 0$, and the first step in the factorization cannot be carried out. A closer analysis shows that the factorization will fail if, and only if, τ equals an eigenvalue to one or more of the $n-1$ leading principal submatrices of A. If τ is chosen in a small interval around each of these values, big growth of elements occurs and the factorization may give the wrong count. In such cases one should perturb τ by a small amount 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 $D = \operatorname{diag}(d_1, \ldots, d_n)$. 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 9.6.2

Tridiagonal Spectrum Slicing Let T be the tridiagonal matrix (9.6.9). 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 9.6.6 is backward stable. A round-off error analysis shows that the computed values $\bar{d}_k$ satisfy exactly (let $\beta_1 = 0$)

$$\begin{aligned}\bar{d}_k &= fl\big((\alpha_k - \beta_k(\beta_k/\bar{d}_{k-1})) - \tau\big)\\ &= \Big(\Big(\alpha_k - \frac{\beta_k^2}{\bar{d}_{k-1}}(1+\epsilon_{1k})(1+\epsilon_{2k})\Big)(1+\epsilon_{3k}) - \tau\Big)(1+\epsilon_{4k}) \qquad (9.6.18)\\ &\equiv \alpha_k' - \tau - (\beta_k')^2/\bar{d}_{k-1}, \qquad k = 1, \ldots, n,\end{aligned}$$

where $|\epsilon_{ik}| \le u$. Hence, the computed number $\bar{\pi}$ is the exact number of eigenvalues greater than τ of a matrix A', where A' has elements satisfying

$$|\alpha_k' - \alpha_k| \le u(2|\alpha_k| + |\tau|), \qquad |\beta_k' - \beta_k| \le 2u|\beta_k|. \qquad (9.6.19)$$

This is a very satisfactory backward error bound. It has been improved even further by Kahan [19, 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 Section 6.1.1) locate λ_k in an interval of length $(\tau_u - \tau_l)/2^p$. From Gershgorin's theorem it follows that all the eigenvalues of a tridiagonal matrix are contained in the union of the intervals $\alpha_i \pm (|\beta_i| + |\beta_{i+1}|)$, $i = 1, \ldots, n$ ($\beta_1 = \beta_{n+1} = 0$).

Using the bound (9.3.19) it follows that the bisection error in each computed eigenvalue is bounded by $|\bar{\lambda}_j - \lambda_j| \le \|A' - A\|_2$, where from (9.5.11), using the

improved bound by Kahan, and the inequalities $|\tau| \leq \|A\|_2$, $|\alpha_k| \leq \|A\|_2$ it follows that

$$|\bar{\lambda}_j - \lambda_j| \leq 5u\|A\|_2. \tag{9.6.20}$$

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 Section 9.6.4) 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 Section 9.9.

Review Questions

1. Describe how an arbitrary square matrix can be reduced to Hessenberg form by a sequence of orthogonal similarity transformations. If this reduction is applied to a real symmetric matrix what condensed form is obtained?
2. Describe the method of spectrum slicing for determining selected eigenvalues of a real symmetric matrix A.

Problems

1. Reduce to tridiagonal form, using an exact orthogonal similarity, the real symmetric matrix

$$A = \begin{pmatrix} 1 & \sqrt{2} & \sqrt{2} & \sqrt{2} \\ \sqrt{2} & -\sqrt{2} & -1 & \sqrt{2} \\ \sqrt{2} & -1 & \sqrt{2} & \sqrt{2} \\ 2 & \sqrt{2} & \sqrt{2} & -3 \end{pmatrix}$$

2. Show that if a real skew symmetric matrix A, $A^T = -A$, is reduced to Hessenberg form H by an orthogonal similarity, then H is a real skew symmetric tridiagonal matrix. Perform the reduction of the circulant matrix A (see Problem 9.1.9) with first row equal to

$$(0, 1, 1, 0, -1, -1).$$

3. To compute the eigenvalues of the following pentadiagonal matrix

$$A = \begin{pmatrix} 4 & 2 & 1 & 0 & 0 & 0 \\ 2 & 4 & 2 & 1 & 0 & 0 \\ 1 & 2 & 4 & 2 & 1 & 0 \\ 0 & 1 & 2 & 4 & 2 & 1 \\ 0 & 0 & 1 & 2 & 4 & 2 \\ 0 & 0 & 0 & 1 & 2 & 4 \end{pmatrix},$$

we first reduce A to tridiagonal form.

(a) Determine a Givens rotation G_{23} which zeros the element in position $(3,1)$ in $G_{23}A$. Compute the the transformed matrix $A^{(1)} = G_{23}AG_{23}^T$.

(b) In the matrix $A^{(1)}$ a new nonzero element has been introduced. Show how this can be zeroed by a new rotation without introducing any new nonzero elements.

(c) Device a "zero chasing" algorithm to reduce a general real symmetric pentadiagonal matrix $A \in \mathbf{R}^{n\times n}$ to symmetric tridiagonal form. How many rotations are needed? How many flops?

4. (a) Use one Givens rotation to transform to tridiagonal form the matrix

$$A = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix}.$$

(b) Compute the largest eigenvalue of A, using spectrum slicing on the tridiagonal form derived in (a). Then compute the corresponding eigenvector.

5. Show that (9.6.17) can be written

$$\hat{d}_k = \alpha_k - \frac{\beta_k^2}{\hat{d}_{k-1}} \frac{(1+\epsilon_{1k})(1+\epsilon_{2k})}{(1+\epsilon_{3,k-1})(1+\epsilon_{4,k-1})} - \frac{\tau}{(1+\epsilon_{3k})}, \quad k = 1, \ldots, n,$$

where we have put $\bar{d}_k = \hat{d}_k(1+\epsilon_{3k})(1+\epsilon_{4k})$, and $|\epsilon_{ik}| \leq u$. Conclude that since $\text{sign}(\hat{d}_k) = \text{sign}(\bar{d}_k)$ the computed number $\bar{\pi}$ is the exact number of eigenvalues a tridiagonal matrix A' whose elements satisfy

$$|\alpha_k' - \alpha_k| \leq u|\tau|, \quad |\beta_k' - \beta_k| \leq 2u|\beta_k|.$$

9.7 The LR and QR Algorithms

9.7.1 The Basic LR and QR Algorithms

The LR algorithm, developed by Rutishauser in [28, 1958], is an iterative method of reducing a matrix to triangular form by a sequence of similarity transformations. Rutishauser observed that if $A = LR$ then a similarity transformation of A is

$$L^{-1}AL = L^{-1}(LR)L = RL.$$

Hence the matrix obtained by multiplying the factors in reverse order gives a matrix similar to A. The LR algorithm is obtained by repeating this process.

Setting $A_1 = A$ we compute $A_{k+1} = L_k^{-1} A_k L_k$ from

$$A_k = L_k R_k, \qquad A_{k+1} = R_k L_k, \quad k = 1, 2, \ldots \tag{9.7.1}$$

Repeated application of (9.7.1) gives

$$A_k = L_{k-1}^{-1} \cdots L_2^{-1} L_1^{-1} A_1 L_1 L_2 \cdots L_{k-1}. \tag{9.7.2}$$

or

$$L_1 L_2 \cdots L_{k-1} A_k = A_1 L_1 L_2 \cdots L_{k-1}. \tag{9.7.3}$$

The two matrices defined by

$$T_k = L_1 \cdots L_{k-1} L_k, \qquad U_k = R_k R_{k-1} \cdots R_1, \tag{9.7.4}$$

are lower and upper triangular respectively. Forming the product $T_k U_k$ and using (9.7.3) we have

$$\begin{aligned} T_k U_k &= L_1 \cdots L_{k-1} (L_k R_k) R_{k-1} \cdots R_1 \\ &= L_1 \cdots L_{k-1} A_k R_{k-1} \cdots R_1 \\ &= A_1 L_1 \cdots L_{k-1} R_{k-1} \cdots R_1. \end{aligned}$$

Repeating this we obtain the basic relation

$$T_k U_k = A_1^k. \tag{9.7.5}$$

This shows that the close relation between the LR algorithm and the power method.

It is possible to show that under certain restrictions the matrix A_k converges to an upper triangular matrix R_∞. The eigenvalues are then equal to the diagonal elements of R_∞. In establishing the convergence result several assumptions need to be made. for example, that the LR factorization exists at every stage. This is not be true for the simple matrix

$$A = \begin{pmatrix} 0 & 1 \\ -3 & 4 \end{pmatrix},$$

with eigenvalues 1 and 3. Although we could equally well work with the shifted matrix $A + I$, which has a triangular factorization, there are other problems with the LR algorithm, which makes a robust implementation difficult.

In order 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, developed independently by Francis [10, 1961] and Kublanovskaya [20, 1961].[4] It then represented a significant and genuinely new contribution to eigensystems computation.

In the QR algorithm applied to $A_1 = A$ the matrix $A_{k+1} = Q_k^T A_k Q_k$, is computed from

$$A_k = Q_k R_k, \qquad A_{k+1} = R_k Q_k, \quad k = 1, 2, \ldots, \tag{9.7.6}$$

[4] 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.

where Q_k is orthogonal and R_k is upper triangular, i.e., in the kth step we first compute the QR decomposition of the matrix A_k and then multiply the factors in reverse order to get A_{k+1}.

The successive iterates of the QR algorithm satisfy relations similar to those derived for the LR algorithm. 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 (9.7.6) it follows that

$$A_{k+1} = P_k^T A P_k. \tag{9.7.7}$$

Further we have

$$\begin{aligned} P_k U_k &= Q_1 \cdots Q_{k-1}(Q_k R_k) R_{k-1} \cdots R_1 && (9.7.8)\\ &= Q_1 \cdots Q_{k-1} A_k R_{k-1} \cdots R_1 && (9.7.9)\\ &= A_1 Q_1 \cdots Q_{k-1} R_{k-1} \cdots R_1. && (9.7.10) \end{aligned}$$

Repeating this gives

$$P_k U_k = A_1^k. \tag{9.7.11}$$

When A is real symmetric and positive definite we can modify the LR algorithm and use the Cholesky factorization $A = LL^T$ instead. The algorithm then takes the form

$$A_k = L_k L_k^T, \qquad A_{k+1} = L_k^T L_k, \quad k = 1, 2, \ldots. \tag{9.7.12}$$

and we have

$$A_{k+1} = L_k^{-1} A_k L_k = L_k^T A_k L_k^{-T}. \tag{9.7.13}$$

Clearly all matrices A_k are symmetric and positive definite and the algorithm is well defined. Repeated application of (9.7.13) gives

$$A_k = T_{k-1}^{-1} A_1 T_{k-1} = T_{k-1}^T A_1 (T_{k-1}^{-1})^T, \tag{9.7.14}$$

where $T_k = L_1 L_2 \cdots L_k$. Further we have

$$A_1^k = (L_1 L_2 \cdots L_k)(L_k^T \cdots L_2^T L_1^T) = T_k T_k^T. \tag{9.7.15}$$

When A is real symmetric and positive definite there is a close relationship between the LR and QR algorithms. For the QR algorithm we have $A_k^T = A_k = R_k^T Q_k^T$ and hence

$$A_k^T A_k = A_k^2 = R_k^T Q_k^T Q_k R_k = R_k^T R_k, \tag{9.7.16}$$

which shows that R_k^T is the lower triangular Cholesky factor of A_k^2.

For the Cholesky LR algorithm we have from (9.7.4) and (9.7.5)

$$A_k^2 = L_k L_{k+1} (L_k L_{k+1})^T. \tag{9.7.17}$$

These two Cholesky factorizations (9.7.16) and (9.7.16) 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 (9.7.14) 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.

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

$$Z_k = A\tilde{Q}_k, \qquad Z_k = \tilde{Q}_{k+1} R_k. \qquad k = 0, 1, \ldots \tag{9.7.18}$$

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 (9.7.18) 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 (9.7.6). From the definition of B_k and (9.7.6) we have $\tilde{Q}_k = P_{k-1}$, and (compare (9.5.4))

$$A^k = \tilde{Q}_k \tilde{R}_k, \quad \tilde{R}_k = R_k \cdots R_2 R_1. \tag{9.7.19}$$

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 $(e_1, \ldots, e_p)$, $p = 1, \ldots, n$. It is important for the understanding of the QR algorithm to recall that therefore, according to Theorem 9.5.1, also inverse iteration by $(A^H)^{-1}$ takes place on the orthogonal complements, i.e., the subspaces spanned by $(e_{p+1}, \ldots, e_n)$, $p = 0, \ldots, n-1$. 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.)

9.7.2 Convergence of the Basic QR Algorithm

Assume that the eigenvalues of A satisfy $|\lambda_p| > |\lambda_{p+1}|$, and let (9.5.16) be a corresponding Schur decomposition. Let $P_k = (P_{k1},\ P_{k2})$, $P_{k1} \in \mathbf{R}^{n\times p}$, be defined by (9.7.6). Then by Theorem 9.5.1 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

$$A_k = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} + O\left(\left(|\lambda_{p+1}/\lambda_p| \right)^k \right).$$

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 9.7.1.

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. See Watkins [38]. □

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.

To speed up convergence the QR algorithm can be applied to the matrix $\tilde{A} = A - \tau I$, where τ is a **shift**. If τ approximates a simple eigenvalue λ_j of A, then in general $|\lambda_i - \tau| \gg |\lambda_j - \tau|$ 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. Usually a different shift in used in each step. If further the shift is restored at the end of the step the QR iteration can be written

$$A_k - \tau_k I = Q_k R_k, \qquad R_k Q_k + \tau_k I = A_{k+1}, \qquad k = 0, 1, 2, \ldots, \tag{9.7.20}$$

It is easily verified that with this shifted QR iteration we have $A_{k+1} = Q_k^T A_k Q_k$, and the relation to the power method is now expressed by the following result.

Theorem 9.7.2.

Let Q_k and R_k be computed by the QR algorithm (9.7.20). Then

$$\begin{aligned} &(A - \tau_k I) \cdots (A - \tau_1 I)(A - \tau_0 I) = P_k U_k, \\ &P_k = Q_0 Q_1 \cdots Q_k, \qquad U_k = R_k R_{k-1} \cdots R_0. \end{aligned} \tag{9.7.21}$$

Proof. For $k = 0$ the relation (9.7.21) is just the defining equation 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

$$A_{k+1} - \tau_k I = P_k^T (A - \tau_k I) P_k. \tag{9.7.22}$$

Hence, $R_k = (A_{k+1} - \tau_k I) Q_k^T = P_k^T (A - \tau_k I) P_k Q_k^T = P_k^T (A - \tau_k I) P_{k-1}$. Postmultiplying 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

$$A_k = Q_k L_k, \qquad L_k Q_k = A_{k+1}, \qquad k = 0, 1, 2, \ldots, \tag{9.7.23}$$

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 Section 9.6.4. 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 Section 9.6.4.) Of course, the same effect can be achieved by explicitly reversing the ordering of the rows and columns.

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. However, if the matrix A is initially reduced, as described in Section 9.6, to upper Hessenberg form, or in the real symmetric case to tridiagonal form, this form is preserved by the QR iteration. The cost is then reduced to only $4n^2$ flops per iteration, or about $12n$ flops per iteration in the real symmetric case. The QR algorithm in practice also depends on several other factors to achieve full accuracy and efficiency. Some of these will be discussed in the following sections.

9.7.3 QR Algorithm for Hessenberg Matrices

We first show that Hessenberg form is preserved by the QR iteration. Let H_k be upper Hessenberg and for $k = 0, 1, 2, \ldots$

$$H_k - \tau_k I = Q_k R_k, \qquad R_k Q_k + \tau_k I = H_{k+1}. \tag{9.7.24}$$

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 6.2.5). Hence $R_k Q_k$ and H_{k+1} are again of upper Hessenberg form.

In the **explicit-shift** QR algorithm we first form the matrix $H_k - \tau_k I$, and then apply a sequence of Givens rotations, $G_{j,j+1}$, $j = 1, \ldots, n-1$ (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 ($n = 5$, $j = 3$) the partially reduced matrix has the form

$$\begin{pmatrix} \rho_{11} & \times & \times & \times & \times \\ & \rho_{22} & \times & \times & \times \\ & & \nu_{33} & \times & \times \\ & & h_{43} & \times & \times \\ & & & \times & \times \end{pmatrix}.$$

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 obviously a Hessenberg matrix.

If an upper Hessenberg matrix H has a zero subdiagonal entry, then it decomposes into the form

$$H = \begin{pmatrix} H_1 & B \\ 0 & H_2 \end{pmatrix}.$$

The eigenvalues of H are then the sum of the eigenvalues of the two Hessenberg matrices H_1 and H_2, and the eigenvalue problem splits into two problems of smaller dimensions. A Hessenberg matrix H with no zero subdiagonal entries $h_{i,i-1} \neq 0$, $i = 2, \ldots, n$, is called **unreduced**.

We now discuss the choice of the shift τ. If τ is an exact eigenvalue of H (and therefore of A) 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 *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 $h'_{nn} = \tau$, Hence the QR algorithm converges in one step to the eigenvalue τ. This indicates that τ should be chosen to approximate an eigenvalue λ of A. Then $h_{n,n-1}$ will converge to zero 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

$$C = \begin{pmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{n,n} \end{pmatrix}, \tag{9.7.25}$$

closest to $h_{n,n}$, although this has the disadvantage of introducing complex arithmetic even when A is real. A way to avoid this is described below.

A important question is when to stop the iterations and accept an eigenvalue approximation. When

$$|h_{n,n-1}| \leq \epsilon(|h_{n-1,n-1}| + |h_{n,n}|),$$

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 all subdiagonal elements tend to zero, although $h_{n,n-1}$ most quickly. Whenever $|h_{p,p-1}| = 0$ for some $p < n$, we can deflate 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. On the average, 1-2 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 develop an **implicit-shift** QR-algorithm, which avoids this type of error. It is based on the observation that the matrix H_{k+1} in a QR iteration (9.7.24) *is essentially uniquely defined by the first column in* Q_k, *provided it is unreduced.* This property is stated in the following theorem.

When combined with a preliminary reduction to Hessenberg or symmetric tridiagonal form (see Section 9.6) the QR algorithm yields a very efficient method for finding all eigenvalues and eigenvectors of small to medium size matrices. Then the necessary modifications to make it into a practical method are described. The general nonsymmetric case is treated in Section 9.7.3 and the real symmetric case in Section 9.7.4.

Theorem 9.7.3. *Implicit Q Theorem.*

Given $A, H, Q \in \mathbf{R}^{n\times n}$, where $Q = (q_1, \ldots, q_n)$ 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 $q_1, \ldots, q_k$ 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 $(q_1, q_2, \ldots, q_n)H = A(q_1, q_2, \ldots, q_n)$ we obtain

$$h_{1,k}q_1 + \cdots + h_{k,k}q_k + h_{k+1,k}q_{k+1} = Aq_k.$$

Multiplying this by q_i^T and using the orthogonality of Q, we obtain

$$h_{ik} = q_i^T Aq_k, \qquad i = 1, \ldots, k.$$

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 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 $\|q_{k+1}\|_2 = 1$. ∎

For simplicity we drop the iteration index and write (9.7.24) as

$$H - \tau I = QR, \qquad H' = RQ + \tau I. \tag{9.7.26}$$

To apply Theorem 9.7.3 to the QR algorithm we must find the first column q_1 in Q. From $H - \tau I = QR$ with R upper triangular it follows that $r_{11}q_1$ equals the first column in $H - \tau I$, which is

$$h_1 = (h_{11} - \tau, h_{21}, 0, \cdots, 0)^T.$$

If we choose a Givens rotation G_{12} so that $G_{12}^T h_1 = \pm\|h_1\|_2 e_1$, then $G_{12}e_1$ is proportional to h_1, and (take $n = 6$)

$$G_{12}^T H = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times, \end{pmatrix} \quad G_{12}^T H G_{12} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

To preserve the Hessenberg form a rotation G_{23} is chosen to zero the element $+$,

$$G_{23}^T G_{12}^T H G_{12} G_{23} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & + & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

We continue to chase the element $+$ down the diagonal, with rotations $G_{34}, \ldots, G_{n-1,n}$ until it disappears. We have then obtained a Hessenberg matrix $Q^T H Q$, where the first column in Q is $G_{12}G_{23}\cdots G_{n-1,n}e_1 = G_{12}e_1$. From Theorem 9.7.3 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.

To avoid complex arithmetic when H is real one can *adopt the implicit-shift QR algorithm to compute the real Schur form* in Theorem 9.2.2, 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 (9.7.25), and consider two QR iterations with these shifts,

$$\begin{aligned} H - \tau_1 I &= Q_1 R_1, \quad & H' &= R_1 Q_1 + \tau_1 I, \\ H' - \tau_2 I &= Q_2 R_2, \quad & H'' &= R_2 Q_2 + \tau_2 I. \end{aligned}$$

We now show how to compute H'' directly from H using real arithmetic. We have $H'' = (Q_1 Q_2)^T H Q_1 Q_2$ and from Theorem 9.7.2

$$\begin{aligned} (Q_1 Q_2)(R_2 R_1) &= (H - \tau_1 I)(H - \tau_2 I) \\ &= H^2 - (\tau_1 + \tau_2)H + \tau_1 \tau_2 I \equiv G, \end{aligned}$$

where $(\tau_1 + \tau_2)$ and $\tau_1\tau_2$ are real. By the uniqueness theorem (Q_1Q_2) is determined from its first column, which is proportional to the first column $g_1 = Ge_1 = (u, v, w, 0, \ldots, 0)^T$ of G. Taking out a factor $h_{21} \neq 0$ this can be written $g_1 = h_{21}(p, q, r, 0, \ldots, 0)^T$, where

$$p = (h_{11}^2 - (\tau_1 + \tau_2)h_{11} + \tau_1\tau_2)/h_{21} + h_{12}, \qquad (9.7.27)$$
$$q = h_{11} + h_{22} - (\tau_1 + \tau_2), \qquad r = h_{32}.$$

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 (9.7.27), and grouping terms to reduce roundoff errors, we get

$$p = [(h_{nn} - h_{11})(h_{n-1,n-1} - h_{11}) - h_{n,n-1}h_{n-1,n}]/h_{21} + h_{12}$$
$$q = (h_{22} - h_{11}) - (h_{nn} - h_{11}) - (h_{n-1,n-1} - h_{11}), \qquad r = h_{32}.$$

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.

Suppose the QR algorithm has converged to the final upper triangular matrix T. Then we have

$$P^T H P = T, \qquad P = Q_0 Q_1 Q_2 \cdots,$$

where Q_k is a product of Givens rotations, and P is the product of all the transformations used. The eigenvectors z_i, $i = 1, 2, \ldots, n$ of T satisfy $Tz_i = \lambda_i z_i$, $z_1 = e_1$, and z_i is a linear combination of $e_1, \ldots, e_i$. The nonzero components of z_i can then be computed by back-substitution

$$z_{ii} = 1, \quad z_{ji} = -\Big(\sum_{k=j+1}^{i} t_{jk} z_{ki}\Big) \Big/ (\lambda_j - \lambda_i), \quad j = i-1, \ldots, 1. \qquad (9.7.28)$$

The eigenvectors of H are then given by Pz_i, $i = 1, 2, \ldots, n$. Finally if H has been obtained by reducing a matrix A to Hessenberg form as described in Section 9.6.3, then the eigenvectors of A can be computed from

$$x_i = UPz_i, \quad i = 1, 2, \ldots, n, \qquad U^H AU = H. \tag{9.7.29}$$

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.

If we do not want the eigenvectors, then it is not necessary to save the sequence of orthogonal transformations. It is even possible to avoid storing the rotations by performing the postmultiplications simultaneously with the premultiplications. 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 postmultiplication with G_{12}^T. Hence we can alternately pre- and postmultiply; in the next step we compute $(G_{34}((G_{23}G_{12}H_k)G_{12}^T))G_{23}^T$, and so on.

9.7.4 QR Algorithm for Symmetric Tridiagonal Matrices

By the methods described in Section 9.6 any Hermitian (real symmetric) matrix can by a unitary (orthogonal) similarity transformation be reduced into real, symmetric tridiagonal form

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}. \tag{9.7.30}$$

A tridiagonal matrix T is called **unreduced** if all off-diagonal elements are nonzero, $\beta_i \neq 0$, $i = 2, \ldots, n$. Let T be unreduced and λ an eigenvalue of T. Then $\operatorname{rank}(T - \lambda I) = n - 1$ (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. *Thus all eigenvalues of an unreduced symmetric tridiagonal matrix are distinct.* In the following we can assume that T is unreduced, since otherwise it can be split up in smaller unreduced tridiagonal matrices.

The QR algorithm also preserves symmetry. Hence it follows that if T is symmetric tridiagonal, and

$$T - \tau I = QR, \qquad T' = RQ + \tau I, \tag{9.7.31}$$

then also $T' = Q^T T Q$ is symmetric tridiagonal.

From the Implicit Q Theorem (Theorem 9.7.3) we have the following result, which can be used to develop an implicit QR algorithm.

Theorem 9.7.4.

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 (9.7.31) such that $Q^T A Q$ is an unreduced tridiagonal matrix. Then by Theorem 9.7.4 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 $n = 5$:

$$G_{12}^T T = \begin{pmatrix} \times & \times & + & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{pmatrix}, \quad G_{12}^T T G_{12} = \begin{pmatrix} \times & \times & + & & \\ \times & \times & \times & & \\ + & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{pmatrix}.$$

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 9.7.3 it follows that the result must be the matrix T' in (9.7.31).

There are several possible ways to choose the shift. The Rayleigh quotient shift $\tau = \alpha_n$, gives the same result as Rayleigh Quotient Iteration starting with e_n. This leads to generic cubic convergence, but not guaranteed. The **Wilkinson shift** is usually preferred. This shift τ equals that eigenvalue of the submatrix

$$\begin{pmatrix} \alpha_{n-1} & \beta_n \\ \beta_n & \alpha_n \end{pmatrix},$$

which is closest to α_n. A suitable formula for computing this shift is

$$\tau = \alpha_n - \operatorname{sign}(\beta) \beta_n^2 \Big/ \left(|\beta| + \sqrt{\beta^2 + \beta_n^2} \right), \quad \beta = (\alpha_{n-1} - \alpha_n)/2 \tag{9.7.32}$$

(cf. Algorithm (9.4.1)). A great advantage of the Wilkinson shift is that it gives guaranteed *global* convergence.[5] It can also be shown to give almost always *local cubic convergence*, although quadratic convergence might be possible.

The simplest way to check convergence is to use the criterion

$$|\beta_i| \leq \epsilon \, \|T\| \approx \epsilon \max_{1 \leq j \leq n} (|\beta_j|, \ |\alpha_j|).$$

This is sufficient for backward stability in the norm sense. However, it may not be accurate enough for graded matrices. For the QR algorithm one can instead use

$$|\beta_i| \leq \epsilon \, \max(|\beta_n|, |\alpha_n|).$$

We will not give more details of the algorithm here. If full account of symmetry is taken then one QR iteration can be implemented in only $9n$ multiplications, $2n$ divisions, $n-1$ square roots and $6n$ additions. By reorganizing the inner loop of the QR algorithm, it is possible to eliminate square roots and lower the operation count to about 4n multiplications, 3n divisions and 5n additions. This **rational QR algorithm** is the fastest way to get the eigenvalues alone, but does not directly yield the eigenvectors.

The Wilkinson shift may not give the eigenvalues in monotonic order. If some of the smallest or largest eigenvalues are wanted, then it is usually recommended to use Wilkinson shifts anyway and risk finding a few extra eigenvalues. To check if all wanted eigenvalues have been found one can use spectrum slicing, see Section 9.6.5. For a detailed discussion of variants of the symmetric tridiagonal QR algorithm, see Parlett [41].

If T has been obtained by reducing a Hermitian matrix to real symmetric tridiagonal form, $U^H A U = T$, then the eigenvectors are given by

$$x_i = UPe_i, \quad i = 1, 2, \ldots, n, \tag{9.7.33}$$

where $P = Q_0 Q_1 Q_2 \cdots$ 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 Section 9.7.1 that the QL algorithm is just the QR algorithm on JAJ. Note that if A is tridiagonal then JAJ is tridiagonal with

[5] A proof is given in Parlett [41, Chapter 8].

diagonals reversed. In the implicit QR algorithm one chooses the shift from the bottom of A and chases bulge from top to bottom. 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.

9.7.5 The Basic QR-SVD algorithm

For the computation of the SVD of a matrix $A \in \mathbf{R}^{m\times n}$ it is usually advisable to first perform a QR decomposition with column pivoting of A

$$A\Pi = Q \begin{pmatrix} R \\ 0 \end{pmatrix}. \tag{9.7.34}$$

(We assume in the following that $m \geq n$. This is no restriction since otherwise we can consider A^T.) Let let $R = U_R \Sigma V^T$ be the SVD of R. Then it follows that

$$A = U\Sigma V^T, \quad U = Q \begin{pmatrix} U_R \\ 0 \end{pmatrix}. \tag{9.7.35}$$

Clearly 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 form those of R.

Starting with $R_1 = R$, let us compute a sequence of upper triangular matrices by the iteration

$$R_k^T = Q_{k+1} R_{k+1}, \quad k = 0, 1, 2, \ldots. \tag{9.7.36}$$

Note that in each step the QR factorization of a *lower* triangular matrix is computed. We will show that this iteration is simultaneously related to the basic unshifted QR algorithm for $R^T R$ *and* $R^T R$.

Using (9.7.36) 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 (9.7.36) 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 (9.7.36) 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

$$R_{k+2}R_{k+2}^T = R_{k+2}R_{k+1}Q_{k+2} = Q_{k+2}^T(R_kR_k^T)Q_{k+2}. \tag{9.7.37}$$

which shows that we are simultaneously performing an iteration on $R_kR_k^T$, again without explicitly forming this matrix.

One iteration of (9.7.36) is equivalent to one iteration of the Cholesky LR algorithm applied to $B_k = R_kR_k^T$. This follows since B_k has the Cholesky factorization $B_k = R_{k+1}^TR_{k+1}$ 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_1 often are surprisingly good approximations to the singular values of A.

9.7.6 The QR-SVD algorithm for Bidiagonal Matrices

For the QR-SVD algorithm to be efficient it is necessary to initially reduce A to a compact form that is preserved during the QR iterations and to introduce shifts. The proper compact form here is a bidiagonal form B. It was described in Section 8.6.6 how any matrix $A \in \mathbf{R}^{m\times n}$ can be reduced to upper bidiagonal form. Performing this reduction on R we have

$$Q_B^TRP_B = \begin{pmatrix} q_1 & e_2 & & & \\ & q_2 & e_3 & & \\ & & \ddots & \ddots & \\ & & & q_{n-1} & e_n \\ & & & & q_n \end{pmatrix}. \tag{9.7.38}$$

with orthogonal transformations from left and right. Using a sequence of Householder transformations

$$Q_B = Q_1\cdots Q_n \in \mathbf{R}^{n\times n}, \qquad P_B = P_1\cdots P_{n-2} \in \mathbf{R}^{n\times n}.$$

the reduction can be carried out in $\frac{4}{3}n^3$ flops. Note that also a complex matrix A can be reduced to *real* bidiagonal form using complex Householder transformations, see Section 9.1.2. The singular values of B equal those of A and the left and right singular vectors can be constructed from those of B.

We first notice that if in (9.7.38) $e_i = 0$, 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, $G_{i,i+1}, G_{i,i+2}, \ldots, G_{i,n}$ 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 $q_1, q_i, e_i,\ i = 2,\ldots,n$ are zero. This assumption implies that the matrix B^TB has nondiagonal elements $\alpha_{i+1} = q_ie_{i+1} \neq 0$, and hence is unreduced. It follows that all eigenvalues of B^TB are positive and distinct, and we have $\sigma_1 > \cdots > \sigma_n > 0$.

If the basic unshifted QR-SVD algorithm (9.7.36) is applied to the bidiagonal matrix we obtain with $B_1 = B$ an iteration

$$B_k^T = Q_{k+1} B_{k+1}, \quad k = 0, 1, 2, \ldots. \tag{9.7.39}$$

In each step the lower bidiagonal matrix B_k^T is transformed into an upper bidiagonal matrix B_{k+1}. Each iteration can be performed with a sequence of $n-1$ Givens rotations at a cost of only $O(n)$ operations. However, the drawback is that it is not possible to include shifts. Since shifts are essential for achieving rapid convergnce and deflation we now look into alternative ways of implementing the QR-SVD algorithm.

We first proceed by forming the symmetric matrix

$$C = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix} \in \mathbf{R}^{2n \times 2n}. \tag{9.7.40}$$

whose eigenvalues are $\pm\sigma_i$, $i = 1, \ldots, n$. After reordering rows and columns by an odd/even permutation C becomes symmetric tridiagonal matrix with zeros on the main diagonal. Hence

$$T = P^T C P = \begin{pmatrix} 0 & q_1 & & & & \\ q_1 & 0 & e_2 & & & \\ & e_2 & 0 & q_2 & & \\ & & q_2 & 0 & \ddots & \\ & & & \ddots & \ddots & q_n \\ & & & & q_n & 0 \end{pmatrix} \tag{9.7.41}$$

where 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 Problem 6) 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.

A closer inspection of the QR algorithm applied to T reveals it to be equivalent to an algorithm where the iterations are carried out directly on B. It is also equivalent to an *implicit* version of the QR algorithm applied to the symmetric tridiagonal matrix $T = B^T B$.

We now consider the application of the implicit shift QR algorithm to $B^T B$. Since forming $B^T B$ would lead to a severe loss of accuracy in the small singular values it is essential to work directly with the matrix B. The Wilkinson shift τ can be determined as the smallest eigenvalue of the lower right 2×2 submatrix in BB^T, or equivalently as the square of the smallest singular value of the 2×2 upper triangular submatrix in (9.7.38)

$$\begin{pmatrix} q_{n-1} & e_n \\ 0 & q_n \end{pmatrix},$$

see Algorithm 9.4.2.

In the implicit shift QR algorithm for B^TB we first determine a Givens rotation $T_1 = G_{12}$ so that

$$G_{12}^T t_1 = \pm\|t_1\|_2 e_1, \qquad t_1 = (q_1^2 - \tau, q_1 e_2, 0, \ldots, 0)^T, \tag{9.7.42}$$

where t_1 is the first column in $B^TB - \tau I$ and τ is the shift. Suppose we next apply a sequence of Givens transformations such that

$$T_{n-1}^T \cdots T_2^T T_1^T B^T B T_1 T_2 \cdots T_{n-1}$$

is tridiagonal, but we wish to avoid doing this explicitly. Let us start by applying the transformation T_1 to B. Then we get (take $n = 5$),

$$BT_1 = \begin{pmatrix} \times & \times & & & \\ + & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{pmatrix}.$$

If we now premultiply by a Givens rotation $S_1^T = R_{12}$ 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 +:

$$S_1^T BT_1 = \begin{pmatrix} \times & \times & + & & \\ 0 & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{pmatrix}, \quad S_1^T BT_1T_2 = \begin{pmatrix} \times & \times & 0 & & \\ & \times & \times & & \\ & + & \times & \times & \\ & & & \times & \times \\ & & & & \times \end{pmatrix}.$$

We can now continue to chase the element + down, with transformations alternately from the right and left until we get a new bidiagonal matrix

$$\hat{B} = (S_{n-1}^T \cdots S_1^T) B (T_1 \cdots T_{n-1}) = U^T B P.$$

But then the matrix

$$\hat{T} = \hat{B}^T \hat{B} = P^T B^T U U^T B P = P^T T P$$

is tridiagonal, where the first column of P equals the first column of T_1. Hence if $\hat{T}$ is unreduced it must be the result of one QR iteration on $T = B^TB$ with shift equal to τ.

The subdiagonal entries of T equal $q_i e_{i+1}$, $i = 1, \ldots, n-1$. 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 premultiplication by a sequence Givens transformations $R_{i,i+1}, \ldots, R_{i,n}$, and the matrix then splits as above. In practice two convergence criteria are used. After each QR step if

$$|e_{i+1}| \leq 0.5u(|q_i| + |q_{i+1}|),$$

where u is the machine unit, we set $e_{i+1} = 0$. We then find the smallest p and the largest q such that B splits into quadratic subblocks

$$\begin{pmatrix} B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \end{pmatrix},$$

of dimensions $p, n-p-q$ and, q where B_3 is diagonal and B_2 has a nonzero subdiagonal. Second, if diagonal elements in B_2 satisfy

$$|q_i| \leq 0.5u(|e_i| + |e_{i+1}|),$$

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 e_{i+1} equal to zero. Also, the error introduced by the tests is not larger than some constant times $u\|B\|_2$.

The implicit QR-SVD algorithm can be shown to be backward stable. This essentially follows form the fact that we have only applied a sequence of orthogonal transformations to A. Hence the computed singular values $\bar{\Sigma} = \text{diag}\,(\bar{\sigma}_k)$ are the exact singular values of a nearby matrix $A + E$, where $\|E\|_2 \leq c(m,n) \cdot u\sigma_1$. Here $c(m,n)$ is a constant depending on m and n and u the machine unit. From Theorem 7.3.4

$$|\bar{\sigma}_k - \sigma_k| \leq c(m,n) \cdot u\sigma_1.$$

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.

When all the superdiagonal elements in B have converged to zero we have $Q_S^T B T_S = \Sigma = \text{diag}\,(\sigma_1, \ldots, \sigma_n)$. Hence

$$U^T A V = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}, \quad U = Q_B \text{diag}(Q_S, I_{m-n}), \quad V = T_B T_S \tag{9.7.43}$$

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 singular vectors are desired, the cost of a QR iteration goes up to $4n^2$ flops and the overall cost to $O(n^3)$. See Table 9.7.6 for a comparison of flop counts for different variants.

To reduce the number of QR iterations where we accumulate transformations we can first compute the singular values without accumulating vectors. If we then choose shifts based on the computed singular values, the *perfect shifts*, convergence occurs in one iteration. This may reduce the cost about 40%. If fewer than 25% of the singular vectors are wanted, then inverse iteration should be used instead. The drawback of this approach is the difficulty of getting orthogonal singular vectors to clustered singular values.

Table 9.7.1. *Comparison of multiplications for SVD algorithms.*

Required	Golub–Reinsch SVD	Chan SVD
Σ, U_1, V	$(3+C)mn^2 + \frac{11}{3}n^3$	$3mn^2 + 2(C+1)n^3$
Σ, U_1	$(3+C)mn^2 - n^3$	$3mn^2 + (C+4/3)n^3$
Σ, V	$2mn^2 + Cn^3$	$mn^2 + (C+5/3)n^3$
Σ	$2mn^2 - 2n^3/3$	$mn^2 + n^3$

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. For example, if in the Chan-SVD column pivoting is used in the initial QR decomposition, then the matrix is usually graded from large at upper left to small at lower right as illustrated below

$$\begin{pmatrix} 1 & 10^{-1} & & \\ & 10^{-2} & 10^{-3} & \\ & & 10^{-4} & 10^{-5} \\ & & & 10^{-6} \end{pmatrix}.$$

From the following perturbation result it follows that it should be possible to compute all singular values of a bidiagonal matrix to *full relative precision independent of their magnitudes.*

Theorem 9.7.5. (Demmel and Kahan [6, 1990])

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 $\eta = (2n-1)\omega < 1$,

$$|\bar{\sigma}_i - \sigma_i| \leq \frac{\eta}{1-\eta}|\sigma_i|, \tag{9.7.44}$$

$$\max\{\sin\theta(u_i, \tilde{u}_i), \sin\theta(v_i, \tilde{v}_i)\} \leq \frac{\sqrt{2}\eta(1+\eta)}{\mathrm{relgap}_i - \eta}, \tag{9.7.45}$$

$i = 1, \ldots, n$, where the **relative gap** *between singular values is*

$$\mathrm{relgap}_i = \min_{j\neq i} \frac{|\sigma_i - \sigma_j|}{\sigma_i + \sigma_j}. \tag{9.7.46}$$

The QR algorithm as described above 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. To avoid this the rows and columns of B could in this case be reversed before the QR algorithm is applied. Alternatively many algorithms check for the direction of grading. Note that the matrix may break up into diagonal blocks which are graded in different ways.

To compute small singular values accurately one can use the QR algorithm without shifts. The QR-iteration can be implemented using (9.7.39) with only $4n$ multiplications. This can be compared to the QR algorithm with nonzero shifts, which requires $12n$ multiplications and $4n$ additions. Moreover the computations can be arranged so that no subtractions are used, and hence each entry of the transformed matrix is computed to high *relative* accuracy. The implementation of this algorithm has been studied in depth by Fernando and Parlett [7].

Review Questions

1. What is meant by a graded matrix, and what precautions need to be taken when transforming such a matrix to condensed form?
2. For a certain class of symmetric matrices small eigenvalues are determined with a very small error compared to $\|A\|_F$. Which?
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. What are the advantages of the implicit shift version of the QR algorithm for a real Hessenberg matrix H?
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.
6. (a) Show that the symmetry of a matrix is preserved during the QR algorithm. What about normality?

   (b) Show that the Hessenberg form is preserved during the QR algorithm.
7. What condensed form is usually chosen for the singular value decomposition? What kind of transformations are used for bringing the matrix to condensed form? How are the singular values computed for the condensed form?

Problems

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 $-\sin^3\theta$.

2. Perform one step of the shifted QR algorithm on the matrix

$$T = \begin{pmatrix} a & b & 0 \\ b & d & \epsilon \\ 0 & \epsilon & h \end{pmatrix}.$$

Show, that with the shift $s = h$, the first step in the reduction to upper triangular form gives a matrix of the form

$$G_{12}(T - sI) = \begin{pmatrix} \hat{a} & \hat{b} & s_1\epsilon \\ 0 & \hat{d} & c_1\epsilon \\ 0 & \epsilon & 0 \end{pmatrix}, \qquad G_{12} = \begin{pmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Further show that

$$\tilde{T} = Q^T(T - sI)Q + sI, \qquad \tilde{t}_{32} = \tilde{t}_{23} = c_1\epsilon^3/(\epsilon^2 + \hat{d}^2),$$

that is, if $\epsilon \ll \hat{d}$ the QR method tends to converge cubically.

3. Let T be the tridiagonal matrix in (9.7.30), 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 $|\tilde{\beta}_n| \le |\beta_n|^3/\gamma^2$.

4. Show that a complex matrix A can be reduced to *real* bidiagonal form using a sequence of unitary Householder transformations, see (9.6.2)–(9.6.3)

5. Let C be the matrix in (9.7.40) and P the permutation matrix whose columns are those of the identity matrix in the order $(n+1, 1, n+2, 2, \ldots, 2n, n)$. Show that the matrix P^TCP becomes a tridiagonal matrix T of the form in (9.7.41).

6. 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 [16, 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!

7. (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 9.6.1.

8. (a) Develop an algorithm for computing singular values by applying Algorithm 9.6.6 for spectrum slicing to the special symmetric tridiagonal matrix T in (9.7.41). Given the elements $q_1, \ldots, q_n$ and $e_2, \ldots, e_n$ of T, the algorithm should generate the number π of singular values of T greater than a given value $\sigma > 0$. The algorithm should be simplified by taking advantage of the zero diagonal.

(b) Show that one slice requires only of the order $2n$ flops.

9.8 Subspace Methods for Large Eigenvalue Problems

In many applications eigenvalue problems arise involving matrices so large that they cannot be conveniently treated by the methods described so far. For such problems, it is not reasonable to ask for a complete set of eigenvalues and eigenvectors, and usually only some extreme eigenvalues (often at one end of the spectrum) are required. In the 1980's typical values could be to compute 10 eigenpairs of a matrix of order 10,000. In 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.

9.8.1 The Rayleigh–Ritz Procedure

Let $\mathcal{S}$ be the subspace of $\mathbf{R}^n$ spanned by the columns of a given matrix $S = (s_1, \ldots, s_m) \in \mathbf{R}^{n \times m}$ (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 9.8.1.

Let A be Hermitian and $Q \in \mathbf{R}^{n\times p}$ be orthonormal, $Q^HQ = I_p$. Then the residual norm $\|AQ - QC\|_2$ is minimized for $C = M$ where

$$M = \rho(Q) = Q^HAQ \tag{9.8.1}$$

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

$$|\lambda_{ij} - \theta_j| \le \|AQ - QM\|_2, \quad j = 1, \dots, p. \tag{9.8.2}$$

Proof. See Parlett [41, Section 11-5]. □

We can now outline the complete procedure:

Algorithm 9.8.1

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
$$M = Q^H(AQ) \in \mathbf{R}^{m\times m}. \tag{9.8.3}$$
3. Compute the $p \le m$ eigenpairs of the Hermitian matrix M which are of interest
$$Mz_i = \theta_1 z_i, \qquad i = 1, \dots, p. \tag{9.8.4}$$
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
$$r_i = Ay_i - y_i\theta_i = (AQ)z_i - y_i\theta_i. \tag{9.8.5}$$
Then each interval
$$\big[\theta_i - \|r_i\|_2, \theta_i + \|r_i\|_2\big], \qquad i = 1, \dots, p, \tag{9.8.6}$$
contains an eigenvalue λ_i of A.

The pairs (θ_i, y_i), $i = 1, \dots, p$ are the best approximate eigenpairs of A which can be derived from the space $\mathcal{S}$. If some of the intervals in (9.8.6) overlap, we cannot be sure to have approximations to p eigenvalues of A. However, there are always p eigenvalues in the intervals defined by (9.8.2).

We can get error bounds for the approximate eigenspaces from an elegant generalization of Theorem 9.3.16. We first need to define the **gap** of the spectrum of A with respect to a given set of approximate eigenvalues.

Definition 9.8.2.

Let $\lambda(A) = \{\lambda_1, \ldots, \lambda_n\}$ be eigenvalues of a Hermitian matrix A. For the set $\rho = \{\theta_1, \ldots, \theta_p\}$, let $s_\rho = \{\lambda_{i_1}, \ldots, \lambda_{i_p}\}$ be a subset of $\lambda(A)$ minimizing $\max_j |\theta_j - \lambda_{i_j}|$. Then we define

$$\text{gap}\,(\rho) = \min_{\lambda \in \lambda(A)} |\lambda - \theta_i|, \quad \lambda \notin s_\rho, \quad \theta_i \in \rho. \tag{9.8.7}$$

Theorem 9.8.3.

Let $Q \in \mathbf{R}^{n\times p}$ be orthonormal and A a Hermitian matrix. Let $\{\theta_1, \ldots, \theta_p\}$ be the eigenvalues of $H = \rho(Q) = Q^H A Q$, and let $s_r = \{\lambda_{i_1}, \ldots, \lambda_{i_p}\}$ 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

$$\theta(\mathcal{Q}, \mathcal{Z}) \le \|AQ - QH\|_2 / \text{gap}\,(\rho). \tag{9.8.8}$$

where $\sin\theta(\mathcal{Q}, \mathcal{Z})$ is the largest angle between the subspaces $\mathcal{Q}$ and $\mathcal{Z}$.

9.8.2 Subspace Iteration for Hermitian Matrices

In Section 9.5.4 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

$$Z_k = AQ_{k-1}, \qquad Q_k R_k = Z_k, \qquad k = 1, 2, \ldots, \tag{9.8.9}$$

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 (9.8.9) 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 Section 9.5.4) that if A has p dominant eigenvalues $\lambda_1, \cdots, \lambda_p$, i.e.,

$$|\lambda_1| \ge \cdots \ge |\lambda_p| > |\lambda_{p+1}| \ge \cdots \ge |\lambda_n|$$

then the subspaces $\mathcal{S}_k$, $k = 0, 1, 2, \ldots$ converge almost always to the corresponding dominating invariant subspace.The convergence is linear with rate $|\lambda_{p+1}/\lambda_p|$.

For the individual eigenvalues $\lambda_i > \lambda_{i+1}$, $i \le p$, it holds that

$$|r_{ii}^{(k)} - \lambda_i| = O(|\lambda_{i+1}/\lambda_i|^k), \qquad i = 1, \ldots, p.$$

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 9.8.2

Orthogonal Iteration, Hermitian Case.

With $Q_0 \in \mathbf{R}^{n\times p}$ compute for $k = 1, 2, \ldots$ 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

$$|\theta_i^{(k)} - \lambda_i| = \mathrm{O}(|\lambda_{p+1}/\lambda_i|^k), \qquad \Theta_k = \mathrm{diag}\,(\theta_1^{(k)}, \ldots, \theta_p^{(k)}),$$

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 9.8.1.

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 9.8.2 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 $p^2 n$ flops, respectively.

Note that the same subspace $\mathcal{S}_k$ is generated by k consecutive steps of 1, as with the complete Algorithm 9.8.2. 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

$$r_i = Aq_i^{(k)} - q_i^{(k)}\theta_i = (AQ_k)u_i^{(k)} - q_i^{(k)}\theta_i. \tag{9.8.10}$$

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 [29].

Algorithm 9.8.2 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 Section 9.5.2)

$$\hat{A} = (A - \mu I)^{-1}.$$

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

$$(A - \mu I)p = q. \tag{9.8.11}$$

This can be done, e.g., by first computing an LU factorization of $A - \mu I$ or by an iterative method.

9.8.3 Krylov Subspaces

Of great importance for iterative methods are the subspaces of the form

$$\mathcal{K}_m(v, A) = \operatorname{span}\big(v, Av, \ldots, A^{m-1}v\big), \tag{9.8.12}$$

generated by a matrix A and a single vector v. These are called **Krylov subspaces** and the corresponding matrix

$$K_m = \big(v, Av, \ldots, A^{m-1}v\big)$$

is called a Krylov matrix. If $m \leq n$ the dimension of $\mathcal{K}_m$ usually equals m unless v is specially related to A.

The 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 9.8.4.

Assume that A is diagonalizable and let the initial vector v have the expansion

$$v = \sum_{k=1}^{n} \alpha_k u_k \tag{9.8.13}$$

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:

$$\|u_i - \Pi_k u_i\|_2 \leq \xi_i \epsilon_i^{(k)}, \quad \xi_i = \sum_{j \neq i} |\alpha_j|/|\alpha_i|, \tag{9.8.14}$$

where

$$\epsilon_i^{(k)} = \min_{p \in P_{k-1}} \max_{\lambda \in \lambda(A) - \lambda_i} |p(\lambda)|. \tag{9.8.15}$$

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 (9.8.13) of v it follows that for any polynomial $p \in P_{k-1}$ with $p(\lambda_i) = 1$ we have

$$\|(I - \Pi_k)\alpha_i u_i\|_2 \leq \left\| \alpha_i u_i - \sum_{j=1}^{n} \alpha_j p(\lambda_j) u_j \right\|_2 \leq \max_{j \neq i} |p(\lambda_j)| \left| \sum_{j \neq i} |\alpha_j| \right|.$$

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. □

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 $|T_k(x)| \leq 1$ for $|x| \leq 1$, and for $|x| \geq 1$ we have

$$T_k(x) = \frac{1}{2}\left[(x + \sqrt{x^2-1})^k + (x - \sqrt{x^2-1})^k\right]. \tag{9.8.16}$$

Now if we take

$$x = l_i(\lambda) = 1 + 2\frac{\lambda - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}, \quad \gamma_i = l_i(\lambda_i) = 1 + 2\frac{\lambda_i - \lambda_{i+1}}{\lambda_i - \lambda_n}. \tag{9.8.17}$$

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 9.8.4. When k is large we have

$$\epsilon_1^{(k)} \leq \max_{\lambda \in \lambda(A) - \lambda_i} |p(\lambda)| \leq \frac{1}{T_{k-1}(\gamma_1)} \approx 2 \Big/ \left(\gamma_1 + \sqrt{\gamma_1^2 - 1}\right)^{k-1}. \tag{9.8.18}$$

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 $q_{i-1}(\lambda_j) = 0$, $j = 1, \ldots, i-1$, and $q_{i-1}(\lambda_i) = 1$. Further

$$\max_{\lambda \in \lambda(A) - \lambda_i} |q_{i-1}(\lambda)| \leq |q_{i-1}(\lambda_n)| = C_i. \tag{9.8.19}$$

Thus when k is large we have

$$\epsilon_i^{(k)} \leq C_i / T_{k-i}(\gamma_i). \tag{9.8.20}$$

This indicates that we can expect interior eigenvalues and eigenvectors to be less well approximated by Krylov-type methods.

9.8.4 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 [22, 1950], can be viewed as a way for reducing a symmetric matrix A to tridiagonal form $T = Q^T A Q$. Here $Q = (q_1, q_2, \ldots, q_n)$ is orthogonal, where q_1 can be chosen arbitrarily, and

$$T = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}. \tag{9.8.21}$$

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

$$Aq_j = \beta_j q_{j-1} + \alpha_j q_j + \beta_{j+1} q_{j+1}, \quad j = 1, \ldots, n-1.$$

where we have put $\beta_1 q_0 \equiv 0$. The requirement that $q_{j+1} \perp q_j$ gives

$$\alpha_j = q_j^T (A q_j - \beta_j q_{j-1}),$$

(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},

$$\beta_{j+1} q_{j+1} = r_{j+1}, \qquad r_{j+1} = A q_j - \alpha_j q_j - \beta_j q_{j-1},$$

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 9.8.3
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 $\mathcal{K}_k(q_1, A)$:

$$\begin{aligned} &r_0 = q_1;\ q_0 = 0; \\ &\beta_1 = \|r_0\|_2 = 1; \\ &\textbf{for } j = 1, 2, 3 \ldots \\ &\quad q_j = r_{j-1}/\beta_j; \\ &\quad r_j = A q_j - \beta_j q_{j-1}; \end{aligned}$$

$\alpha_j = q_j^T r_j;$
$r_j = r_j - \alpha_j q_j;$
$\beta_{j+1} = \|r_j\|_2;$
if $\beta_{j+1} = 0$ **then** *exit*;
end

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

$$AQ_k = Q_k T_k + \beta_{k+1} q_{k+1} e_k^T. \tag{9.8.22}$$

The Lanczos process stops if $\beta_{k+1} = 0$ since then q_{k+1} is not defined. However, then by (9.8.22) 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 $\text{span}(Q_k) = \mathcal{K}_k(A, b)$. Multiplying (9.8.22) 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, $i = 1, \ldots, k$. 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 (9.8.22) 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

$$\|Ay_i - y_i\theta_i\|_2 = \beta_{k+1} |e_k^T z_i|. \tag{9.8.23}$$

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 (9.8.23) 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 $q_k, \ldots, q_1$. 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 [25, 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.

9.8.5 Golub–Kahan 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

$$A = U \begin{pmatrix} B \\ 0 \end{pmatrix} V^T, \quad U^T U = I_m, \quad V^T V = I_n, \tag{9.8.24}$$

where $U = (u_1, \ldots, u_m)$ and $V = (v_1, \ldots, v_n)$ are chosen as products of Householder transformations and B is upper bidiagonal. If we set $U_1 = (u_1, \ldots, u_n)$ then from (9.8.24) we have

$$AV = U_1 B, \qquad A^T U_1 = V B^T. \tag{9.8.25}$$

In an alternative approach, given by Golub and Kahan [14, 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

$$B_n = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_n \\ & & & \beta_{n+1} \end{pmatrix} \in \mathbf{R}^{(n+1)\times n}. \tag{9.8.26}$$

(Note that B_n is not square.) Equating columns in (9.8.25) we obtain, setting $\beta_1 v_0 \equiv 0$, $\alpha_{n+1} v_{n+1} \equiv 0$, the recurrence relations

$$\begin{aligned} A^T u_j &= \beta_j v_{j-1} + \alpha_j v_j, \\ A v_j &= \alpha_j u_j + \beta_{j+1} u_{j+1}, \quad j = 1, \ldots, n. \end{aligned} \tag{9.8.27}$$

Starting with a given vector $u_1 \in \mathbf{R}^m$, $\|u_1\|_2 = 1$, we can now recursively generate the vectors $v_1, u_2, v_2, \ldots, u_{m+1}$ and corresponding elements in B_n using, for $j =$

$1, 2, \ldots$, the formulas

$$r_j = A^T u_j - \beta_j v_{j-1}, \quad \alpha_j = \|r_j\|_2, \quad v_j = r_j/\alpha_j, \tag{9.8.28}$$

$$p_j = A v_j - \alpha_j u_j, \qquad \beta_{j+1} = \|p_j\|_2, \quad u_{j+1} = p_j/\beta_{j+1}. \tag{9.8.29}$$

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, $i = 1, \ldots, n$, and that the corresponding eigenvectors equal the left and right singular vectors of A, respectively.

The GKBD process (9.8.28)–(9.8.29) generates in exact arithmetic the same sequences of vectors $u_1, u_2, \ldots$ and $v_1, v_2, \ldots$ 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.

9.8.6 Arnoldi's Method.

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 9.8.4
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 $V_k = (v_1, \ldots, v_k)$ with orthogonal columns spanning the Krylov subspace $\mathcal{K}_k(v_1, A)$:

for $j = 1 : k$

for $i = 1 : j$
$h_{ij} = v_i^H (A v_j)$;
end
$r_j = A v_j - \sum_{i=1}^{j} h_{ij} v_i$;
$h_{j+1,j} = \|r_j\|_2$;
if $h_{j+1,j} = 0$ **then** *exit*;
$v_{j+1} = r_j / h_{j+1,j}$;
end

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

$$A V_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^H, \tag{9.8.30}$$

$$V_k^H A V_k = H_k. \tag{9.8.31}$$

The process will break down at step j if and only if the vector r_j vanishes. When this happens we have $A V_k = V_k H_k$, and so $\mathcal{R}(V_k)$ is an invariant subspace of A. By (9.8.30) $H_k = V_k^H A V_k$ 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. (9.8.23))

$$\|(A - \theta_i I) y_i\|_2 = h_{m+1,m} |e_k^T z_i|. \tag{9.8.32}$$

The proof of this relation is left as an exercise.

Review Questions

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.
2. 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.

Problems

1. (To be added.)

9.9 Generalized Eigenvalue Problems

9.9.1 Introduction

In this section we consider the **generalized eigenvalue problem** of computing nontrivial solutions (λ, x) of

$$Ax = \lambda Bx, \tag{9.9.1}$$

where A and B are square matrices of order n. The family of matrices $A - \lambda B$ is called a **matrix pencil**.[6] It is called a **regular** pencil if $\det(A - \lambda B) \not\equiv 0$, else it is a **singular** pencil. 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

$$\det(A - \lambda B) = 0. \tag{9.9.2}$$

If the degree of the characteristic polynomial is $n - p$, then we say that $A - \lambda B$ has p eigenvalues at ∞.

Example 9.9.1.

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 $B - \lambda A$.

If S and T are nonsingular matrices then (9.9.2) is equivalent to

$$\det S(A - \lambda B)T = \det(SAT - \lambda SBT) = 0.$$

The two pencils $A - \lambda B$ and $SAT - \lambda SBT$ are said to be **equivalent**. They have the same eigenvalues and the eigenvectors are simply related.

If A and B are 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 Q is orthogonal. Such transformations are stable since they preserve the 2-norm,

$$\|Q^T A Q\|_2 = \|A\|_2, \qquad \|Q^T B Q\|_2 = \|B\|_2.$$

[6] The word "pencil" comes from optics and geometry, and is used for any one parameter family of curves, matrices, etc.

9.9.2 Canonical Forms

The algebraic and analytic theory of the generalized eigenvalue problem is much more complicated than for the standard problem, and a complete treatment is outside the scoop of this book. There is a canonical form for regular matrix pencils corresponding to the Jordan canonical form, Theorem 9.2.6, which we state without proof.

Theorem 9.9.1. Kronecker's Canonical Form.

Let $A - \lambda B \in \mathbf{C}^{n\times n}$ be a regular matrix pencil. Then there are nonsingular matrices $X, Z \in \mathbf{C}^{n\times n}$, such that $X^{-1}(A - \lambda B)Z = \hat{A} - \lambda\hat{B}$, where

$$\begin{aligned} \hat{A} &= \operatorname{diag}\left(J_{m_1}(\lambda_1), \ldots, J_{m_s}(\lambda_s), I_{m_{s+1}}, \ldots, I_{m_t}\right), \\ \hat{B} &= \operatorname{diag}\left(I_{m_1}, \ldots, I_{m_s}, J_{m_{s+1}}(0), \ldots, J_{m_t}(0)\right), \end{aligned} \tag{9.9.3}$$

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. There is also a generalization of the Schur Canonical Form (Theorem 9.2.1), which can be computed stably and more efficiently.

Theorem 9.9.2. 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 [1973, Ch. 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.

9.9.3 Reduction to Standard Form

When B is nonsingular the eigenvalue problem (9.9.1) 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 [41, Section 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 (9.9.1) 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 9.9.3.

Let A and B be real symmetric square matrices and B also positive definite. Then there exists a nonsingular matrix X such that

$$X^TAX = D_A, \qquad X^TBX = D_B \tag{9.9.4}$$

are real and diagonal. The eigenvalues of $A - \lambda B$ are given by

$$\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) = D_A D_B^{-1}.$$

Proof. Let $B = LL^T$ be the Cholesky factorization of B. Then

$$L^{-1}(A - \lambda B)L^{-T} = \tilde{A} - \lambda I, \qquad \tilde{A} = \tilde{A} = L^{-1}AL^{-T}, \tag{9.9.5}$$

where $\tilde{A}$ is real and symmetric. Let $\tilde{A} = Q^T D_A Q$ be the eigendecomposition of $\tilde{A}$. Then we have

$$X^T(A - \lambda B)X = D_A - \lambda D_B, \qquad X = (QL^{-1})^T,$$

and the theorem follows. □

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

$$\lambda = (\gamma\hat{\lambda} + \sigma)/(-\sigma\hat{\lambda} + \gamma). \tag{9.9.6}$$

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

$$\inf_{x \neq 0} \left((x^TAx)^2 + (x^TBx)^2 \right)^{1/2} > 0$$

then there exist such γ and σ.

Under the assumptions in Theorem 9.9.3 the symmetric pencil $A - \lambda B$ has n real roots. Moreover, the eigenvectors can be chosen to be B-orthogonal, i.e.,

$$x_i^TBx_j = 0, \qquad i \neq j.$$

This generalizes the standard symmetric case for which $B = I$.

Numerical methods can be based on the *explicit reduction to standard form* in (9.9.5). $Ax = \lambda Bx$ is then equivalent to $Cy = \lambda y$, where

$$C = L^{-1}AL^{-T}, \qquad y = L^Tx. \tag{9.9.7}$$

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, [41]. 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 [5] 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. 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 9.9.2. *Wilkinson and Reinsch [41, 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.

9.9.4 Methods for Generalized Eigenvalue Problems

We first note that the power method and inverse iteration can both be extended to the generalized eigenvalue problems. Starting with some q_0 with $\|q_0\|_2 = 1$, these iterations now become

$$\begin{aligned} B\hat{q}_k &= Aq_{k-1}, \quad q_k = \hat{q}_k/\|\hat{q}_k\|, \\ (A - \sigma B)\hat{q}_k &= Bq_{k-1}, \qquad q_k = \hat{q}_k/\|\hat{q}_k\|, \qquad k = 1, 2, \ldots \end{aligned}$$

respectively. Note that $B = I$ gives the iterations in equations (9.4.4) and (9.4.6). The Rayleigh Quotient Iteration also extends to the generalized eigenvalue problem: For $k = 0, 1, 2, \ldots$ compute

$$(A - \rho(q_{k-1})B)\hat{q}_k = Bq_{k-1}, \qquad q_k = \hat{q}_k/\|q_k\|_2, \tag{9.9.8}$$

where the (generalized) Rayleigh quotient of x is

$$\rho(x) = \frac{x^H A x}{x^H B x}.$$

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 9.9.4.
Let $A - \sigma B$ have the Cholesky factorization

$$A - \sigma B = LDL^T, \qquad D = \operatorname{diag}(d_1, \ldots, d_n),$$

where L is unit lower triangular. If B is positive definite then the number of eigenvalues of A greater than σ equals the number of positive elements $\pi(D)$ in the sequence $d_1, \ldots, d_n$.

Proof. The proof follows from Sylvester's Law of Inertia (Theorem 9.3.10) and the fact that by Theorem 9.9.1 A and B are congruent to D_A and D_B with $\Lambda = D_A D_B^{-1}$. □

For a nearly singular pencil (A, B) it may be preferable to use the **QZ algorithm** of Moler and Stewart which is a generalization of the implicit QR algorithm. Here the matrix A is first reduced to upper Hessenberg form H_A and simultaneously B to upper triangular form R_B using standard Householder transformations and Givens rotations. Infinite eigenvalues, which correspond to zero diagonal elements of R_B are then eliminated. Finally the implicit shift QR algorithm is applied to $H_A R_B^{-1}$, without explicitly forming this product. This is achieved by computing unitary matrices Q and Z such that QAZ is upper Hessenberg, QBZ upper triangular and choosing the first column of Q proportional to the first column of $H_A R_B^{-1} - \sigma I$. A double shift technique can also be used if A and B are real. 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 [32, 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.

9.9.5 The CS Decomposition.

The CS decomposition is a special case of the generalized SVD (GSVD) which is of interest in its own right.

Theorem 9.9.5. CS Decomposition. *Let $Q \in \mathbf{R}^{(m+p)\times n}$ have orthonormal columns, and be partitioned as*

$$Q = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} \begin{matrix} \}m \\ \}p \end{matrix} \in \mathbf{R}^{(m+p)\times n}, \qquad m \geq n, \tag{9.9.9}$$

i.e., $Q^TQ = Q_1^TQ_1 + Q_2^TQ_2 = I_n$. Then there are orthogonal matrices $U_1 \in \mathbf{R}^{m\times m}$, $U_2 \in \mathbf{R}^{p\times p}$, and $V \in \mathbf{R}^{n\times n}$, and square nonnegative diagonal matrices

$$C = \operatorname{diag}(c_1, \ldots, c_q), \quad S = \operatorname{diag}(s_1, \ldots, s_q), \qquad q = \min(n, p), \tag{9.9.10}$$

satisfying $C^2 + S^2 = I_q$ such that

$$\begin{pmatrix} U_1^T & 0 \\ 0 & U_2^T \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} V = \begin{pmatrix} U_1^TQ_1V \\ U_2^TQ_2V \end{pmatrix} = \begin{pmatrix} \Sigma_1 \\ \Sigma_2 \end{pmatrix} \begin{matrix} \}m \\ \}p \end{matrix} \tag{9.9.11}$$

has one of the following forms:

$$p \geq n: \quad \begin{pmatrix} C \\ 0 \\ S \\ 0 \end{pmatrix} \begin{matrix} \}n \\ \}m-n \\ \}n \\ \}p-n \end{matrix}, \qquad p < n: \quad \begin{pmatrix} C & 0 \\ 0 & I \\ 0 & 0 \\ \underbrace{S}_{p} & \underbrace{0}_{n-p} \end{pmatrix} \begin{matrix} \}p \\ \}n-p \\ \}m-n \\ \}p \end{matrix}.$$

The diagonal elements c_i and s_i are

$$c_i = \cos(\theta_i), \quad s_i = \sin(\theta_i), \quad i = 1, \ldots, q,$$

where without loss of generality, we may assume that

$$0 \leq \theta_1 \leq \theta_2 \leq \cdots \leq \theta_q \leq \pi/2.$$

Proof. To construct U_1, V, and C, note that since U_1 and V are orthogonal and C is a nonnegative diagonal matrix, (9.9.11) is the SVD of Q_1. Hence the elements c_i are the singular values of Q_1. If we put $\tilde{Q}_2 = Q_2V$, then the matrix

$$\begin{pmatrix} C \\ 0 \\ \tilde{Q}_2 \end{pmatrix} = \begin{pmatrix} U_1^T & 0 \\ 0 & I_p \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} V$$

has orthonormal columns. Thus $C^2 + \tilde{Q}_2^T\tilde{Q}_2 = I_n$, which implies that $\tilde{Q}_2^T\tilde{Q}_2 = I_n - C^2$ is diagonal and hence the matrix $\tilde{Q}_2 = (\tilde{q}_1^{(2)}, \ldots, \tilde{q}_n^{(2)})$ has orthogonal columns.

We assume that the singular values $c_i = \cos(\theta_i)$ of Q_1 have been ordered according to (9.9.5) and that $c_r < c_{r+1} = 1$. Then the matrix $U_2 = (u_1^{(2)}, \ldots, u_p^{(2)})$ is constructed as follows. Since $\|\tilde{q}_j^{(2)}\|_2^2 = 1 - c_j^2 \neq 0, \quad j \leq r$ we take

$$u_j^{(2)} = \tilde{q}_j^{(2)} / \|\tilde{q}_j^{(2)}\|_2, \qquad j = 1, \ldots, r,$$

and fill the possibly remaining columns of U_2 with orthonormal vectors in the orthogonal complement of $\mathcal{R}(\tilde{Q}_2)$. From the construction it follows that $U_2 \in \mathbf{R}^{p\times p}$ is orthogonal and that

$$U_2^T \tilde{Q}_2 = U_2 Q_2 V = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}, \qquad S = \operatorname{diag}(s_1, \ldots, s_q)$$

with $s_j = (1 - c_j^2)^{1/2} > 0$, if $j = 1, \ldots, r$, and $s_j = 0$, if $j = r+1, \ldots, q$. □

In the theorem above we assumed that $m \geq n$. The general case gives rise to four different forms corresponding to cases where Q_1 and/or Q_2 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, 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.

9.9.6 The Generalized SVD.

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 $m \geq n$.

Theorem 9.9.6. 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

$$U_A^T A = \begin{pmatrix} D_A \\ 0 \end{pmatrix} Z, \qquad U_B^T B = \begin{pmatrix} D_B & 0 \\ 0 & 0 \end{pmatrix} Z, \tag{9.9.12}$$

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

$$0 \le \alpha_1 \le \cdots \le \alpha_k \le 1, \qquad 1 \ge \beta_1 \ge \cdots \ge \beta_q \ge 0,$$
$$\alpha_i^2 + \beta_i^2 = 1, \quad i = 1, \ldots, q, \quad \alpha_i = 1, \quad i = q+1, \ldots, k,$$

and the singular values of Z equal the nonzero singular values of M.

Proof. We now give a constructive proof of Theorem 9.9.6 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

$$\Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_k), \qquad \sigma_1 \ge \cdots \ge \sigma_k > 0.$$

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

$$\begin{pmatrix} A \\ B \end{pmatrix} P = \begin{pmatrix} AP_1 & 0 \\ BP_1 & 0 \end{pmatrix} = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} (\Sigma_1 \quad 0). \tag{9.9.13}$$

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 (9.9.13) we obtain

$$AP = U_A \begin{pmatrix} C \\ 0 \end{pmatrix} V^T (\Sigma_1 \quad 0),$$
$$BP = U_B \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} V^T (\Sigma_1 \quad 0),$$

and (9.9.12) follows with

$$D_A = C, \quad D_B = S, \quad Z = V^T (\Sigma_1 \quad 0)\, P^T.$$

Here $\sigma_1 \ge \cdots \ge \sigma_k > 0$ 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.

Review Questions

1. What is meant by a regular matrix pencil? Give examples of a singular pencil, and a regular pencil that has an infinite eigenvalue.
2. Formulate a generalized Schur Canonical Form. Show that the eigenvalues of the pencil are easily obtained from the canonical form.
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

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.

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

$$A_1 = A, \quad A_{k+1} = Q_k L_k^{-1} A_k L_k^{-T} Q_k^T, \quad k = 1, \ldots, n.$$

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

Complex Givens rotations and complex Householder transformations are treated in detail by Wilkinson [40, pp. 47–50]. For implementation details of complex Householder transformations, see the survey by R. B. Lehoucq [23, 1996].

For a more complete treatment of matrix functions see Chapter V in Gantmacher [11, 1959] and Lancaster [21, 1985]. Stewart and Sun [34] is a lucid treatise of matrix perturbation theory, with many historical comments and a very useful bibliography.

An analysis and a survey of inverse iteration for a single eigenvector is given by Ipsen [18]. The relation between simultaneous iteration and the QR algorithm and is explained in Watkins [38].

A still unsurpassed text on computational methods for the eigenvalue problem is Wilkinson [40, 1965]. Also the Algol subroutines and discussions in Wilkinson and Reinsch [41, 1971] are very instructive. An excellent discussion of the symmetric eigenvalue problem is given in Parlett [41, 1980]. Methods for solving large scale eigenvalue problems are treated by Saad [30, 1992].

A stable algorithm for computing the SVD based on an initial reduction to bidiagonal form was first sketched by Golub and Kahan in [14]. 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 [13]. The "final" form of the QR algorithm for computing the SVD was given by Golub and Reinsch [15]. The GSVD was first studied by Van Loan [16, 1996]. Paige and Saunders [26, 1981] extended the GSVD to handle all possible cases, and gave a computationally more amenable form.

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. [31, 1976], B. S. Garbow et al. [12, 1977], and E. Anderson et al. [1, 1999].

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, editors. *LAPACK Users' Guide. 3rd ed.* SIAM, Philadelphia, PA, 1999.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. A. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide.* SIAM, Philadelphia, PA, 2000.

[3] J. Barlow and J. Demmel. Computing accurate eigensystems of scaled diagonally dominant matrices. *SIAM J. Numer. Anal*, 27:762–791, 1990.

[4] S. Chandrasekaran and I. C. F. Ipsen. Analysis of a QR algorithm for computing singular values. *SIAM J. Matrix Anal. Appl.*, 16:2:520–535, 1995.

[5] C. R. Crawford. Reduction of a band-symmetric generalized eigenvalue problem. *Comm. ACM*, 16:41–44, 1973.

[6] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Statist. Comput.*, 11:873–912, 1990.

[7] K. V. Fernando and B. N. Parlett. Accurate singular values and differential qd algorithms. *Numer. Math.*, 67:191–229, 1994.

[8] R. Fletcher and D. C. Sorensen. An algorithmic derivation of the Jordan canonical form. *American Mathematical Monthly*, 90, 1983.

[9] G. E. Forsythe and P. Henrici. The cyclic jacobi method for computing the principal values of a complex matrix. *Tran. Amer. Math. Soc.*, 94:1–23, 1960.

[10] J. G. F. Francis. The QR transformation. Part I and II. *Computer J.*, 4:265–271, 332–345, 1961–1962.

[11] F. R. Gantmacher. *The Theory of Matrices. Vols. I and II.* Chelsea, New York, 1959.

[12] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and G. W. Stewart. *Matrix Eigensystems Routines: EISPACK Guide Extension.* Springer-Verlag, New York, 1977.

[13] G. H. Golub. Least squares, singular values and matrix approximations. *Aplikace Matematiky*, 13:44–51, 1968.

[14] G. H. Golub and W. Kahan. Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal. Ser. B*, 2:205–224, 1965.

[15] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solution. *Numer. Math*, 14:403–420, 1970.

[16] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[17] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal. *SIAM J. Matrix. Anal. Appl.*, 16:172–191, 1995.

[18] I. Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 39:254–291, 1997.

[19] W. Kahan. Accurate eigenvalues of a symmetric tri-diagonal matrix. Tech. Report No. CS41, Computer Science Department, Stanford University, CA, 1966.

[20] V. N. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *USSR Comput. Math. Phys.*, 3:637–657, 1961.

[21] P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, New York, 1985.

[22] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards, Sect. B*, 45:255–282, 1950.

[23] R. B. Lehoucq. The computations of elementary unitary matrices. *ACM Trans. Math. Software*, 22:393–400, 1996.

[24] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20:801–836, 1978.

[25] C. C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices.* PhD thesis, University of London, 1971.

[26] C. C. Paige and M. A. Saunders. Toward a generalized singular value decomposition. *SIAM J. Numer. Anal.*, 18:398–405, 1981.

[27] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Classics in Applied Mathematics 20. SIAM, Philadelphia, PA, 1998.

[28] H. Rutishauser. Solution of eigenvalue problems with the lr-transformation. *Nat. Bureau of Standards, Appl. Math. Ser.*, 49:47–81, 1958.

[29] H. Rutishauser. The Jacobi method for real symmetric matrices. *Numer. Math.*, 9:1–10, 1966.

[30] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Halstead Press, New York, 1992.

[31] 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.

[32] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, New York, 1973.

[33] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems.* SIAM, Philadelphia, PA, 2001.

[34] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, Boston, MA, 1990.

[35] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley, MA, 1988.

[36] L. N. Trefethen. Pseudospectra of linear operators. *SIAM Review*, 39:383–406, 1997.

[37] C. F. Van Loan. Generalizing the singular value decomposition. *SIAM J. Numer. Anal.*, 13:76–83, 1976.

[38] D. S. Watkins. Understanding the QR algorithm. *SIAM Review*, 24:427–440, 1982.

[39] D. S. Watkins. *Fundamentals of Matrix Computation.* Wiley-InterScience, New York, 2002.

[40] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, UK, 1965.

[41] J. H. Wilkinson and eds. C. Reinsch. *Handbook for Automatic Computation. Vol. II, Linear Algebra.* Springer-Verlag, New York, 1971.

Chapter 10

Iterative Methods for Linear Systems

10.1 Classical Iterative Methods

10.1.1 Introduction

The methods discussed so far for solving systems of linear equations $Ax = b$, have been direct methods based on matrix factorization and yielding the 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). 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 cyclic iterative methods started.

Iterative methods are used most often for the solution of very large linear systems. Basic iterative methods work directly with the original matrix A and only need extra storage for a few vectors. Since A is involved only in terms of matrix by vector products there is usually no need even to store the matrix A. Such methods are particularly useful for sparse system, 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, sometimes involving several million unknowns. The LU factors of matrices in such applications typically contain order of magnitudes more nonzero elements than A itself. Hence, because of the storage and number of arithmetic operations required, direct methods may become far too costly 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.

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 applica-

tions.

10.1.2 Relaxation Methods

Consider a linear system of equations $Ax = b$, where A is square and nonsingular. All iterative methods assume that an initial approximation $x^{(0)}$ is given (e.g., we could take $x^{(0)} = 0$) and by some rule compute a sequence of approximations $x^{(1)}$, $x^{(2)}, \ldots$. The iterative methods used before the age of high speed computers were usually rather unsophisticated. We describe here two classical, so called, **relaxation methods**.

Assume that A has nonzero diagonal entries, i.e, $a_{ii} \neq 0$, $i = 1, 2, \ldots, n$. 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

$$x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\Big), \qquad i = 1, 2 \ldots, n. \tag{10.1.1}$$

In a (minor) step of the iteration we pick one equation, say the ith, and then adjust the ith component of $x^{(k)}$ so that this equation becomes exactly satisfied. Hence, given $x^{(k)}$ we compute

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)}. \tag{10.1.2}$$

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 $i = 1, 2 \ldots, n$, in a cyclic fashion. The resulting iterative method is called the method of simultaneous displacements or **Jacobi's method**. 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**[1] differs from the Jacobi method by using new values $x_j^{(k+1)}$ as soon as they are available as follows:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)}, \quad i = 1, 2 \ldots, n. \tag{10.1.3}$$

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

[1] It was noted by G. Forsythe that Gauss nowhere mentioned this method and Seidel never advocated using it!

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 (10.1.1) and therefore the system $Ax = b$.

Example 10.1.1. Consider the symmetric, positive definite linear system

$$A = \begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \end{pmatrix}.$$

With the Jacobi method we get the following sequence of approximations:

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	0.25	0.5	0	0.25
2	0.375	0.625	0.125	0.375
3	0.4375	0.6875	0.1875	0.4375
4	0.46875	0.71875	0.21875	0.46875
5	0.48344	0.73438	0.23438	0.48344
	...	...	...	...
∞	0.5	0.75	0.25	0.5

If we instead use the Gauss–Seidel method we obtain:

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	0.25	0.5625	0.0625	0.40625
2	0.40625	0.70312	0.20312	0.47656
3	0.47656	0.73828	0.23828	0.49414
4	0.49414	0.74707	0.24707	0.49854
5	0.49854	0.74927	0.24927	0.49963
	...	...	...	...

In both cases the iteration converges, but rather slowly. Note that the convergence with Gauss–Seidel's method is, in this example, about twice as fast as with the Jacobi method. This will be shown in Section 10.3.1 to be true for an important class of matrices, so called consistently ordered matrices, see Def. 10.3.4. However, there are examples for which the Gauss–Seidel method diverges and the Jacobi method converges, and vice versa.

It was noted early that more rapid convergence could often be achieved by making the correction

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\gamma}{a_{ii}} r_i^{(k)},$$

with $\gamma > 1$ (over-relaxation) or $\gamma < 1$ under-relaxation).

Another classical iterative method is **Richardson's method**,[2] is defined by

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 0, 1, 2, \ldots, \tag{10.1.4}$$

where $\omega_k > 0$ are parameters to be chosen. It follows easily from (10.1.4) 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 $\omega = \omega_k$ we have

$$x^{(k)} - x = (I - \omega A)^k(x^{(0)} - x).$$

If A has a nonzero diagonal it can be scaled to have all diagonal elements equal to 1. In this case Richardson's method with $\omega = 1$ is equivalent to Jacobi's method.

The convergence of the three methods introduced here will be studied in Section 10.2.

10.1.3 A Model Problem

The biggest source of large linear systems is partial differential equations. An equation which is often encountered is Poisson's equation, where $u(x, y)$ satisfies

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f, \qquad (x, y) \in \Omega = (0, 1) \times (0, 1).$$

On the boundary Ω we assume $u(x, y)$ to be prescribed. We will frequently use this as a model problem.

To approximate the solution we impose a uniform square mesh of side $h = 1/n$ on Ω. Taking $f = 0$ (Laplace equation) and approximating the second derivatives by symmetric difference quotients gives a difference equation

$$\frac{1}{h^2}\big(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}\big) = 0, \qquad 0 < i, j < n,$$

for the unknown values u_{ij} at interior mesh points; see Section 13.1.2. If the mesh points are enumerated line by line (the so called "natural ordering") and a vector u is formed of the function values, the difference equation can then be written in matrix form as

$$Au = h^2 b, \qquad u = (u_1, u_2, \ldots, u_{n-1}),$$

where u_i is the vector of unknowns in the ith line and the matrix A is symmetric, with at most five nonzero elements in each row.

It can be verified that the nonzero elements of A lie on five diagonals. In block form the matrix can be written as

$$A = \begin{pmatrix} 2I + T & -I & & \\ -I & 2I + T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & 2I + T \end{pmatrix} \in \mathbf{R}^{(n-1)^2 \times (n-1)^2}, \tag{10.1.5}$$

[2] L. F. Richardson [1910]

where T is symmetric tridiagonal,

$$T = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbf{R}^{(n-1)\times(n-1)}. \tag{10.1.6}$$

Taking $n = 3$ we obtain the matrix used in Example 10.1.1.

It can be shown that in the Cholesky factorization $A = LL^T$ nearly all zero elements within the outermost nonzero diagonals of L will be filled in. Hence L contains about n^3 nonzero elements compared to only about $5n^2$ in A.

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 (10.1.6) the dimension is n^2 and the bandwidth equals n. Hence 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 growth 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 $T = \operatorname{trid}(c, a, b) \in \mathbf{R}^{n\times n}$ 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*

$$\lambda_i = a + 2\sqrt{bc}\cos\frac{i\pi}{n+1}, \quad i = 1, \ldots, n.$$

Further, the jth component of the eigenvector v_i corresponding to λ_i is

$$v_{ij} = \left(\frac{b}{c}\right)^{j/2} \sin\frac{ij\pi}{n+1}, \quad j = 1, \ldots, n.$$

From Lemma 10.1.1) it follows that the eigenvalues of T are $\lambda_i = 2(1 + \cos(i\pi/n))$, $i = 1, \ldots, n-1$. In particular we have that

$$\lambda_{\max} = 2(1 + \cos(\pi/n)) \approx 4, \qquad \lambda_{\min} = 2(1 - \cos(\pi/n)) \approx (\pi/n)^2.$$

Hence the condition number of T is approximately equal to $4n^2/\pi^2$.

The matrix $A = 4(I - L - U)$ in (10.1.5) can be written in terms of the Kronecker product (see Section 6.2.3) as

$$A = (I \otimes T) + (T \otimes I),$$

i.e., A is the Kronecker sum of T and T. It follows that the $(n-1)^2$ eigenvalues of A are $(\lambda_i + \lambda_j)$, $i, j = 1, \ldots, n-1$, 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.

10.2 Stationary Iterative Methods

The Jacobi, Gauss–Seidel, and Richardson methods are all special cases of a class of iterative methods, the general form of which is

$$Mx^{(k+1)} = Nx^{(k)} + b, \qquad k = 0, 1, \ldots. \tag{10.2.1}$$

Here

$$A = M - N \tag{10.2.2}$$

is a **splitting** of the matrix coefficient matrix A with M nonsingular. If the iteration (10.2.1) converges, i.e., $\lim_{k\to\infty} x^{(k)} = x$, then $Mx = Nx + b$ and it follows from (10.2.2) 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.

The iteration (10.2.1) is equivalent to

$$x^{(k+1)} = Bx^{(k)} + c, \qquad k = 0, 1, \ldots, \tag{10.2.3}$$

where

$$B = M^{-1}N = I - M^{-1}A, \qquad c = M^{-1}b.$$

An iteration of the form of (10.1.6) is called a (one-step) **stationary iterative method**, and B the **iteration matrix**. (In a non-stationary method the iteration matrix B depends on k.) Subtracting the equation $x = Bx + c$ from (10.1.6) we obtain the recurrence formula

$$x^{(k+1)} - x = B(x^{(k)} - x) \tag{10.2.4}$$

for the errors in successive approximations.

Richardson's method (10.1.4) can, for fixed $\omega_k = \omega$, be written in the form (10.2.1) with

$$M = I, \qquad N = B = I - \omega A.$$

To write the Jacobi and Gauss–Seidel methods in the form of one-step stationary iterative methods we introduce the **standard splitting**

$$A = D_A - L_A - U_A, \tag{10.2.5}$$

where $D_A = \operatorname{diag}(a_{11}, \ldots, a_{nn})$,

$$L_A = -\begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \quad U_A = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & 0 & a_{n-1,n} \\ & & & 0 \end{pmatrix},$$

and L_A and U_A are strictly lower and upper triangular, respectively. Assuming that $D_A > 0$, we can also write

$$D_A^{-1}A = I - L - U, \qquad L = D_A^{-1}L_A, \qquad U = D_A^{-1}U_A.$$

With these notations the Jacobi method, (10.1.2), can be written $D_A x^{(k+1)} = (L_A + U_A)x^{(k)} + b$ or

$$x^{(k+1)} = (L + U)x^{(k)} + c, \qquad c = D_A^{-1}b. \tag{10.2.6}$$

The Gauss–Seidel method, (10.1.3), 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

$$B_J = D_A^{-1}(L_A + U_A) = L + U,$$
$$B_{GS} = (D_A - L_A)^{-1}U_A = (I - L)^{-1}U.$$

10.2.1 Convergence Analysis

The iterative method (10.1.6) is called **convergent** if the sequence $\{x^{(k)}\}_{k=1,2,\ldots}$ converges for *all* initial vectors $x^{(0)}$. Of fundamental importance in the study of convergence of stationary iterative methods is conditions for a sequence of powers of a matrix to converge to the null matrix. For this we need some results from the theory of eigenvalues of matrices.

In Section 6.2.2 we introduced the spectral radius of a matrix A as the non-negative number

$$\rho(A) = \max_{1 \le i \le n} |\lambda_i(A)|.$$

We have the following important result:

Theorem 10.2.1. *A given matrix $B \in \mathbf{R}^{n\times n}$ is said to be* **convergent** *if $\rho(B) < 1$. It holds that*

$$\lim_{k\to\infty} B^k = 0 \quad \Leftrightarrow \quad \rho(B) < 1. \tag{10.2.7}$$

Proof. We will show that the following four conditions are equivalent:

(i) $\lim_{k\to\infty} B^k = 0,$

(ii) $\lim_{k\to\infty} B^k x = 0, \quad \forall x \in \mathbf{C}^n,$

(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). ∎

The following necessary and sufficient criterion for convergence of a stationary iterative method follows from Theorem 10.2.1.

Theorem 10.2.2. *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 (10.2.4) it follows that

$$x^{(k)} - x = B^k (x^{(0)} - x). \tag{10.2.8}$$

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 10.2.1. ∎

The problem of obtaining the spectral radius of B is usually no less difficult than solving the linear system. Hence the following upper bound is useful when trying to prove convergence.

Lemma 10.2.3.
For any matrix $A \in \mathbf{R}^{n\times n}$ and for any consistent matrix norm we have

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

Proof. Let λ be an eigenvalue of A such that $|\lambda| = \rho(A)$. Then $Ax = \lambda x$, $x \ne 0$, 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 10.2.3 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 (10.2.8) the error at step k, $e^{(k)} = x^{(k)} - x$, satisfies $e^{(k)} = B^k e^{(0)}$, we have for any consistent pair of norms

$$\|e^{(k)}\| \le \|B^k\|\,\|e^{(0)}\| \le \|B\|^k\|\,\|e^{(0)}\|.$$

Thus, to reduce the norm of the error by a given factor $\delta < 1$, it suffices to perform k iterations, where k is the smallest integer for which $\|B^k\| \le \delta$. Taking logarithms we obtain the condition

$$k \ge -\log\delta / R_k(B), \qquad R_k(B) = -\frac{1}{k}\log\|B^k\|.$$

An expression for the asymptotic rate follows from the relation

$$\rho(B) = \lim_{k\to\infty} (\|B^k\|)^{1/k},$$

which holds for any consistent matrix norm. This is a non-trivial result, but can be proved by using the Jordan normal form, see Problem 10.2.4.

This motivates the following definition:

Definition 10.2.4. *Assume that the iterative method* (10.1.6) *is convergent. For any matrix norm* $\|\cdot\|$ *we define the* **average rate** *of convergence by*

$$R_k(B) = -\frac{1}{k}\log\|B^k\|, \tag{10.2.9}$$

The corresponding **asymptotic rate** *of convergence is given by*

$$R_\infty(B) = \lim_{k\to\infty} R_k(B) = -\log\rho(B).$$

10.2.2 Convergence of Some Classic Methods

For fixed $\omega_k = \omega$ we have seen that Richardson's method is a stationary iterative method with iteration matrix $B = I - \omega A \in bfR^{n\times n}$.

Theorem 10.2.5.

Assume that the eigenvalues λ_i *of* A *are all real and satisfy*

$$0 < a \le \lambda_i \le b, \quad i = 1, \ldots, n.$$

Then Richardson's method is convergent for $0 < \omega < 2/b$.

Proof. The eigenvalues of $B = I - \omega A$ are $\mu_i = 1 - \lambda_i$ and thus satisfy $1 - \omega b \le \mu_i \le 1 - \omega a$. 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 $\omega < 2/b$. □

Assuming that $a = \lambda_{min}$ and $b = \lambda_{max}$, which value of ω will minimize the spectral radius

$$\rho(B) = \max\{|1 - \omega a|, |1 - \omega b|\}\,?$$

It is left as an exercise to show that the optimal ω satisfies $1 - \omega a = \omega b - 1$. (*Hint*: Plot the graphs of $|1 - \omega a|$ and $|1 - \omega b|$ for $\omega \in (0, 2/b)$.) Hence $\omega_{\text{opt}} = 2/(b + a)$, for which

$$\rho(B) = \frac{b - a}{b + a} = \frac{\kappa - 1}{\kappa + 1} = 1 - \frac{2}{\kappa + 1},$$

where $\kappa = b/a$ is the condition number of A. Note that for large values of κ the rate of convergence with ω_{opt} is proportional to κ^{-1}. This illustrates a typical fact for iterative methods: ill-conditioned systems require in general more work to achieve a certain accuracy!

Theorem 10.2.6. *The Jacobi method is convergent if A is strictly row-wise diagonally dominant, i.e.,*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n.$$

Proof. For the Jacobi method the iteration matrix $B_J = L + U$ has elements $b_{ij} = -a_{ij}/a_{ii}$, $i \neq j$, $b_{ij} = 0$, $i = j$. From the assumption it then follows that

$$\|B_J\|_\infty = \max_{1 \leq i \leq n} \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|/|a_{ii}| < 1.$$

□

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 10.2.6 is of importance in applications. (Note that, e.g., the matrix A in (10.1.1) is not strictly diagonal dominant!) For irreducible matrices (see Def. 10.1.1) the row sum criterion in Theorem 10.2.6 can be sharpened.

Theorem 10.2.7. *The Jacobi method is convergent if A is irreducible,*

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n,$$

and inequality holds for at least one row.

The column sum criterion can be similarly improved. The conditions in Theorem 10.2.6–10.2.7 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

$$\|B_{GS}\|_\infty \le \|B_{GS}\|_\infty \|L^T e_k\|_1 + \|U^T e_k\|_1.$$

Since A is strictly row-wise diagonally dominant we have $\|L^T e_k\|_1 + \|U^T e_k\|_1 \le \|B_J\|_\infty < 1$, and it follows that

$$\|B_{GS}\|_\infty \le \|U^T e_k\|_1 / \big(1 - \|L^T e_k\|_1\big) < 1.$$

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 10.2.1. 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 $(\lambda_i + \lambda_j)$, $i, j = 1, \ldots, n-1$, where $\lambda_i = 2(1 + \cos(i\pi/n))$. It follows that the eigenvalues of the corresponding Jacobi iteration matrix $B_J = L + U = (1/4)(A - 4I)$ are

$$\mu_{ij} = \frac{1}{2}(\cos i\pi h + \cos j\pi h), \quad i, j = 1, 2, \ldots, n-1,$$

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.[3] 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 explains the observation made in Example ex10.1.1 that Gauss–Seidel's method converged twice as fast as Jacobi's method. However, for both methods the number of iterations is proportional to $\kappa(A)$ for the model problem.

The rate of convergence of the Jacobi and Gauss–Seidel methods, as exhibited in the above example, is in general much too slow to make these methods of any practical use. In Section 10.3.1 we show how with a simple modification of the Gauss–Seidel method the rate of convergence can be improved by a factor of n for the model problem.

10.2.3 The Effect 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

[3] This is one of the basic observations used in the multigrid method, which uses a sequence of different meshes to efficiently damp all frequencies.

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

$$B = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}, \quad 0 < \mu \leq \lambda < 1, \quad \alpha \gg 1, \tag{10.2.10}$$

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

$$B^k = \begin{pmatrix} \lambda^k & \beta_k \\ 0 & \mu^k \end{pmatrix}, \quad \beta_k = \begin{cases} \alpha \dfrac{\lambda^k - \mu^k}{\lambda - \mu} & \text{if } \mu \neq \lambda; \\ \alpha k \lambda^{k-1} & \text{if } \mu = \lambda. \end{cases} \tag{10.2.11}$$

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:

Example 10.2.2. 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 $\rho(B) = 0.15$, and hence the iteration should converge to the exact solution of the equation $(I - B)x = 0$, which is $x = 0$. From Fig. 10.2.1 it is seen that $\|x^{(n)}\|_2$ increases by almost a factor 10^{15} until it starts to decrease after 25 iterations! Although in the long run the norm is reduced by about a factor of 0.5 at each iteration, large intermediate values of $x^{(n)}$ give rise to persistent rounding errors.

The curve in Figure 10.2.1 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 10.2.1 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

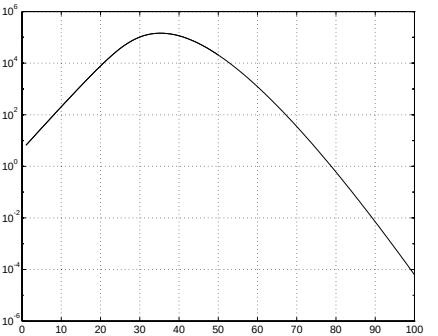


Figure 10.2.1. $\|x^{(n)}\|_2$, *where* $x^{(k+1)} = Bx^{(k)}$, *and* $x = (1, 1, \ldots, 1)^T$.

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 $x_2, \ldots, x_n$ 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 $a_{1i}(1+\delta_i)$, $i = 2, \ldots, n$. 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.

Example 10.2.3. (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 $0.963259\ldots$, but no progress is made towards the true solution $x_1 =$

$0.39473\ldots$, $x_2 = 0.62470\ldots$. The ill-conditioning has affected the computations adversely. Convergence is so slow that the modifications to be made in each step are less than $0.5 \cdot 10^{-5}$.

Stationary iterative methods may be badly affected by rounding errors for nonnormal matrices. We have seen that the "hump" phenomenon can make it possible for $\|x^{(k)}\|_2$ to increase substantially, even when the iteration matrix B is convergent; see Example 10.2.2. 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 as nonnormal matrix B asymptotic convergence in finite precision arithmetic is no longer guaranteed even if the condition $\rho(B) < 1$ is true 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 results is more useful than the exact result!

10.2.4 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

$$\|x - x^{(k)}\| \leq \epsilon \tag{10.2.12}$$

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 (10.2.12) may never be satisfied.

Instead of (10.2.12) 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

$$\|r^{(k)}\| \leq \epsilon \|b\|, \tag{10.2.13}$$

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 $x - x^{(k)} = A^{-1}r^{(k)}$, (10.2.13) 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

$$\omega = \max_i \frac{|r_i^{(k)}|}{(E|x^{(k)}| + f)_i}, \tag{10.2.14}$$

where $E > 0$ and $f > 0$, and stop when $\omega \leq \epsilon$. It then follows from Theorem 6.6.4 that $x^{(k)}$ is the exact solution to a perturbed linear system

$$(A + \delta A)x = b + \delta b, \quad |\delta A| \leq \omega E, \quad |\delta b| \leq \omega f.$$

We could in (10.2.14) 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

$$E = \|A\|_\infty e e^T, \quad f = \|b\|_\infty e, \quad e = (1, 1, \ldots, 1)^T.$$

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

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 $A^TAx = A^Tb$.
(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.

Computer Exercises

1. Let B be the 2×2 matrix in (10.2.11), and take $\lambda = \mu = 0.99$, $\alpha = 4$. Verify that $\|B^k\|_2 \ge 1$ for all $k < 805$!

2. Let $B \in \mathbf{R}^{20\times 20}$ be an upper bidiagonal matrix with diagonal elements equal to $0.025, 0.05, 0.075, \ldots, 0.5$ 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

$$x^{(k+1)} = Bx^{(k)}, \quad x^{(0)}0 = (1, 1, \ldots, 1)^T.$$

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.

10.3 Successive Overrelaxation Methods

10.3.1 The SOR Method

It was shown by Young [20] that for a certain class of matrices a great improvement in the rate of convergence can be obtained by the simple means of introducing a

relaxation parameter ω in the Gauss–Seidel method. This leads to the famous **Successive Over Relaxation (SOR) method**, which is derived from the Gauss–Seidel method by multiplying each increment $x_i^{(k+1)} - x_i^{(k)}$ by ω,

$$x_i^{(k+1)} = x_i^{(k)} + \omega \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, 2 \ldots, n, \quad (10.3.1)$$

where the expression in brackets is the current residual of the ith equation. For $\omega = 1$, the SOR method reduces to the Gauss–Seidel method. The idea is now to choose ω so that the asymptotic rate of convergence is maximized. The SOR method can be written in matrix form as

$$x^{(k+1)} = x^{(k)} + \omega \Big(c + L x^{(k+1)} - (I - U) x^{(k)} \Big),$$

where $c = D_A^{-1} b$, or after rearranging

$$(I - \omega L) x^{(k+1)} = [(1 - \omega) I + \omega U] x^{(k)} + \omega c.$$

The iteration matrix for SOR therefore is

$$B_\omega = (I - \omega L)^{-1} [(1 - \omega) I + \omega U]. \quad (10.3.2)$$

We now consider the convergence of the SOR method and first show that only values of ω, $0 < \omega < 2$ are of interest.

Theorem 10.3.1.

Let $B = L + U$ be any matrix with zero diagonal and let B_ω be the corresponding iteration matrix in the SOR method. Then we have

$$\rho(B_\omega) \geq |\omega - 1|, \quad (10.3.3)$$

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 $|\omega - 1|$. □

The following theorem asserts that if A is a positive definite matrix, then the SOR method converges if $0 < \omega < 2$.

Theorem 10.3.2. *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 10.5.4. □

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 10.3.3. *The matrix A is said to have property A if there exists a permutation matrix P such that PAP^T has the form*

$$\begin{pmatrix} D_1 & U_1 \\ L_1 & D_2 \end{pmatrix}, \tag{10.3.4}$$

where D_1, D_2 are diagonal matrices.

Equivalently, the matrix $A \in \mathbf{R}^{n\times n}$ has property A if the set $\{1, 2, \ldots, n\}$ can be divided into two non-void complementary subsets S and T such that $a_{ij} = 0$ unless $i = j$ or $i \in S$, $j \in T$, or $i \in T$, $j \in S$.

Example 10.3.1. The tridiagonal matrix A below

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}, \qquad P^TAP = \begin{pmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{pmatrix}$$

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 10.3.4. *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 (10.3.4) is consistently ordered, 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 hence have the same eigenvalues. Similarly one can prove more generally that 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 consistently ordered.

Theorem 10.3.5.

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

$$\mu = \frac{\lambda + \omega - 1}{\omega \lambda^{1/2}} \tag{10.3.5}$$

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 (10.3.2)

$$\det[(I - \omega L)(\lambda I - B_\omega)] = \det[\lambda(I - \omega L) - (1 - \omega)I - \omega U] = 0.$$

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 (10.3.5). Hence μ is an eigenvalue of $L + U$. □

If we put $\omega = 1$ in (10.3.1) 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 for consistently ordered matrices A the Gauss–Seidel method converges twice as fast as the Jacobi method.

We now state an important result due to Young [20].

Theorem 10.3.6.

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

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho_J^2}}. \tag{10.3.6}$$

For this optimal value we have

$$\rho(B_{\omega_{opt}}) = \omega_{opt} - 1. \tag{10.3.7}$$

Proof. (See also Young [21, 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,

$$f_\omega(\lambda) = \frac{\lambda + \omega - 1}{\omega}, \qquad g(\lambda, \mu) = \mu \lambda^{1/2}.$$

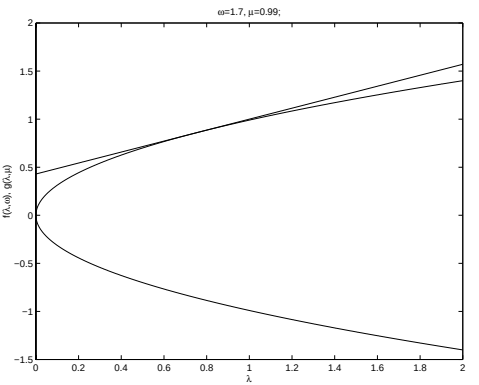


Figure 10.3.1. *$f_\omega(\lambda)$ and $g(\lambda,\mu)$ as functions of λ ($\mu = 0.99$, $\omega = \omega_b = 1.7527$).*

Here $f_\omega(\lambda)$ is a straight line passing through the points $(1,1)$ and $(1-\omega,0)$, and $g(\lambda,\mu)$ a parabola. The relation (10.3.5) can now be interpreted as the intersection of these two curves. For given μ and ω we get for λ the quadratic equation

$$\lambda^2 + 2\Big((\omega-1) - \frac{1}{2}\mu^2\omega^2\Big)\lambda + (\omega-1)^2 = 0. \tag{10.3.8}$$

which has two roots

$$\lambda_{1,2} = \frac{1}{2}\mu^2\omega^2 - (\omega-1) \pm \mu\omega\Big(\frac{1}{4}\mu^2\omega^2 - (\omega-1)\Big)^{1/2}.$$

The larger of these roots decreases with increasing ω until eventually $f_\omega(\lambda)$ becomes a tangent to $g(\lambda,\mu)$, when $\mu^2\omega^2/4 - (\omega-1) = 0$ (see Fig. 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 (10.3.8) 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 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. □

Example 10.3.2. By (10.3.6) for SOR $\omega_{opt} = 2/(1+\sin\pi h)$, giving

$$\rho(B_{\omega_{opt}}) = \omega_{opt} - 1 = \frac{1-\sin\pi h}{1+\sin\pi h} \approx 1 - 2\pi h. \tag{10.3.9}$$

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}.

Table 10.3.1. *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

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:

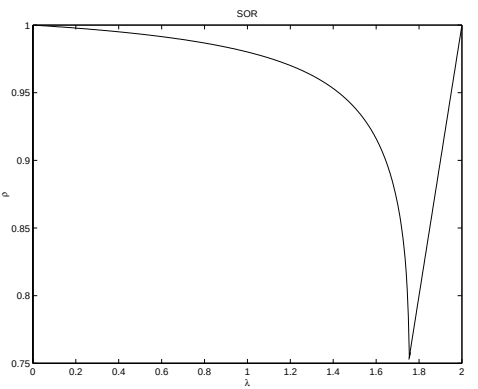


Figure 10.3.2. *The spectral radius* $\rho(B_\omega)$ *as a function of* ω *(*$\rho = 0.99$, $\omega_b = 1.7527$*).*

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 (10.3.6). 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 [21, p. 210].

In more complicated cases when ρ_J is not known, we have to estimate ω_{opt} in the SOR method. From Fig. 10.2.2 we conclude that it is much better to *overestimate* ω_{opt} than to underestimate it.

10.3.2 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

$$M_\omega = \frac{\omega}{2-\omega}\Big(\frac{1}{\omega}D_A - L_A\Big)D_A^{-1}\Big(\frac{1}{\omega}D_A - U_A\Big). \tag{10.3.10}$$

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 10.5.4.

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 that provided $\rho(LU) < 1/4$ a suitable value for ω is ω_b, where

$$\omega_b = \frac{2}{1+\sqrt{2(1-\rho_J)}}, \qquad \rho(S_{\omega_b}) \le \frac{1-\sqrt{(1-\rho_J)/2}}{1+\sqrt{(1-\rho_J)/2}}.$$

In particular, for the model problem in Section 10.1.3 it follows that

$$\rho(B_{\omega_b}) \le \frac{1-\sin\pi h/2}{1+\sin\pi h/2} \approx 1-\pi h.$$

This is half the rate of convergence for SOR with ω_{opt}.

10.3.3 Block Iterative Methods

The basic iterative methods described so far can be generalized for block matrices A. Assume that

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \dots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix},$$

where the diagonal blocks are square and nonsingular. For this block matrix we consider the splitting

$$A = D_A - L_A - U_A, \qquad D_A = \operatorname{diag}(A_{11}, A_{22}, \dots, A_{nn}),$$

and where L_A and U_A are strictly lower and upper triangular. As before Jacobi's method can be written $D_A x^{(k+1)} = (L_A + U_A)x^{(k)} + b$, or with x is partitioned conformally

$$A_{ii}\left(x_i^{(k+1)} - x_i^{(k)}\right) = b - \sum_{j=1}^{n} A_{ij} x_j^{(k)}, \quad i = 1, \dots, n.$$

Hence it is important that linear systems in the diagonal blocks A_{ii} can be solved efficiently.

Example 10.3.3. 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 (10.1.1). 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

$$A_{ii}\left(x_i^{(k+1)} - x_i^{(k)}\right) = \omega\left(b - \sum_{j=1}^{i-1} A_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} A_{ij} x_j^{(k)}\right), \quad i = 1, \dots, n.$$

(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 10.3.2 and 10.3.5 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 (10.3.6). 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, $i = 1, \ldots, n$ were diagonal. In particular we conclude that with the block splitting the matrix A in (10.1.1) is consistently ordered.

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.

Problems

1. (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 $(I + A)^k$, $k = 1, 2, 3, \ldots$ becomes a full matrix for some value of k?

2. The matrix A in (10.1.1) is block-tridiagonal, but its diagonal blocks are *not* diagonal matrices. Show that in spite of this the matrix is consistently ordered.

   Hint: Perform a similarity transformation with the diagonal matrix

$$D(\alpha) = \text{diag}\,(D_1(\alpha), D_2(\alpha), \ldots, D_n(\alpha)),$$

   where $D_1(\alpha) = \text{diag}\,(1, \alpha, \ldots, \alpha^{n-1})$, $D_{i+1}(\alpha) = \alpha D_i(\alpha)$, $i = 1, 2, \ldots, n-1$.

10.4 Convergence Acceleration

10.4.1 Nonstationary and Semi-iterative Methods

Consider the stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \qquad k = 0, 1, \ldots, \tag{10.4.1}$$

which corresponds to a matrix splitting $A = M - N$, and iteration matrix

$$B = M^{-1}N = I - M^{-1}A.$$

In this section we describe an important method for accelerating the convergence of the method (10.4.1) provided it is **symmetrizable**.

Definition 10.4.1. *The stationary iterative method* (10.4.1) *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 $I - B$ has real positive eigenvalues. 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 $M = W^T W$, and

$$W(I - B)W^{-1} = WM^{-1}AW^{-1} = WW^{-1}W^{-T}AW^{-1} = W^{-T}AW^{-1},$$

which again is positive definite.

Example 10.4.1. If A is positive definite then in the standard splitting (10.2.5) $D_A > 0$, and hence the Jacobi method is symmetrizable with $W = D_A^{1/2}$. From (10.3.10) it follows that also the SSOR method is symmetrizable.

It is often possible to find an associated method which will converge faster than the given method by taking a weighted arithmetic mean of the first k approximations generated by the method (10.4.1),

$$\tilde{x}^{(k)} = \sum_{i=0}^{k} \gamma_{ki} x^{(i)}, \qquad \sum_{i=0}^{k} \gamma_{ki} = 1, \quad k = 0, 1, 2, \ldots, \tag{10.4.2}$$

Such methods, introduced by Varga (1957), are non-stationary and called **semi-iterative methods**. It follows from the error equation $x^{(k)} - x = B^k(x^{(0)} - x)$ that

$$\tilde{x}^{(k)} - x = \sum_{i=0}^{k} \gamma_{ki}(x^{(i)} - x) = \sum_{i=0}^{k} \gamma_{ki} B^i (x^{(0)} - x).$$

Introducing the generating polynomial we can write this

$$\tilde{x}^{(k)} - x = p_k(B)(x^{(0)} - x), \qquad p_k(\lambda) = \sum_{i=0}^{k} \gamma_{ki} \lambda^i, \tag{10.4.3}$$

where $p_k(\lambda)$ is a polynomial of degree k. Therefore this procedure is also known as **polynomial acceleration**. Note that from (10.4.2) it follows that $p_k(1) = 1$. In the special case that $M = I$ we obtain for the residual $\tilde{r}^{(k)} = b - A\tilde{x}^{(k)}$

$$\tilde{r}^{(k)} = A(x - \tilde{x}^{(k)}) = q_k(A)r^{(0)}, \qquad q_k(\lambda) = p_k(1 - \lambda). \tag{10.4.4}$$

where we have used that $A = I - B$. The polynomials $q_k(\lambda)$ have the property that $q_k(0) = 1$ and are known as **residual polynomials**.

Example 10.4.2. Consider the non-stationary Richardson iteration,

$$x^{(i+1)} = x^{(i)} + \omega_i(b - Ax^{(i)}), \quad i = 1, 2, \ldots,$$

(cf. (10.1.4)). It is easily seen that the residual vector $r^{(k)} = b - Ax^{(k)}$ satisfies

$$r^{(k)} = q_k(A)r^{(0)}, \qquad q_k(\lambda) = \prod_{i=0}^{k-1}(I - \omega_i\lambda). \tag{10.4.5}$$

Hence, by choosing a suitable sequence of parameters $\{\omega_i\}_{i=0}^{k-1}$ we can obtain any residual polynomial $q_k(\lambda)$. If the spectrum of A is real, $a < \lambda(A) < b$, we would like $|q_k(\lambda)|$ to be small in (a, b).

10.4.2 Chebyshev Acceleration

The most important case is Chebyshev acceleration, which we now develop. We assume that the eigenvalues $\{\lambda_i\}_{i=1}^n$ of $M^{-1}A$ are real and satisfy

$$0 < a \le \lambda_i < b. \tag{10.4.6}$$

From (10.4.3) we get the error estimate

$$\|\tilde{x}^{(k)} - x\| = \|q_k(M^{-1}A)\|\,\|x^{(0)} - x\|,$$

If $M^{-1}A$ is Hermitian, then $\|q_k(M^{-1}A)\|_2 = \rho(q_k(M^{-1}A))$, and after k steps of the accelerated method the 2-norm of the error is reduced by at least a factor of

$$\rho(q_k(M^{-1}A)) = \max_i |q_k(\lambda_i)| \le \max_{\lambda\in[a,b]} |q_k(\lambda)|.$$

Therefore a suitable polynomial q_k is obtained by solving the minimization problem

$$\min_{q\in\Pi_k^1} \max_{\lambda\in[a,b]} |q(\lambda)|,$$

where Π_k^1 denotes the set of residual polynomials q_k of degree $\le k$ such that $q_k(0) = 1$. By a similar argument as used in the proof of the minimax property of Chebyshev polynomials, see Section 9.3.4, the solution to the above minimization problem is given by the shifted and normalized Chebyshev polynomials

$$q_k(\lambda) = T_k(z(\lambda))/T_k(z(0)), \tag{10.4.7}$$

where $T_k(z)$ is the Chebyshev polynomial of degree k and $z(\lambda)$ the linear transformation, which maps the interval $\lambda \in [a, b]$ onto $z \in [-1, 1]$. Hence

$$z(\lambda) = \frac{b + a - 2\lambda}{b - a} = \mu - \frac{2}{b - a}\lambda, \tag{10.4.8}$$

where

$$\mu = z(0) = \frac{b + a}{b - a} = \frac{\kappa + 1}{\kappa - 1} > 1, \qquad \kappa = \frac{b}{a}. \tag{10.4.9}$$

Note that if $M^{-1}A$ is symmetrizable κ is the condition number of $M^{-1}A$.

Since $|T_k(z)| \leq 1$, $z \in [-1, 1]$, and $\mu > 1$, we have

$$\rho(q_k(M^{-1}A)) \leq 1/T_k(\mu) < 1.$$

Hence k iterations will reduce the error norm by at least a factor

$$T_k(\mu) = \cosh(k\gamma) = \tfrac{1}{2}(e^{k\gamma} + e^{-k\gamma}) > \tfrac{1}{2}e^{k\gamma},$$

where $\mu = \cosh\gamma = (e^{\gamma} + e^{-\gamma})/2$ or $e^{\gamma} = \mu + \sqrt{\mu^2 - 1}$. We obtain using (10.4.9) 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 $T_k(\mu) > \frac{1}{2}e^{2k/\sqrt{\kappa}}$, and to reduce the error norm at least by a factor of $\delta < 1$ it suffices to perform k iterations, where

$$k > \frac{1}{2}\sqrt{\kappa}\log\frac{2}{\delta}, \tag{10.4.10}$$

Thus the number of iterations required for a certain accuracy for the accelerated method is proportional to $\sqrt{\kappa}$ rather than κ. This is a great improvement! Note that the matrix M can be interpreted as a **preconditioner**. To increase the rate of convergence M should be chosen so that the conditioning of the matrix $M^{-1}A$ is improved.

Since the zeros of the Chebyshev polynomials $T_k(z)$ are known it is possible to implement Chebyshev acceleration as follows (cf. Example 10.4.2). To perform N steps we compute

$$x^{(k+1)} = x^{(k)} + \omega_k M^{-1}(b - Ax^{(k)}), \quad k = 0, 1, \ldots, N - 1, \tag{10.4.11}$$

where

$$\omega_k = 2\left[(b + a) - (b - a)\cos\left(\left(k + \tfrac{1}{2}\right)/N\right)\right]^{-1}, \quad k = 0, 1, \ldots, N - 1. \tag{10.4.12}$$

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 10.2.) However, this scheme was shown by David Young to be very sensitive to rounding error effects unless N is small. The instability can be eliminated by using a certain reordering of the iteration parameters ω_k; see Computer exercise

1. However, one disadvantage remains, namely, the number N has to be fixed in advance.

The best way to compute the vectors $\tilde{x}^{(k)}$ is instead based on the three term recurrence relation for the Chebyshev polynomials. We have (see Section 9.3.4) $T_0(z) = 1$,

$$T_1(z) = zT_0, \qquad T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z), \quad k \geq 1. \tag{10.4.13}$$

By (10.4.7) $T_k(z(\lambda)) = T_k(\mu)q_k(\lambda)$, and substituting $M^{-1}A$ for z in (10.4.13), we obtain

$$T_{k+1}(\mu)q_{k+1}(M^{-1}A) = 2z(M^{-1}A)T_k(\mu)q_k(M^{-1}A) - T_{k-1}(\mu)q_{k-1}(M^{-1}A).$$

Multiplying by $(\tilde{x}^{(0)} - x)$, using (10.4.3) and (10.4.8) we obtain

$$T_{k+1}(\mu)(\tilde{x}^{(k+1)} - x) = 2\Big(\mu I - \frac{2}{b-a}M^{-1}A\Big)T_k(\mu)(\tilde{x}^{(k)} - x) - T_{k-1}(\mu)(\tilde{x}^{(k-1)} - x).$$

From (10.4.13) with $z = \mu$ it then follows that for $k \geq 1$

$$T_{k+1}(\mu)\tilde{x}^{(k+1)} = 2\mu T_k(\mu)\tilde{x}^{(k)} - \frac{4T_k(\mu)}{b-a}M^{-1}A(\tilde{x}^{(k)} - x) - T_{k-1}(\mu)\tilde{x}^{(k-1)}.$$

Further, we have

$$M^{-1}A(\tilde{x}^{(k)} - x) = M^{-1}(A\tilde{x}^{(k)} - b) = -M^{-1}r^{(k)}.$$

Substituting $-T_{k-1}(\mu) = -2\mu T_k(\mu) + T_{k+1}(\mu)$ and dividing with $T_{k+1}(\mu)$ we obtain

$$\tilde{x}^{(k+1)} = \tilde{x}^{(k-1)} + \delta_k M^{-1}r^{(k)} + \omega_k(\tilde{x}^{(k)} - \tilde{x}^{(k-1)}), \qquad k = 1, 2, \ldots,$$

where $r^{(k)} = b - A\tilde{x}^{(k)}$, and with $\alpha = 2/(b+a)$,

$$\omega_k = 2\mu\frac{T_k(\mu)}{T_{k+1}(\mu)}, \qquad \delta_k = \alpha\omega_k, \quad k \geq 1.$$

A similar calculation for $k = 0$ gives $\tilde{x}^{(1)} = \tilde{x}^{(0)} + \alpha M^{-1}r^{(0)}$. Dropping the tilde this leads to the following algorithm:

Algorithm 10.4.1
The Chebyshev Semi-Iterative Method
Assume that the eigenvalues $\{\lambda_i\}_{i=1}^n$ of $M^{-1}A$ are real and satisfy $0 < a \leq \lambda_i < b$. Then

$$x^{(k+1)} = \begin{cases} x^{(0)} + \alpha M^{-1}r^{(0)}, & k = 0, \\ x^{(k-1)} + \omega_k\big(\alpha M^{-1}r^{(k)} + x^{(k)} - x^{(k-1)}\big), & k = 1, 2, \ldots, \end{cases} \tag{10.4.14}$$

where $\mu = (b+a)/(b-a)$, $\alpha = 2/(b+a)$, and

$$\omega_0 = 2, \quad \omega_k = \left(1 - \frac{\omega_{k-1}}{4\mu^2}\right)^{-1}, \quad k = 1, 2, \ldots.$$

A disadvantage of Chebyshev convergence acceleration is that it requires knowledge of an interval $[a, b]$ enclosing the (real) spectrum of $M^{-1}A$ is needed. If this the enclosure is too crude, then the process loses efficiency.

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 [21, 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.

Review Questions

1. 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

1. Verify the recursion for ω_k for the Chebyshev semi-iteration method.

2. Show that

$$\log\big((1+s)/(1-s)\big) = 2(s + s^3/3 + s^5/5 + \ldots), \quad 0 \le s < 1,$$

and use this result to prove (10.4.10).

3. 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 $|b| \gg c$.

Computer Exercises

1. 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 (10.4.11)–(10.4.12). 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 Matlab program:

```
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

$$A = \text{tridiag}(-I, T + 2I, -I) \in \mathbf{R}^{n^2 \times n^2}, \qquad T = \text{tridiag}(-1, 2 - 1) \in \mathbf{R}^{n \times n}.$$

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!

10.5 Projection Methods

10.5.1 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. 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.,

$$\hat{x} \in \mathcal{K}, \qquad b - A\hat{x} \perp \mathcal{L}. \tag{10.5.1}$$

Many important classes of iterative methods can be interpreted as being projection methods in this general sense. The conditions (10.5.1) are often known as **Petrov–Galerkin conditions**.

We can obtain a matrix form of (10.5.1) by introducing basis vectors in the two subspaces. If we let

$$\mathcal{K} = \mathcal{R}(U), \qquad \mathcal{L} = \mathcal{R}(V), \tag{10.5.2}$$

where $U = (u_1, \ldots, u_m)$, $V = (v_1, \ldots, v_m)$, then we can write (10.5.1) as $V^T(b - AUz) = 0$, for some $z \in \mathbf{R}^m$. Hence $\hat{x} = Uz$ is obtained by solving the reduced system

$$\hat{A}z = V^T b, \qquad \hat{A} = V^T AU. \tag{10.5.3}$$

We usually have $m \ll n$, and when m is small this system can be solved by a direct method.

Example 10.5.1. Even though A is nonsingular the matrix $\hat{A}$ may be singular. Take, e.g., $m = 1$, $U = V = e_1$, 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 also nonsingular.

2. Let A be nonsingular and $\mathcal{L} = A\mathcal{K}$. Then we can take $V = AU$, and we get $\hat{A} = U^T A^T AU$. Here $A^T A$ is s.p.d. and hence $\hat{A}$ is nonsingular.

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 10.5.1. *For an s.p.d. matrix A we define a related A-inner product and A-norm by*

$$(u, v)_A = u^T Av, \qquad \|u\|_A = (u^T Au)^{1/2}, \tag{10.5.4}$$

It is easily verified that $\|u\|_A$ satisfies the conditions for a norm.

Lemma 10.5.2.

Let A by symmetric, positive definite (s.p.d.) and consider the case $\mathcal{L} = \mathcal{K}$, $(V = U)$. Then $\hat{x} = U(U^T AU)^{-1}U^T b$ minimizes the A-norm of the error over all vectors $x \in \mathcal{K}$, i.e., $\hat{x}$ solves the problem

$$\min_{x \in \mathcal{K}} \|x - x^*\|_A, \quad x^* = A^{-1}b. \tag{10.5.5}$$

Proof. By (10.5.1) $\hat{x}$ 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 $v = 0$. □

A related result is obtained for the second case.

Lemma 10.5.3.

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.,

$$\min_{x \in \mathcal{K}} \|b - Ax\|_2. \tag{10.5.6}$$

Proof. Using $x = Uz$ we have $\|b - Ax\|_2 = \|b - AUz\|_2$, which is minimized when z satisfies the normal equations $U^T A^T AUz = U^T A^T b$. This gives the desired result. □

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.[4]

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

$$r_k = b - Ax_k, \qquad \hat{z} = (V^T AU)^{-1}V^T r_k, \qquad x_{k+1} = x_k + Uz. \tag{10.5.7}$$

for computing the new approximation x_{k+1}. A generic projection algorithm is obtained by starting from some x_0 (e.g., $x_0 = 0$), and repeatedly perform (10.5.7) for a sequence of subspaces $\mathcal{L} = \mathcal{L}_k$, $\mathcal{K} = \mathcal{K}_k$, $k = 1, 2, \ldots$.

10.5.2 The One-Dimensional Case

The simplest case of a projection method is when $m = 1$. Then in step k we take $\mathcal{L}_k = \text{span}(v_k)$, and $\mathcal{K}_k = \text{span}(u_k)$. Starting from some x_0, we update x_k in the kth step by

$$r_k = b - Ax_k, \quad \alpha_k = \frac{v_k^T r_k}{v_k^T Au_k}, \quad x_{k+1} = x_k + \alpha_k u_k, \tag{10.5.8}$$

where we have to require that $v_k^T Au_k \neq 0$. By construction the new residual r_{k+1} is orthogonal to v_k. Note that r_k can be computed recursively from

$$r_k = r_{k-1} - \alpha_{k-1} Au_{k-1}. \tag{10.5.9}$$

[4] 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.

This expression is obtained by multiplying $x_k = x_{k-1} + \alpha_{k-1}u_{k-1}$ by A and using the definition $r_k = b - Ax_k$ of the residual. 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. we can take $v_k = u_k$ and the above formulas become

$$r_k = b - Ax_k, \quad \alpha_k = \frac{u_k^T r_k}{u_k^T A u_k}, \quad x_{k+1} = x_k + \alpha_k u_k, \tag{10.5.10}$$

In this case x_{k+1} minimizes the quadratic functional

$$\phi(x) = \|x - x^*\|_A^2 = (x - x^*)^T A(x - x^*) \tag{10.5.11}$$

for all vectors of the form $x_k + \alpha_k u_k$.

The vectors u_k are often called **search directions**. Expanding the function $\phi(x_k + \alpha u_k)$ with respect to α, we obtain

$$\phi(x_k + \alpha u_k) = \phi(x_k) - \alpha u_k^T(b - Ax_k) + \frac{1}{2}\alpha^2 u_k^T A u_k. \tag{10.5.12}$$

Taking $\alpha = \omega\alpha_k$ where α_k is given by (10.5.10) we obtain

$$\phi(x_k + \omega\alpha_k u_k) = \phi(x_k) - \rho(\omega)\frac{(u_k^T r_k)^2}{u_k^T A u_k}, \quad \rho(\omega) = \frac{1}{2}\omega(2 - \omega), \tag{10.5.13}$$

which is a quadratic function of ω. In a projection step ($\omega = 1$) the line $x_k + \alpha u_k$ is tangent to the ellipsoidal level surface $\phi(x) = \phi(x_{k+1})$, and $\phi(x_k + \alpha_k u_k) < \phi(x_k)$ provided that $u_k^T r_k \neq 0$. More generally, if $u_k^T r_k \neq 0$ we have from symmetry that

$$\phi(x_k + \omega\alpha_k u_k) < \phi(x_k), \qquad 0 < \omega < 2.$$

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.

Example 10.5.2. 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 10.5.4.

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 $y_{k+1} = Ky_k$, for some matrix K. Here $\|Ky_k\|_A < \|y_k\|_A$ 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 $\|Ky_k\|_A < \|y_k\|_A$, and thus $\|K\|_A < 1$. It follows that

$$\|K^n y_0\|_A \leq \|K\|_A^n \|y_0\|_A \to 0 \quad \text{when} \quad n \to \infty,$$

i.e., the iteration converges.

If we define the minor step as $x := \omega \hat{a} e_i$, where ω is a fix relaxation factor, the convergence proof also holds. (We may even let ω vary with i, although the proof assumes that ω for the same i has the same value in all major steps.) This shows that the SOR method is convergent and by Theorem 10.2.2 this is equivalent to $\rho(B_\omega) < 1$, $0 < \omega < 2$. ∎

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, $P = (p_1, \ldots, p_n)$, i.e., if $P^T AP = D$, then the exact solution will be obtained in n steps using search directions $p_1, \ldots, p_n$. Note that this condition is equivalent to the requirement that the vectors $\{p_j\}_{i=1}^n$ should be A-orthogonal, $p_i^T A p_j = 0, \qquad i \neq j$.

10.5.3 The Method of Steepest Descent

From the expansion (10.5.12) it is clear that the negative gradient of $\phi(x)$ with respect to x equals $-\nabla\phi(x) = b - Ax$. Hence the direction in which the function ϕ decreases most rapidly at the point x_k equals the residual $r_k = b - Ax_k$. The **method of steepest descent** is a one-dimensional projection method where we take $v_k = u_k = r_k$. Then

$$r_k = b - Ax_k, \quad \alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}, \quad x_{k+1} = x_k + \alpha_k r_k. \tag{10.5.14}$$

and according to (10.5.13) when $r_k \neq 0$, we have $\phi(x_{k+1}) < \phi(x_k)$.

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 (10.5.14) for α_k we obtain

$$\|e_{k+1}\|_A^2 = \|e_k\|_A^2 \left(1 - \frac{r_k^T r_k}{r_k^T A r_k} \frac{r_k^T r_k}{r_k^T A^{-1} r_k}\right). \tag{10.5.15}$$

Assume that A is s.p.d. with eigenvalues equal to $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Then from a well known inequality by Kantorovich it follows that (see, e.g., Axelsson and Barker [1984, Sec. 1.2]) that for the method of steepest descent

$$\|x - x_k\|_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x - x_0\|_A, \qquad \kappa = \lambda_1/\lambda_n. \tag{10.5.16}$$

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 Fig. 10.5.3 for a two dimensional case. Note that successive search directions are orthogonal.

Figure 10.5.1. *Convergence of the steepest descent method.*

We now consider the more general case when $u_0 = p_0 = r_0$ (i.e., the steepest descent direction) and the search direction $u_{k+1} = p_{k+1}$ is chosen as a linear combination of the negative gradient r_{k+1} and the previous search direction p_k, i.e.,

$$p_{k+1} = r_{k+1} + \beta_k p_k, \quad k = 0, 1, 2 \ldots. \tag{10.5.17}$$

Here the parameter β_k remains to be determined. (Note that $\beta_k = 0$ gives the method of steepest descent.) From (10.5.13) we know that to get $\phi(x_{k+1}) < \phi(x_k)$ we must have $p_k^T r_k \neq 0$. Replacing $(k+1)$ by k in (10.5.17) and multiplying by r_k^T, we obtain

$$r_k^T p_k = r_k^T r_k + \beta_{k-1} r_k^T p_{k-1} = r_k^T r_k, \tag{10.5.18}$$

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 (10.5.18) we also obtain the alternative expression

$$\alpha_k = (r_k^T r_k)/(p_k^T A p_k). \tag{10.5.19}$$

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 $p_i = e_i$, $i = 1, 2, \ldots, n$ 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?

10.6 Krylov Subspace Methods

The Lanczos algorithm and the conjugate gradient algorithm of Hestenes and Stiefel are the most important examples of Krylov subspace methods. They were developed already in the early 1950s, but did not come into wide use until twenty years later. Now these methods, combined with preconditioning techniques, have been developed into sophisticated solvers for large scale linear systems. Today these are the standard method for solving linear systems involving large, sparse, symmetric (or Hermitian) systems.

10.6.1 The Conjugate Gradient Method

We consider now projection methods where the subspaces $\mathcal{K}$ and $\mathcal{L}$ are chosen to be the sequence of **Krylov subspaces** $\mathcal{K}_{k+1}(r_0, A)$, $k = 0, 1, 2, \ldots$, where $r_0 = b - Ax_0$, and

$$\mathcal{K}_m(r_0, A) = \text{span}\,\{r_0, Ar_0, \ldots, A^{m-1}r_0\}. \tag{10.6.1}$$

This choice leads to one of the most important iterative methods for solving large symmetric positive definite systems, the **conjugate gradient method**.

If $p_0 = r_0$, and the recurrence (10.5.17) is used to generate p_{k+1} then a simple induction argument shows that the vectors p_k and r_k both will lie in the $\mathcal{K}_{k+1}(r_0, A)$. 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.,

$$p_{k+1}^T A p_k = 0. \tag{10.6.2}$$

(A motivation to this choice is given in the second remark to Theorem 10.5.4.) Multiplying (10.5.17) by $p_k^T A$ and using (10.6.2) it follows that

$$\beta_k = -(p_k^T A r_{k+1})/(p_k^T A p_k). \tag{10.6.3}$$

We now prove the important result that this choice will in fact make p_{k+1} A-conjugate to *all* previous search directions!

Lemma 10.6.1.

In the conjugate gradient algorithm the residual vector r_k is orthogonal to all previous search directions and residual vectors

$$r_k^T p_j = 0, \quad j = 0, \ldots, k-1, \tag{10.6.4}$$

and the search directions are mutually A-conjugate

$$p_k^T A p_j = 0, \quad j = 0, \ldots, k-1. \tag{10.6.5}$$

Proof. We first prove the relations (10.6.4) and (10.6.5) jointly by induction. Clearly r_k is orthogonal to the previous search direction p_{k-1}, and (10.6.2) shows that also (10.6.5) 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 (10.6.4) holds for $k := k+1$. Using equation (10.5.17), the induction hypothesis and equation (10.5.9) and then (10.5.17) 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 (10.6.4). For $j = 0$ we use $r_0 = p_0$ in forming the last line of the equation. For $j = k$ we use (10.6.2), which yields (10.6.5). ∎

Since the vectors $p_0, \ldots, p_{k-1}$ span the Krylov subspace $\mathcal{K}_k(r_0, A)$ the equation (10.6.4) shows that $r_k \perp \mathcal{K}_k(r_0, A)$. This relation shows that the conjugate gradient implements the projection method obtained by taking $\mathcal{K} = \mathcal{L} = \mathcal{K}_k(r_0, A)$. Hence from Lemma 10.5.2 we have the following *global minimization property.*

Theorem 10.6.2.

The vector x_k in the conjugate gradient method solves the minimization problem

$$\min_x \phi(x) = \frac{1}{2}\|x - x^*\|_A^2, \quad x - x_0 \in \mathcal{K}_k(r_0, A) \tag{10.6.6}$$

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 [9]).

Since the vectors $r_0, \ldots, r_{k-1}$ span the Krylov subspace $\mathcal{K}_k(r_0, A)$ the following orthogonality relations also hold:

$$r_k^T r_j = 0, \quad j = 0, \ldots, k-1. \tag{10.6.7}$$

Equation (10.6.7) ensures that in exact arithmetic the conjugate gradient method will terminate after at most n steps. For suppose the contrary is true. Then $r_k \neq 0$, $k = 0, 1, \ldots, n$ and by (10.6.7) 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 $r_0, r_1, \ldots, r_k$ are the same vectors as those obtained from the sequence $r_0, Ar_0, \ldots, A^k r_0$ by Gram-Schmidt orthogonalization. This gives a connection to the Lanczos process described in Section 10.8.4, which is further discussed below in Section 10.6.3. The vectors $p_0, p_1, p_2, \ldots$ may be constructed similarly from the conjugacy relation (10.6.5).

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 (10.6.7) to get $r_{k+1}^T r_{k+1} = -\alpha_k r_{k+1}^T A p_k$. Equations (10.5.19) and (10.6.3) then yield

$$\beta_k = r_{k+1}^T r_{k+1} / r_k^T r_k.$$

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:

Algorithm 10.6.1
The Conjugate Gradient Method

$r_0 = b - Ax_0; \quad p_0 = r_0;$

for $k = 0, 1, 2, \ldots$ **while** $\|r_k\|_2 > \epsilon$ **do**

$\alpha_k = (r_k, r_k)/(p_k, Ap_k);$

$x_{k+1} = x_k + \alpha_k p_k;$

$r_{k+1} = r_k - \alpha_k A p_k;$

$\beta_k = (r_{k+1}, r_{k+1})/(r_k, r_k);$

$p_{k+1} = r_{k+1} + \beta_k p_k;$

end

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 when forming Ap, two vector inner products and three scalar by vector products.

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.

10.6.2 Convergence of the Conjugate Gradient Method

In a Krylov subspace method the approximations are of the form $x_k - x_0 \in \mathcal{K}_k(r_0, A)$, $k = 1, 2, \ldots$. With $r_k = b - Ax_k$ it follows that $r_k - r_0 \in A\mathcal{K}_k(r_0, A)$. Hence the residual vectors can be written

$$r_k = q_k(A)r_0,$$

where $q_k \in \tilde{\Pi}_k^1$, the set of polynomials q_k of degree k with $q_k(0) = 1$. Since

$$\phi(x) = \tfrac{1}{2}\|x - x^*\|_A^2 = \tfrac{1}{2}r^T A^{-1} r = \|r\|_{A^{-1}}^2,$$

the optimality property in Theorem 10.6.2 can alternatively be stated as

$$\|r_k\|_{A^{-1}}^2 = \min_{q_k \in \tilde{\Pi}_k^1} \|q_k(A)r_0\|_{A^{-1}}^2. \tag{10.6.8}$$

Denote by $\{\lambda_i, v_i\}$, $i = 1, \ldots, n$, the eigenvalues and eigenvectors of A. Since A is symmetric we can assume that the eigenvectors are orthonormal. Expanding the right hand side as

$$r_0 = \sum_{i=1}^{n} \gamma_i v_i, \tag{10.6.9}$$

we have for *any* $q_k \in \tilde{\Pi}_k^1$

$$\|r_k\|_{A^{-1}}^2 \le \|q_k(A)r_0\|_{A^{-1}}^2 = r_0^T q_k(A)^T A^{-1} q_k(A) r_0 = \sum_{i=1}^{n} \gamma_i^2 \lambda_i^{-1} q_k(\lambda_i)^2.$$

In particular, taking

$$q_n(\lambda) = \Big(1 - \frac{\lambda}{\lambda_1}\Big)\Big(1 - \frac{\lambda}{\lambda_2}\Big) \cdots \Big(1 - \frac{\lambda}{\lambda_n}\Big), \tag{10.6.10}$$

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 (10.6.10) 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 r_0. Hence, CG is particularly effective when A has low rank! More generally, if the grade of r_0 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 (10.6.9) $\gamma_i \neq 0$ 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 (10.6.9) 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

$$\max_{\lambda \in S} |\tilde{q}_k(\lambda)| \leq M_k.$$

Then it follows that

$$\|r_k\|_{A^{-1}}^2 \leq M_k^2 \sum_{i=1}^{n} \gamma_i^2 \lambda_i^{-1} = M_k^2 \|r_0\|_{A^{-1}}^2$$

or

$$\|x - x_k\|_A \leq M_k \|x - x_0\|_A. \tag{10.6.11}$$

We now select a set S on the basis of some assumption regarding the eigenvalue distribution of A and seek a polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ such that $M_k = \max_{\lambda \in S} |\tilde{q}_k(\lambda)|$ is small.

A simple choice is to take $S = [\lambda_1, \lambda_n]$ and seek the polynomial $\tilde{q}_k \in \tilde{\Pi}_k^1$ which minimizes $\max_{\lambda_1 \leq \lambda \leq \lambda_n} |q_k(\lambda)|$. The solution to this problem is known to be a shifted and scaled Chebyshev polynomial of degree k, see the analysis in Section 10.4.2. It follows that

$$\|x - x_k\|_A < 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|x - x_0\|_A. \tag{10.6.12}$$

where $\kappa = \lambda_n(A)/\lambda_1(A)$. This estimate is the same as (10.4.10) for Chebyshev semi-iteration.

We note that the convergence of the conjugate residual method can be analyzed using a similar technique.

Example 10.6.1. 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. One often observes, in practice, a *superlinear convergence* for the conjugate gradient method. This can be theoretically explained for the case 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 10.4.2 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. However, in the presence of rounding errors, the orthogonality relations in Theorem 10.5.4 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. Its main use is instead as an *iterative method for solving large, sparse, well-conditioned linear systems*, using *far fewer than n iterations.*

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 (10.6.12) 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 (10.6.12) 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.

Table 10.6.1. *Maximum error for Example 10.6.2 using Chebyshev iteration with optimal parameters and the conjugate gradient algorithm.*

Iteration	Chebyshev	Conj. 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	$1.2 \cdot 10^{-10}$	$5.7 \cdot 10^{-12}$

Example 10.6.2. Consider the elliptic equation

$$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+p(x,y)u=f(x,y), \qquad p=\frac{6(x^2+y^2)}{1+\frac{1}{2}(x^4+y^4)},$$

$0<x,y<1$, and let the boundary conditions be determined so that

$$u(x,y)=2\Big((x-1/2)^2+(y-1/2)^2\Big).$$

The Laplacian operator is approximated with 32 mesh points in each direction. In Table 10.6.2 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 Algorithm 10.6.1 yields a smaller error without the need to estimate the parameters.

10.6.3 The Lanczos Formulation

In Section 9.9.4 we described the Lanczos process for a real symmetric matrix A. If this process can be carried out for k steps, starting with a vector v_1, it generates 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 $V_k=(v_1,\dots,v_k)$ with orthogonal columns spanning the Krylov subspace $\mathcal{K}_k(v_1,A)$ such that

$$AV_k=V_kT_k+\beta_{k+1}v_{k+1}e_k^T. \tag{10.6.13}$$

In the context of solving the linear system $Ax=b$, using Krylov subspaces the appropriate choice of starting vector is

$$\beta_1 v_1=r_0=b-Ax_0, \qquad \beta_1=\|r_0\|_2.$$

We write the kth approximation as $x_k=x_0+V_ky_k\in\mathcal{K}_k(r_0,A)$. Here y_k is determined by the condition that $r_k=b-Ax_k$ is orthogonal to $\mathcal{K}_k(r_0,A)$, i.e., $V_k^Tr_k=0$. Using (10.6.13) we have

$$r_k=r_0-AV_ky_k=\beta_1v_1-V_kT_ky_k-\beta_{k+1}(e_k^Ty_k)v_{k+1}. \tag{10.6.14}$$

Since $V_k^Tv_{k+1}=0$ and $V_k^Tv_1=e$, multiplying by V_k^T gives

$$V_k^Tr_k=0=\beta_1e_1-T_ky_k.$$

Hence y_k is obtained by solving the tridiagonal system

$$T_k y_k = \beta_1 e_1, \tag{10.6.15}$$

and then $x_k = x_0 + V_k y_k$. Mathematically this gives the same sequence of approximations as generated by the CG method. Moreover, 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 (10.6.14)

$$0 = r_k = \beta_1 v_1 - V_k T_k y_k = r_0 - AV_k y_k = r_0 - A(x_k - x_0).$$

It follows that $Ax_k = b$, i.e. x_k is an exact solution.

The recursion in the conjugate gradient method is obtained from (10.6.15) by computing the Cholesky factorization of T_k. This is always possible. Suppose the Lanczos process stops for $k = l \leq n$. Then, since A is a positive definite matrix then $T_l = V_l^T A V_l$ 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 $v_k, \ldots, v_1$. 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!

10.6.4 Symmetric Indefinite Systems

For symmetric positive definite matrices A the conjugate gradient method computes iterates x_k that satisfy the minimization property

$$\min_{x \in S_k} \|\hat{x} - x\|_A, \qquad S_k = x_0 + \mathcal{K}_k(r_0, A).$$

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 [15, 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 (10.6.15). However, when A is indefinite, although the Lanczos process is still well defined, the Cholesky factorization of T_k may not exist. Moreover, it may happen that T_k is singular at certain steps, and then y_k is not defined.

If the Lanczos process stops for some $k \leq n$ then $AV_k = V_k T_k$. 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 (10.6.15) 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

$$\bar{W}_k = (w_1, \ldots, w_{k-1}, \bar{w}_k),$$
$$\bar{z}_k = (\zeta_1, \ldots, \zeta_{k-1}, \bar{\zeta}_k) = Q_k y_k.$$

Here quantities without bars will be unchanged when k increases, and $\bar{W}_k$ can be updated with $\bar{T}_k$. The system (10.6.15) 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.

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

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 (10.6.13) using an LQ factorization of T_k. Derive an alternative algorithm, which solves this system with Gaussian elimination with partial pivoting.

10.7 Nonsymmetric Problems

An ideal conjugate gradient-like method for nonsymmetric systems would be characterized by one of the properties (10.6.6) or (10.6.14). 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.

10.7.1 The Normal Equations

When $A \in \mathbf{R}^{m \times n}$, rank $(A) = n$, the normal equations of the first kind

$$A^T Ax = A^T b \tag{10.7.1}$$

give the conditions for the solution of the linear least squares problems

$$\min_x \|Ax - b\|_2, \tag{10.7.2}$$

Similarly, When $\text{rank}\,(A) = m$, the normal equations of the second kind

$$AA^T y = b, \qquad x = A^T y, \tag{10.7.3}$$

give the conditions for the solution of the minimum norm problem

$$\min \|x\|_2, \qquad Ax = b. \tag{10.7.4}$$

In particular, if A is nonsingular, then both systems are symmetric 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.

Because of the relation $\kappa(A^T A) = \kappa(AA^T) = \kappa^2(A)$, the condition number is squared compared to the original system $Ax = b$. From the estimate (10.6.12) we note that this can lead to a substantial decrease in the rate of convergence.

10.7.2 Classical iterative methods.

The non-stationary Richardson iteration applied to the normal equations $A^T Ax = A^T b$ can be written in the form

$$x^{(k+1)} = x^{(k)} + \omega_k A^T (b - Ax^{(k)}), \quad k = 1, 2, \ldots,$$

This method is often referred to as **Landweber's method**. 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

$$\lambda_k(G) = 1 - \alpha\sigma_k^2, \quad k = 1, \ldots, n,$$

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

$$x^{(0)} \in \mathcal{R}(A^T), \qquad 0 < \alpha < 2/\sigma_1^2(A).$$

Assume that all columns in A are nonzero, and let

$$A = (a_1, \ldots, a_n) \in \mathbf{R}^{m\times n}, \quad d_j = a_j^T a_j = \|a_j\|_2^2 > 0. \tag{10.7.5}$$

In **Jacobi's** method a sequence of approximations

$$x^{(k)} = \left(x_1^{(k)}, \ldots, x_n^{(k)} \right)^T, \qquad k = 1, 2, \ldots,$$

is computed from

$$x_j^{(k+1)} = x_j^{(k)} + a_j^T (b - Ax^{(k)})/d_j, \quad j = 1, 2, \ldots, n. \tag{10.7.6}$$

Jacobi's method can be written (10.7.6) in matrix form as

$$x^{(k+1)} = x^{(k)} + D_A^{-1} A^T (b - Ax^{(k)}), \tag{10.7.7}$$

where $D_A = \operatorname{diag}(d_1, \ldots, d_n) = \operatorname{diag}(A^T A)$. Jacobi's method is symmetrizable since

$$D_A^{1/2}(I - D_A^{-1} A^T A) D_A^{-1/2} = I - D_A^{-1/2} A^T A D_A^{-1/2}.$$

Jacobi's method can also be used to solve the normal equations of second type (10.7.3). This method can be written in the form

$$x^{(k+1)} = x^{(k)} + A^T D_A^{-1} (b - Ax^{(k)}), \quad D_A = \operatorname{diag}(AA^T). \tag{10.7.8}$$

The Gauss–Seidel method is a special case of the following class of **residual reducing** methods. Let $p_j \notin \mathcal{N}(A)$, $j = 1, 2, \ldots$, be a sequence of nonzero n-vectors and compute a sequence of approximations of the form

$$x^{(j+1)} = x^{(j)} + \alpha_j p_j, \qquad \alpha_j = p_j^T A^T (b - Ax^{(j)})/\|Ap_j\|_2^2. \tag{10.7.9}$$

It is easily verified that $r^{(j+1)} \perp Ap_j = 0$, where $r_j = b - Ax^{(j)}$, and hence

$$\|r^{(j+1)}\|_2^2 = \|r^{(j)}\|_2^2 - |\alpha_j|^2 \|Ap_j\|_2^2 \le \|r^{(j)}\|_2^2,$$

which shows that this class of methods (10.7.9) is residual reducing.

If A has linearly independent columns we obtain the Gauss–Seidel method for the normal equations by taking p_j in (10.7.9) equal to the unit vectors e_j in cyclic order. Then if $A = (a_1, a_2, \ldots, a_n)$, 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

$$z^{(j+1)} = z^{(j)} + e_j a_j^T r^{(j)}/d_j, \qquad r^{(j)} = b - Az^{(j)}, \tag{10.7.10}$$

$j = 1, 2, \ldots, n$. 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 recursions

$$\begin{aligned} &z^{(j+1)} = z^{(j)} + \delta_j e_j, \quad r^{(j+1)} = r^{(j)} - \delta_j a_j, \\ &\delta_j = a_j^T r^{(j)}/d_j, \quad j = 1, \ldots, n. \end{aligned} \tag{10.7.11}$$

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 $A^T A$ explicitly. In contrast to the

Jacobi method the Gauss–Seidel method is not symmetrizable and the ordering of the columns of A will influence the convergence.

The Jacobi method has the advantage over Gauss–Seidel's method that it is more easily adapted to parallel computation, since (10.7.8) 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 $A^T r$ can conveniently be computed also if A can only be accessed by rows. In this case, if $a_1^T, \ldots, a_m^T$ are the rows of A, then we have

$$(Ax)_i = a_i^T x, \quad i = 1, \ldots, n, \quad A^T r = \sum_{i=1}^{m} a_i r_i.$$

That is, for Ax we use an inner product formulation, and for $A^T r$, an outer product formulation.

The **successive overrelaxation (SOR) method** for the normal equations $A^T Ax = A^T b$ is obtained by introducing an **relaxation parameter** ω in the Gauss–Seidel method (10.7.12),

$$\begin{aligned} z^{(j+1)} &= z^{(j)} + \delta_j e_j, \quad r^{(j+1)} = r^{(j)} - \delta_j a_j, \\ \delta_j &= \omega a_j^T r^{(j)}/d_j, \quad j = 1, \ldots, n. \end{aligned} \tag{10.7.12}$$

The SOR method always converges when $A^T A$ is positive definite and ω satisfies $0 < \omega < 2$. The SOR shares with the Gauss–Seidel the advantage of simplicity and small storage requirements.

The Gauss–Seidel method for solving the normal equations of second kind can also be implemented without forming AA^T. We define a class of **error reducing** methods as follows: Let $p_i \notin \mathcal{N}(A)$, $i = 1, 2, \ldots$, be a sequence of nonzero m-vectors and compute approximations of the form

$$x^{(j+1)} = x^{(j)} + \alpha_j A^T p_j, \qquad \alpha_j = p_j^T (b - Ax^{(j)})/\|A^T p_j\|_2^2. \tag{10.7.13}$$

If the system $Ax = b$ is consistent there is a unique solution x of minimum norm. If we denote the error by $d^{(j)} = x - x^{(j)}$, then by construction $d^{(j+1)} \perp A^T p_i$. Thus

$$\|d^{(j+1)}\|_2^2 = \|d^{(j)}\|_2^2 - |\alpha_j|^2 \|A^T p_j\|_2^2 \leq \|d^{(j)}\|_2^2,$$

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 $A^T p_j = a_{j\cdot}$, where $a_{j\cdot}^T$ is the jth row of A^T and the iterative method (10.7.13) takes the form

$$x^{(j+1)} = x^{(j)} + a_{j\cdot}(b_j - a_{j\cdot}^T x^{(j)})/d_j, \quad j = 1, \ldots, n. \tag{10.7.14}$$

the approximation $y^{(j)}$ is updated by

$$\Delta y^{(j)} = e_j (b_j - a_{j\cdot}^T A^T y^{(i)})/d_j,$$

and with $x^{(j)} = A^T y^{(j)}$ and $x^{(j+1)} = x^{(j)} + A^T \Delta y^{(j)}$ we recover (10.7.14). This shows that if we take $x^{(0)} = Ay^{(0)}$, then for an arbitrary $y^{(0)}$ (10.7.14) is equivalent to the Gauss–Seidel method for (10.7.3).

The SOR method applied to the normal equations of the second kind can be obtained by introducing an acceleration parameter ω, i.e.

$$x^{(j+1)} = x^{(j)} + \omega a_{j.}(c_j - a_{j.}^T x^{(j)})/d_j, \quad j = 1, \ldots, n. \tag{10.7.15}$$

10.7.3 The conjugate gradient method

The implementation of CG applied to the normal equations of the first kind becomes as follows:

Algorithm 10.7.1
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**
  $q_k = Ap_k;$
  $\alpha_k = \|s_k\|_2^2 / \|q_k\|_2^2;$
  $x_{k+1} = x_k + \alpha_k p_k;$
  $r_{k+1} = r_k - \alpha_k q_k;$
  $s_{k+1} = A^T r_{k+1};$
  $\beta_k = \|s_{k+1}\|_2^2 / \|s_k\|_2^2;$
  $p_{k+1} = s_{k+1} + \beta_k p_k;$
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 10.7.2
CGNE

$r_0 = b - Ax_0; \quad p_0 = A^T r_0;$
for $k = 0, 1, \ldots$ **while** $\|r_k\|_2 > \epsilon$ **do**
  $\alpha_k = \|r_k\|_2^2 / \|p_k\|_2^2;$

$$
\begin{aligned}
x_{k+1} &= x_k + \alpha_k p_k;\\
r_{k+1} &= r_k - \alpha_k A p_k;\\
\beta_k &= \|r_{k+1}\|_2^2/\|r_k\|_2^2;\\
p_{k+1} &= A^T r_{k+1} + \beta_k p_k;
\end{aligned}
$$

end

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!

10.7.4 Least Squares and LSQR.

As shown by Paige and Saunders the Golub–Kahan bidiagonalization process developed in Section 10.9.4 can be used for developing methods related to CGLS and CGNE for solving the linear least squares problem (10.7.1) and the minimum norm problem (10.7.1), respectively.

To compute a sequence of approximate solutions to the least squares problem we start the recursion (10.9.19)–(10.19.20) by

$$\beta_1 u_1 = b - Ax_0, \qquad \alpha_1 v_1 = A^T u_1, \tag{10.7.16}$$

and for $j = 1, 2, \ldots$ compute

$$
\begin{aligned}
\beta_{j+1} u_{j+1} &= Av_j - \alpha_j u_j,\\
\alpha_{j+1} v_{j+1} &= A^T u_{j+1} - \beta_{j+1} v_j,
\end{aligned} \tag{10.7.17}
$$

where $\alpha_{j+1} \geq 0$ and $\beta_{j+1} \geq 0$ are determined so that $\|u_{j+1}\|_2 = \|v_{j+1}\|_2 = 1$.

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

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} \in \mathbf{R}^{(k+1)\times k}. \tag{10.7.18}$$

The recurrence relations (10.7.16)–(10.7.17) can be written in matrix form as

$$\beta_1 U_{k+1} e_1 = b, \tag{10.7.19}$$

where e_1 denotes the first unit vector, and

$$AV_k = U_{k+1} B_k, \qquad A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \tag{10.7.20}$$

We now seek an approximate solution $x_k \in \mathcal{K}_k = \mathcal{K}_k(A^T b, A^T A)$. Since $\mathcal{K}_k = \text{span}(V_k)$, we can write

$$x_k = x_0 + V_k y_k. \tag{10.7.21}$$

Multiplying the first equation in (10.7.20) by y_k we obtain $Ax_k = AV_k y_k = U_{k+1} B_k y_k$, and then from (10.7.19)

$$b - Ax_k = U_{k+1} t_{k+1}, \qquad t_{k+1} = \beta_1 e_1 - B_k y_k. \tag{10.7.22}$$

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 $x_k \in \text{span}(V_k)$ by taking y_k to be the solution to the least squares problem

$$\min_{y_k} \|B_k y_k - \beta_1 e_1\|_2. \tag{10.7.23}$$

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 $\mathcal{K}_k = \mathcal{K}_k(A^T b, A^T A)$. Thus *mathematically* LSQR generates the same sequence of approximations as Algorithm 10.7.3 CGLS.

To solve (10.7.23) 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

$$\begin{aligned} x_k &= x_0 + V_k y_k = x_0 + \beta_1 V_k R_k^{-1} d_k = x_0 + \beta_1 W_k d_k \\ &= x_0 + \beta_1 (W_{k-1},\ w_k) \begin{pmatrix} d_{k-1} \\ \tau_k \end{pmatrix} = x_{k-1} + \beta_1 \tau_k w_k. \end{aligned} \tag{10.7.24}$$

Consider now the minimum norm problem for a consistent system $Ax = b$. Let L_k be the upper bidiagonal matrix formed by the first k rows of B_k

$$L_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{pmatrix} \in \mathbf{R}^{k \times k}. \tag{10.7.25}$$

The relations (10.7.20) can now be rewritten as

$$AV_k = U_k L_k + \beta_{k+1} u_{k+1} e_k^T, \qquad A^T U_k = V_k L_k^T. \tag{10.7.26}$$

The iterates x_k in Craig's method can be computed as

$$L_k y_k = \beta_1 e_1, \qquad x_k = V_k z_k. \tag{10.7.27}$$

Using (10.7.26) and (10.7.16) 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

$$\eta_k = -\frac{\beta_k}{\alpha_k} \eta_{k-1}, \qquad x_k = x_{k-1} + \eta_k v_k.$$

10.7.5 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 $\kappa(A^T A) = \kappa^2(A)$.

We now consider a method for general nonsymmetric systems based instead on the Arnoldi process (see Section 9.9.6). Given a starting vector v_1 this process computes an orthogonal basis for the Krylov subspace $\mathcal{K}_k(v_1, A)$. In exact arithmetic the result after k steps is a a matrix $V_k = (v_1, \ldots, v_k)$, with orthogonal columns, and a related Hessenberg matrix $H_k = (h_{ij}) \in \mathbf{R}^{k \times k}$.

In the following implementation of the Arnoldi process we perform the orthogonalization by the modified Gram-Schmidt method.

Algorithm 10.7.3 The Arnoldi process.

```
for j = 1 : k do
    z_j = A v_j;
    for i = 1 : j do
        h_ij = z_j^T v_i;
        z_j = z_j − h_ij v_i;
    end
    h_{j+1,j} = ||z_j||_2;
    if abs(h_{j+1,j}) < ε, break end
    v_{j+1} = z_j / h_{j+1,j};
end
```

By construction it is easily seen that if we take

$$v_1 = r_0/\beta_1, \quad r_0 = b - Ax_0, \quad \beta_1 = \|r_0\|_2,$$

then in in Arnoldi's process we have

$$v_k \in \text{span}(r_0, Ar_0, \ldots, A^{k-1}r_0) \equiv \mathcal{K}_k(r_0, A),$$

and the vectors $v_1, \ldots, v_k$ form an orthogonal basis in the Krylov subspace $\mathcal{K}_k(r_0, A)$. Further we have

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T.$$

The Arnoldi process will break down at step j if and only if the vector z_j vanishes. When this happens we have $h_{k+1,k} = 0$ and hence $AV_k = V_k H_k$, and V_k is a right invariant subspace of A.

In the generalized minimum residual (GMRES) method we seek at step $j = k$ an approximate solution of the form

$$x_k = x_0 + V_k y_k \in x_0 + \mathcal{K}_k(r_0, A). \tag{10.7.28}$$

If we write (10.7.29) as $AV_k = V_{k+1}\bar{H}_k$, where

$$\bar{H}_k = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & & h_{2k} \\ & \ddots & \ddots & \vdots \\ & & h_{k,k-1} & h_{kk} \\ & & & h_{k+1,k} \end{pmatrix} \in \mathbf{R}^{(k+1)\times k}.$$

then the residual $r_k = b - Ax_k$ can be expressed as

$$r_k = r_0 - AV_k y_k = \beta_1 v_1 - V_{k+1}\bar{H}_k y_k = V_{k+1}(\beta_1 e_1 - \bar{H}_k y_k). \tag{10.7.29}$$

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

$$\min_{y_k} \|\beta_1 e_1 - \bar{H}_k y_k\|_2. \tag{10.7.30}$$

The QR factorization of the Hessenberg matrix $\bar{H}_k$ can be obtained by a sequence of k plane rotations, and we write

$$Q_k^T(\bar{H}_k \ e_1) = \begin{pmatrix} R_k & d_k \\ 0 & \rho_k \end{pmatrix}, \quad Q_k^T = G_{k,k+1}G_{k-1,k}\cdots G_{12}, \tag{10.7.31}$$

where $G_{j+1,j}$ is chosen to zero the subdiagonal element $h_{j+1,j}$. Then the solution to (10.7.30) is obtained from the triangular system

$$R_k y_k = \beta_1 d_k,$$

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 *update the QR factorization* (10.7.31) at each step of the Arnoldi process. To derive the updating formulas for step $j = k$ we write

$$Q_k^T \bar{H}_k = G_{k,k+1} \begin{pmatrix} Q_{k-1}^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{pmatrix} = \begin{pmatrix} R_k \\ 0 \end{pmatrix},$$

where, since $G_{k,k+1}$ will not affect R_{k-1},

$$R_k = \begin{pmatrix} R_{k-1} & r_{k-1} \\ 0 & \gamma_k \end{pmatrix}, \qquad h_k = \begin{pmatrix} h_{1k} \\ \vdots \\ h_{kk} \end{pmatrix}. \tag{10.7.32}$$

The rotation $G_{k,k+1}$ and the kth column in R_k is determined from

$$Q_k^T h_k = G_{k-1,k}\cdots G_{12}h_k = \begin{pmatrix} r_{k-1} \\ \delta_k \end{pmatrix}, \qquad G_{k,k+1} \begin{pmatrix} r_{k-1} \\ \delta_k \\ h_{k+1,k} \end{pmatrix} = \begin{pmatrix} c_{k-1} \\ \gamma_k \\ 0 \end{pmatrix}. \tag{10.7.33}$$

For the residual $r_k = b - Ax_k$ it holds

$$\|r_k\|_2 = \beta_1 \rho_k. \tag{10.7.34}$$

(Note that Q_k^T, which is a full upper Hessenberg matrix, should not be explicitly formed.)

Proceeding similarly with the right hand side, we have

$$Q_k^T e_1 = G_{k,k+1} \begin{pmatrix} Q_{k-1}^T e_1 \\ 0 \end{pmatrix} = G_{k,k+1} \begin{pmatrix} d_{k-1} \\ \rho_{k-1} \\ 0 \end{pmatrix} = \begin{pmatrix} d_{k-1} \\ \tau_k \\ \rho_k \end{pmatrix} \equiv \begin{pmatrix} d_k \\ \rho_k \end{pmatrix}. \tag{10.7.35}$$

(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 affected.

The new approximate solution could be obtained from (10.7.28), but note that the whole vector y_k differs from y_{k-1}. Since $\|r_k\|_2 = |\rho_k|$ is available without forming x_k, this expensive operation can be delayed until ρ_k is small enough.

An updating formula for x_k can be derived using the same trick as in (10.7.24). There we have set $W_k R_k = V_k$, which can be written

$$(W_{k-1},\ w_k)\begin{pmatrix} R_{k-1} & c_{k-1} \\ 0 & \gamma_k \end{pmatrix} = (V_{k-1},\ v_k).$$

Equation the first block columns we get $W_{k-1}R_{k-1} = V_{k-1}$, which shows that the first $k-1$ columns of W_k equal W_{k-1}. Equating the last column we get

$$w_k = (v_k - W_{k-1}r_{k-1})/\gamma_k, \tag{10.7.36}$$

which determines 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 (10.7.33). This also determines τ_k and $\rho_k = \|r_k\|_2$.
3. Compute the new column w_k using (10.7.35) and x_k from (10.7.24).

In exact arithmetic GMRES cannot break down and will give the exact solution in at most n steps. If at some step we have $\tilde{v}_{k+1} = 0$ then $h_{k+1,k} = 0$, and the vector v_{k+1} cannot be constructed. However, then it is seen that $AV_k = V_k H_k$, where H_k is the matrix consisting of the first k rows of $\bar{H}_k$. This means that $x_k = x_0 + V_k y_k$, with $y_k = H_k^{-1}\beta_1 e_1$ is the exact solution. It follows that the GMRES terminates in at most n steps.

Suppose that the matrix A is diagonalizable, $A = X\Lambda X^{-1}$, where $\Lambda = \mathrm{diag}(\lambda_i)$. Then, using the property that the GMRES approximations minimize the Euclidian norm of the residual $r_k = b - Ax_k$ in the Krylov subspace $\mathcal{K}_k(r_0, A)$, it can be shown that

$$\frac{||r_k||_2}{||r_0||_2} \le \kappa_2(X) \min_{q_k} \max_{i=1,2,\ldots,n} |q_k(\lambda_i)|, \tag{10.7.37}$$

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 10.6.2. This results looks similar to that obtained for the symmetric case. However, because of the factor $\kappa_2(X)$ in (10.7.37) the rate of convergence can no longer be deduced from the spectrum $\{\lambda_i\}$ of the matrix A alone. In the special case that A is normal we have $\kappa_2(X) = 1$, and the convergence can be determined via the complex approximation problem

$$\min_{q_k} \max_{i=1,2,\ldots,n} |q_k(\lambda_i)|, \qquad q_k(0) = 1.$$

Because complex approximation problems are harder than real ones no simple results are available even for this special case.

If GMRES is applied to a real symmetric system, it can be implemented with three-term recurrences, which avoids the necessity to store all basis vectors v_j. This leads to the method MINRES by Paige and Saunders mentioned in Section 10.7.5. (However, a version of GMRES, which uses the three-term recurrence but stores all the Lanczos vectors, has been shown to be less affected by rounding errors than MINRES.)

In practice the number of steps taken by GMRES must be limited because of limited memory and computational complexity. This can be achieved by **restarting** GMRES after each m iterations, and we denote the corresponding algorithm GMRES(m). Note that storage requirements grow linearly with m, and the computational requirement quadratically with m. In practice typically $m \in [5, 20]$. GMRES(m) can not break down in exact arithmetic before the true solution has been produced. Unfortunately it may never converge for $m < n$.

10.7.6 Lanczos Bi-orthogonalization

The GMRES method was based on an algorithm for reducing a nonsymmetric matrix to Hessenberg form by an orthogonal similarity. Another generalization proposed by Lanczos in [11] is based on the reduction of $A \in \mathbf{C}^{n\times n}$ to tridiagonal form by a general similarity transformation. As discussed in Wilkinson [19, pp. 388–405], this reduction can be performed in two steps. First an orthogonal similarity is used to reduce A to lower Hessenberg form. Second the appropriate elements in the lower triangular half are zeroed column by column using a sequence of similarity transformations by elimination matrices of the form in (6.3.15).

$$H := (I - l_j e_j^T)H(I + l_j e_j^T), \qquad j = 1, \ldots, n-1.$$

In this step row pivoting can not be used, since this would destroy the lower Hessenberg structure. As a consequence, the reduction will fail if a zero pivot element is encountered. 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

$$W^T A V = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \gamma_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_n \\ & & & \gamma_n & \alpha_n \end{pmatrix},$$

where $V = (v_1, \ldots, v_n)$ and $W = (w_1, \ldots, w)$, and $W^T V = I$. The two vector sequences then are **bi-orthogonal**, i.e.,

$$w_i^T v_j = \begin{cases} 1, & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{10.7.38}$$

Comparing columns in $AV = VT$ and $A^TW = WT^T$ we find (with $v_0 = w_0 = 0$) the recurrence relations

$$\gamma_{k+1}v_{k+1} = \tilde{v}_{k+1} = (A - \alpha_k I)v_k - \beta_k v_{k-1}, \tag{10.7.39}$$
$$\beta_{k+1}w_{k+1} = \tilde{w}_{k+1} = (A^T - \alpha_k I)w_k - \gamma_k w_{k-1}, \tag{10.7.40}$$

Multiplying equation (10.7.39) 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 (10.7.40) 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. We now have the following algorithm for generating the two sequences of vectors $v_1, v_2, \ldots$ and $w_1, w_2, \ldots$:

Algorithm 10.7.4
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 $T_k = \text{trid}\,(\gamma_j, \alpha_j, \beta_{j+1})$ 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 $j = 1, 2, \ldots$
  $\alpha_j = w_j^T A v_j;$
  $v_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1};$
  $w_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1};$
  $\delta_{j+1} = |w_{j+1}^T v_{j+1}|^{1/2};$
  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.

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

$$AV_k = V_k T_k + \gamma_{k+1} v_{k+1} e_k^T, \tag{10.7.41}$$
$$A^T W_k = W_k T_k^T + \beta_{k+1} w_{k+1} e_k^T. \tag{10.7.42}$$

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 (10.7.41)–(10.7.42) 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 by Wilkinson, occurs when $\tilde{w}_k^T \tilde{v}_k = 0$, with neither $\tilde{v}_{k+1}$ nor $\tilde{w}_{k+1}$ null.

10.7.7 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

$$r_k = b - Ax_k = \beta_1 v_1 - AV_k y_k = \beta_1 v_1 - V_k T_k y_k - \gamma_{k+1} \tilde{v}_{k+1} e_{k+1}^T y_k$$

where y_k is determined so that the Galerkin condition $r_k \perp \mathcal{K}_k(w_1, A^T)$ is satisfied. Using (10.7.41) and the bi-orthogonality conditions $W_k^T V_k = 0$ this gives

$$0 = W_k^T(\beta_1 v_1 - AV_k y_k) = \beta_1 e_1 - T_k y_k.$$

Hence, if the matrix T_k is nonsingular y_k is determined by solving the tridiagonal system $T_k y_k = \beta_1 e_1$. If A is symmetric, this method becomes the SYMMLQ method, see Section10.7.5. We remark again that 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.

The nonsymmetric Lanczos process 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 BCG method, see, e.g., Barret et al. [pp. 21–22, 1994]. This form is obtained from the Lanczos formulation by computing the LU decomposition of T_k. Hence the BCG method cannot proceed when a singular T_k occurs, and this is an additional cause for breakdown.

There are few results known about the convergence of the BCG and related methods. In practice it has been observed that sometimes convergence can be as fast as for GMRES. However, often 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 modification of BCG due to Sonneveld is **CGS**, which stands for "CG squared". Sonneveld observed that by reorganizing the BCG algorithm in a certain way one can obtain an algorithm for which, with the same amount of work, the errors and residuals satisfy

$$e_k = q_k^2(A)e_0, \quad r_k = q_k^2(A)r_0,$$

where q_k is the same polynomial as in BCG. Furthermore, no matrix-vector products with A^T are required. CGS tends to converge faster than BCG by a factor between 1 and 2, and is particularly attractive in case computations with A^T are impractical. The residual norms in CGS may behave very erratically and often a stabilized version of this method Bi-CGSTAB is preferred.

Another related method can be developed as follows. After k steps of the nonsymmetric Lanczos process we have from the relation (10.7.41) 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

$$\begin{aligned} b - Ax_k &= b - A(x_0 + V_k y) = r_0 - AV_k y \\ &= \beta v_1 - V_{k+1}\hat{T}_k y = V_{k+1}(\beta e_1 - \hat{T}_k y). \end{aligned} \tag{10.7.43}$$

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 $x_k = x_0 + V_k y_k$. This is called the **Quasi-Minimal Residual (QMR) method**.

Recent surveys on progress in iterative methods for non-symmetric systems are given by Freund, Golub and Nachtigal [4, 1991] and Golub and van der Vorst [7]. 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 [14] 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.

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 (BCG) 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 BCG method compared to GMRES.

   c) How are the approximations x_k defined in QMR?

Problems

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.

10.8 Preconditioned Iterative Methods

The term "preconditioning" dates back to Turing in 1948, and is in general taken to mean the transformation of a problem to a form that can more efficiently be solved. 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**

$$M^{-1}Ax = M^{-1}b. \tag{10.8.1}$$

or **right-preconditioned system**

$$AM^{-1}u = b, \qquad u = Mx. \tag{10.8.2}$$

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. In this case, it is natural to consider a **split preconditioner**. Let $M = LL^T$ where L is the Cholesky factor of M. Then we consider the preconditioned linear system

$$\tilde{A} = L^{-1}AL^{-T}, \qquad \tilde{x} = L^T x, \qquad \tilde{b} = L^{-1}b, \tag{10.8.3}$$

and the spectrum of $\tilde{A}$ is real. Note that the spectrum of $A = C^{-1}AC^{-T}$ is the same as for $L^{-T}L^{-1}A = M^{-1}A$.

10.8.1 The Preconditioned CG Method

The conjugate gradient algorithm 10.6.1 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 (10.8.3).

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

$$M = LL^T, \tag{10.8.4}$$

we can obtain the following implementation of the **preconditioned conjugate gradient method**:

Algorithm 10.8.1
Preconditioned Conjugate Gradient Method

$$r_0 = b - Ax_0; \quad p_0 = z_0 = M^{-1}r_0;$$
$$\textbf{for } k = 0, 1, 2, \ldots, \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do}$$
$$w = Ap_k;$$
$$\beta_k = (z_k, r_k)/(p_k, Ap_k);$$
$$x_{k+1} = x_k + \beta_k p_k;$$

$$
\begin{aligned}
r_{k+1} &= r_k - \beta_k A p_k; \\
z_{k+1} &= M^{-1} r_{k+1}; \\
\beta_k &= (z_{k+1}, r_{k+1})/(z_k, r_k); \\
p_{k+1} &= z_{k+1} + \beta_k p_k;
\end{aligned}
$$

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 (10.6.12). Note, however, that

$$\|\tilde{x} - \tilde{x}_k\|_{\tilde{A}}^2 = (\tilde{x} - \tilde{x}_k)^T L^{-1} A L^{-T} (\tilde{x} - \tilde{x}_k) = \|x - x_k\|_A^2,$$

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. [5].

10.8.2 Preconditioned CGLS and CGNE.

To precondition CGLS Algorithm (10.7.3) 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 10.8.2
Preconditioned CGLS.

$$
\begin{aligned}
&r_0 = b - Ax_0; \quad p_0 = s_0 = S^{-T}(A^T r_0);\\
&\textbf{for } k = 0, 1, \ldots \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do}\\
&\quad t_k = S^{-1} p_k;\\
&\quad q_k = A t_k;\\
&\quad \alpha_k = \|s_k\|_2^2 / \|q_k\|_2^2;\\
&\quad x_{k+1} = x_k + \alpha_k t_k;\\
&\quad r_{k+1} = r_k - \alpha_k q_k;\\
&\quad s_{k+1} = S^{-T}(A^T r_{k+1});\\
&\quad \beta_k = \|s_{k+1}\|_2^2 / \|s_k\|_2^2;\\
&\quad p_{k+1} = s_{k+1} + \beta_k p_k;\\
&\textbf{end}
\end{aligned}
$$

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 10.8.3
Preconditioned CGNE

$$
\begin{aligned}
&r_0 = b - Ax_0; \quad z_0 = S^{-1} r_0; \quad p_0 = A^T(S^{-T} z_0);\\
&\textbf{for } k = 0, 1, \ldots \textbf{ while } \|r_k\|_2 > \epsilon \textbf{ do}\\
&\quad \alpha_k = \|z_k\|_2^2 / \|p_k\|_2^2;\\
&\quad x_{k+1} = x_k + \alpha_k p_k;\\
&\quad r_{k+1} = r_k - \alpha_k A p_k;\\
&\quad z_{k+1} = S^{-1} r_{k+1};\\
&\quad \beta_k = \|z_{k+1}\|_2^2 / \|z_k\|_2^2;\\
&\quad p_{k+1} = A^T(S^{-T} z_{k+1}) + \beta_k p_k;\\
&\textbf{end}
\end{aligned}
$$

Algorithm PCCGLS still minimizes the error functional $\|\hat{r} - r^{(k)}\|_2$, where $r = b - Ax$, but over a different Krylov subspace

$$x^{(k)} = x^{(k)} + \mathcal{K}_k, \qquad \mathcal{K}_k = (S^{-1}S^{-T}A^TA, S^{-1}S^{-T}A^Tr_0).$$

Algorithm PCCGNE minimizes the error functional $\|\hat{x} - x^{(k)}\|_2$, over the Krylov subspace

$$x^{(k)} = x^{(k)} + \mathcal{K}_k, \qquad \mathcal{K}_k = (A^TS^{-T}S^{-1}A, A^TS^{-T}S^{-1}r_0).$$

The rate of convergence for PCGTLS depends on $\kappa(AS^{-1})$, and for PCCGNE on $\kappa(S^{-1}A) = \kappa(A^TS^{-T})$.

10.8.3 Preconditioned GMRES

For nonsymmetric linear systems there are two options for applying the preconditioner. We can use the left preconditioned system (10.8.1) or the right preconditioned system (10.8.2). (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

$$z_j = M^{-1}Av_j, \quad j = 1, 2, \ldots, k.$$

All computed residual vectors will be preconditioned residuals $M^{-1}(b - Ax_m)$. This is a disadvantage since most stopping criteria depend are based on 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

$$x_0 + \mathcal{K}_m(M^{-1}r_0, M^{-1}A). \tag{10.8.5}$$

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

$$z_j = AM^{-1}v_j, \quad j = 1, 2, \ldots, k.$$

The kth approximation is $x_k = x_0 + M^{-1}V_ky_k$, where y_k solves

$$\min_{y_k} \|\beta_1 e_1 - \bar{H}_k y_k\|_2.$$

As before this can be written as

$$x_k = x_{k-1} + \beta_1\tau_k M_k^{-1}w_k, \quad w_k = R_k y_k,$$

see (10.7.24).

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

$$x_0 + M^{-1}\mathcal{K}_m(r_0, AM^{-1}). \tag{10.8.6}$$

Somewhat surprisingly the two affine subspaces (10.8.5) and (10.8.6) 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 $\Lambda = \mathrm{diag}(\lambda_i)$ we proved the error estimate

$$\frac{||r_k||_2}{||r_0||_2} \leq \kappa_2(X) \min_{q_k} \max_{i=1,2,\ldots,n} |q_k(\lambda_i)|, \tag{10.8.7}$$

where q_k is a polynomial of degree $\leq k$ and $q_k(0) = 1$. Because of the factor $\kappa_2(X)$ in (10.8.7) 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 that the convergence behavior will be similar when if A is close to normal.

10.9 Preconditioners

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.

10.9.1 Preconditioners from Matrix Splittings

The stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \qquad k = 0, 1, \ldots, \tag{10.9.1}$$

corresponds to a matrix splitting $A = M - N$, and the iteration matrix

$$B = M^{-1}N = I - M^{-1}A.$$

The iteration (10.9.1) 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 10.2 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 10.9.1. Van der Sluis [1969]

Let $A = D_A - L_A - L_A^T$ be a symmetric positive definite matrix. Then if A has at most q nonzero elements in any row it holds that

$$\kappa(D_A^{-1/2} A D_A^{-1/2}) = \min_{D>0} \kappa(DAD).$$

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 10.3.2 it was shown that for a symmetric matrix A the SSOR iteration method corresponds to a splitting with the matrix

$$M_{SSOR} = \frac{1}{\omega(2-\omega)} (D_A - \omega L_A) D_A^{-1} (D_A - \omega U_A).$$

Since M_{SSOR} is given in the form of an LDL^T factorization it is easy to solve linear systems 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).

10.9.2 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 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 10.9.1
Incomplete LU Factorization

for $k = 1, \ldots, n-1$
  for $i = k+1, \ldots, n$
    if $(i,k) \in \mathcal{P}$ $l_{ik} = a_{ik}/a_{kk}$;
    for $j = k+1, \ldots, n$
      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

$$\widetilde{A}_k = A_{k-1} + R_k, \qquad A_k = L_k \widetilde{A_k}, \quad k = 1, \ldots, n-1.$$

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 $L_k R_m = R_m$, 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 10.9.2 can be improved by noting that any elements in the resulting $(n-k)\times(n-k)$ lower part of the reduced matrix not in $\mathcal{P}$ need not be carried along and can also be included in the defect matrix R_k. This is achieved simply by changing line five in the algorithm to

$$\textbf{if } (k,j) \in \mathcal{P} \ \textbf{ and } (i,j) \in \mathcal{P} \ \ a_{ij} = a_{ij} - l_{ik}a_{kj};$$

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 10.9.2.

Example 10.9.1. For the model problem using a five-point approximation the non-zero structure of the resulting matrix is given by

$$\mathcal{P}_A = \{(i,j) \,|\, |i-j| = -n, -1, 0, 1, n\}.$$

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. Many matrices arising from the discretization of partial differential equations have the following property.

Definition 10.9.2. *A matrix $A = (a_{ij})$ is an M-matrix if $a_{ij} \leq 0$ for $i \neq j$, A is nonsingular and $A^{-1} \geq 0$.*

The following result was proved by Meijerink and van der Vorst [13, 1977].

Theorem 10.9.3.
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. In particular the matrix arising from the model problem is a symmetric M-matrix. A symmetric M-matrix is also called a **Stieltjes** matrix.

An implementation of the incomplete Cholesky factorization in the general case is given below.

Algorithm 10.9.2
Incomplete Cholesky Factorization

for $j = 1, 2, \ldots, n$

$$l_{jj} = \Big(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\Big)^{1/2};$$

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

10.9.3 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

$$A = \begin{pmatrix} D_1 & A_2^T & & & \\ A_2 & D_2 & A_3^T & & \\ & A_3 & \ddots & \ddots & \\ & & \ddots & \ddots & A_N^T \\ & & & A_N & D_N \end{pmatrix} = D_A - L_A - L_A^T, \tag{10.9.2}$$

with square diagonal blocks D_i. For the model problem with the natural ordering of mesh points we obtain this form with $A_i = -I$, $D_i = \text{tridiag}(-1 \quad 4 \quad -1)$. If systems with D_i can be solved efficiently a simple choice of preconditioner is the block diagonal preconditioner

$$M = \text{diag}\,(D_1, D_2, \ldots, D_N).$$

The case $N = 2$ is of special interest. For the system

$$\begin{pmatrix} D_1 & A_2^T \\ A_2 & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \tag{10.9.3}$$

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 (10.3.4) 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 (10.9.3) then we obtain

$$Sx_2 = b_2 - A_2 D_1^{-1} b_1, \qquad S = D_2 - A_2 D_1^{-1} A_2^T, \tag{10.9.4}$$

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 (10.9.4). 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 (10.9.4) 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 [3], which has proved to be very useful. We assume in the following that in (10.9.2) 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

$$\Delta_1 = D_1, \qquad \Delta_i = D_i - A_i \Lambda_{i-1}^{-1} A_i^T, \quad i = 2, \ldots, N. \tag{10.9.5}$$

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 $T^{-1} = L^{-T}L$ 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 $L^{-T}(1)L^{-1}(1)$ a tridiagonal matrix. Since we have assumed that A_i are diagonal matrices, the approximations Δ_i generated by (10.9.5) will in this case be tridiagonal.

10.9.4 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

$$A = \begin{pmatrix} B & T & & & \\ T & B & T & & \\ & T & B & \ddots & \\ & & \ddots & \ddots & T \\ & & & T & B \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{pmatrix}, \tag{10.9.6}$$

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 have a common eigensystem, and we let

$$Q^T B Q = \Lambda = \operatorname{diag}(\lambda_i), \qquad Q^T T Q = \Omega = \operatorname{diag}(\omega_i).$$

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 $x_j = Qz_j$, $y_j = Q^T b_j$, $j = 1, \ldots, q$.

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
$$y_j = Q^T b_j, \qquad j = 1, \ldots, q.$$

2. Rearrange taking one element from each vectors y_j,
$$\hat{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iq})^T, \quad i = 1, \ldots, p,$$

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

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*:

$$A_k B_l = \begin{cases} C_{k+l}, & \text{if } k + l \leq n - 1; \\ 0, & \text{otherwise,} \end{cases}$$

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 $S = B_2 - C B_1^{-1} C^T$ 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.

Computer Exercises

1. 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 $i = j$.

(a) 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.

(b) 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?

(c) 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!

(d) 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

Two classical texts on iterative methods for linear systems are Varga [18, 1962] and Young [21, 1971]. Axelsson [1, 1994] is an excellent source of more modern material. The book by Barret et al. [2, 1994] gives a compact survey of iterative methods and their implementation. An up to date treatment of iterative methods and advanced preconditioners are given by Saad [17, 1996]. See also available software listed in Chapter 15.

For the case of a square matrix A this method was originally devised by Kaczmarz [10, 1937]. 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).

Krylov subspace iteration, 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 [9], 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 has developed a closely related method (see Chapter 9).

References

[1] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, UK, 1994.

[2] R. Barret, M. W. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, editors. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, PA, 1993.

[3] P. Concus, G. H. Golub, and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Statist. Comput.*, 6:220–252, 1985.

[4] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1991.

[5] P. E. Gill, W. Murray, D. B. Ponceleón, and M. A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix. Anal. Appl.*, 13:292–311, 1992.

[6] G. H. Golub and D. O'Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Review*, 31:50–102, 1989.

[7] G. H. Golub and H. 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.

[8] A. Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, Philadelphia, PA, 1997.

[9] 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.

[10] S. Kaczmarz. Angenäherte auflösung von systemen linearen gleichungen. *Acad. Polon. Sciences et Lettres*, pages 355–357, 1937.

[11] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards, Sect. B*, 45:255–282, 1950.

[12] L. Landweber. An iterative formula for fredholm integral equations of the first kind. *Amer. J. Math.*, 73:615–624, 1951.

[13] J. A. Meijerink and H. 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.

[14] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM J. Matrix. Anal. Appl.*, 13:778–795, 1992.

[15] C. C. Paige and M. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[16] J. K. Reid. On the method of conjugate gradients for the solution of large systems of linear equations. In J. K. Reid, editor, *Large Sparse Sets of Linear Equations*, pages 231–254. Academic Press, New York, 1971.

[17] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, MA, 1996.

[18] R. S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, 1962.

[19] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, UK, 1965.

[20] D. M. Young. *Iterative methods for solving partial differential equations of elliptic type.* PhD thesis, Harward University, 1950.

[21] D. M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.

[22] D. M. Young. A historical overview of iterative methods. *Computer Phys. Comm.*, 53:1–17, 1989.

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

$$f_i(x_1, x_2, \ldots, x_n) = 0, \qquad i = 1, 2, \ldots, n. \tag{11.1.1}$$

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 11.3.

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

$$c_i(x) = 0, \quad i = 1, \ldots, m_1, \quad c_i(x) \geq 0, \quad i = m_1 + 1, \ldots, m. \tag{11.1.6}$$

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 Generalized Linear Methods

In Chapter 5 we developed methods for solving a single nonlinear equation $f(x) = 0$, $f: \mathbf{R} \to \mathbf{R}$. A simple way to extend these methods for solving a nonlinear system (11.1.1) is as follows.

Given approximations $x_1^{(k)}, \ldots, x_n^{(k)}$, we use the ith equation to solve for a new approaimation for x_i,

$$f_i\left(x_1^{(k)}, \ldots, x_i, \ldots, x_n^{(k)}\right) = 0, \qquad i = 1, 2, \ldots, n, \tag{11.1.7}$$

and take the result as $x_i^{(k+1)}$. (Note that this assumes that the f_i depends on the variable x_i.) We repeat this for $k = 0, 1, 2, \ldots$, where $x^{(0)}$ is some initial approximation. The resulting method is called the **nonlinear Jacobi** method. Since (11.1.7) is a nonlinear equation in one unknown, the methods developed in Chapter 5 can be used to solve it.

An obvious variation is to use the most recent approximations for the other components and instead of (11.1.7) solve

$$f_i\left(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i, \ldots, x_n^{(k)}\right) = 0, \qquad i = 1, 2, \ldots, n. \tag{11.1.8}$$

This gives the **nonlinear Gauss–Seidel** method. Note that in this method the *orderings of equations and variables* are important. This method can be generalized in an obvious manner to give the **nonlinear SOR** method. A great many variations are possible corresponding to which method is used to solve the secondary one-dimensional problem. In this way methods are obtained, which can be called Jacobi-Newton, Gauss–Seidel–Secant, etc. For a further discussion of such methods we refer to Ortega and Rheinboldt, [28, 1970, Section 7.4].

11.1.3 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

$$x_i = g_i(x_1, x_2, \ldots, x_n), \qquad i = 1, 2, \ldots, n,$$

suggests an iterative method where, for $k = 0, 1, 2, \ldots$, we compute

$$x_i^{(k+1)} = g_i\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right), \qquad i = 1, 2, \ldots, n, \tag{11.1.9}$$

Using vector notations this can be written

$$x^{(k+1)} = g\big(x^{(k)}\big), \qquad k = 0, 1, 2, \ldots, \tag{11.1.10}$$

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 (11.1.10). 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.,

$$u, v \in S_r \Rightarrow \|T(u) - T(v)\| \le L\|u - v\|, \tag{11.1.11}$$

where $L < 1$. Then if

$$\|u_0 - T(u_0)\| \le (1 - L)r \tag{11.1.12}$$

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

$$\|u_k - u^*\| \le \|u_k - u_{k-1}\|\frac{L}{1-L} \le \|u_1 - u_0\|\frac{L^k}{1-L}. \tag{11.1.13}$$

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 (11.1.12) 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 (11.1.11) 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 (11.1.11) T is continuous, and it follows that $\lim_{k\to\infty} T(u_k) = T(u^*) = u^*$. The demonstration of the error estimates (11.1.13) 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[1] 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.

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*

$$\nabla\phi(x) = \Big(\frac{\partial f_i}{\partial x_1}, \ldots, \frac{\partial f_i}{\partial x_n}\Big)^T \in \mathbf{R}^n \tag{11.1.14}$$

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

$$J(x) = \begin{pmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_n(x)^T \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \in \mathbf{R}^{n\times n}, \tag{11.1.15}$$

is called the **Jacobian** *of f.*

[1] 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 [12, 1961].

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

$$\|f(x) - f(a)\| < \int_0^1 \|f'(a + t(x-a))\| \, \|x-a\| dt \le \max_{y \in \mathcal{D}} \|f'(y)\| \, \|x-a\|.$$

□

11.1.4 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[2] and assume that $f_i(x)$ is twice differentiable at x_k. Then by Taylor's formula

$$f_i(x) = f_i(x_k) + (\nabla f_i(x_k))^T (x - x_k) + O(\|x - x_k\|^2), \qquad i = 1, \ldots, n.$$

Using the Jacobian matrix (11.1.15) 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

$$J(x_k)(x - x_k) = -f(x_k). \tag{11.1.16}$$

If $J(x_k)$ is nonsingular then (11.1.16) has a unique solution x_{k+1}, which can be expected to be a better approximation. The resulting iterative algorithm can be written

$$x_{k+1} = x_k - \big(J(x_k)\big)^{-1} f(x_k). \tag{11.1.17}$$

[2] In the following we use vector notations so that x_k will denote the kth approximation and not the kth component of x.

which is **Newton's method**. Note that, in general the inverse Jacobian matrix should not be computed. Instead the linear system (11.1.16) 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

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*

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} = C, \tag{11.1.18}$$

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 (11.1.18) 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 [11, pp. 19–21], and Chapter 9 of Ortega and Rheinboldt [28].)

In order to analyze the convergence of Newton's method we need to study how well the linear model (11.1.16) 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

$$\|J(x) - J(y)\| \le \gamma\|x - y\|, \qquad \forall x, y \in \mathcal{D},$$

where $\mathcal{D} \subset \mathbf{R}^n$ is a convex set. Then for all $x, y \in \mathcal{D}$ it holds that

$$\|f(x) - f(y) - J(y)(x - y)\| \le \frac{\gamma}{2}\|x - y\|^2.$$

Proof. The function $g(t) = f(y + t(x - y))$, $x, y \in \mathcal{D}$ is differentiable for all $0 \le t \le 1$, and by the chain rule $g'(t) = J(y + t(x - y))(x - y)$. It follows that

$$\|g'(t) - g'(0)\| = \|\big(J(y + t(x - y)) - J(y)\big)(x - y)\| \le \gamma t\|x - y\|^2. \qquad (11.1.19)$$

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 (11.1.19) it follows that

$$\|f(x) - f(y) - J(y)(x - y)\| \le \int_0^1 \|g'(t) - g'(0)\|dt \le \gamma\|x - y\|^2 \int_0^1 tdt.$$

□

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) $\qquad \|J(x) - J(y)\| \le \gamma\|x - y\|, \qquad \forall \; x, y \in S_r(x^*),$

(b) $J(x^*)^{-1}$ *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*

$$x_{k+1} = x_k - J(x_k)^{-1} f(x_k), \quad k = 0, 1, \ldots$$

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

$$\begin{aligned} \|x_1 - x^*\| &\le \|J(x_0)^{-1}\|\,\|f(x^*) - f(x_0) - J(x_0)(x^* - x_0)\| \\ &\le 2\beta\gamma/2\|x_0 - x^*\|^2, \end{aligned}$$

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 [28, 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,

$$J(x_p)(x_{k+1} - x_k) = -f(x_k), \quad k = p, \ldots, p + m - 1. \tag{11.1.20}$$

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

$$\|g'(x)\| \leq \|J(x_p)^{-1}\| \, \|J(x_p) - J(x)\| \leq \beta\gamma\|x_p - x\|.$$

Since $g' \neq 0$ the modified Newton method will only have *linear* rate of convergence. Also, far away from the solution the modified method may diverge in cases where Newton's method converges.

Example 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

11.1.5 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

$$\frac{\partial f(x)}{\partial x_j} \approx \Delta_j f(x) \equiv \frac{f(x + h_j e_j) - f(x)}{h_j}, \quad j = 1, \ldots, n,$$

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 $D = \text{diag}(h_1, h_2, \ldots, h_n)$ is a nonsingular diagonal matrix. This shows that $J(x_k, d)$ is nonsingular if and only if the vectors

$$f(x_k + h_j e_j) - f(x_k), \quad j = 1, 2, \ldots, n,$$

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

$$\frac{f(x + h_j q_k) - f(x)}{h_j}, \quad j = 1, \ldots, n, \quad k = j, \ldots, n,$$

where $Q = (q_1, \ldots, q_n)$ 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 [26, 1979].

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

$$B_k h_k = -f(x_k), \tag{11.1.21}$$

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**

$$B_{k+1} s_k = y_k \tag{11.1.22}$$

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

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k}. \tag{11.1.23}$$

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 (11.1.23). 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 [10]. □

We can compute B_{k+1} from (11.1.23) in only $2n^2$ operations and no extra function evaluations. To solve (11.1.21) 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 $G_{j,j+1}$, $j = n-1, \ldots, 1$ such that

$$G_{1,2}^T \ldots G_{n-1,n}^T u_k = \alpha e_1, \qquad \alpha = \pm \|u_k\|_2.$$

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

$$\bar{H} = G_{1,2}^T \ldots G_{n-1,n}^T (R_k + u_k v_k^T) = H + \alpha e_1 v_k^T.$$

It is easily verified that in the product $H = G_{1,2}^T \ldots G_{n-1,n}^T R_k$ the Givens rotations will introduce extra nonzero elements only in positions $(j, j+1)$, $j = 1, 2, \ldots, n$, 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 $\bar{h}_{j+1,j}$, $j = 1, \ldots, n-1$. Then

$$\bar{G}_{n-1,n}^T \ldots \bar{G}_{1,2}^T \bar{H} = \bar{R}_{k+1}$$

is the updated triangular factor. The orthogonal factor equals the product

$$\bar{Q}_k = G_{n-1,n} \ldots G_{1,2} \bar{G}_{1,2} \ldots \bar{G}_{n-1,n}.$$

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.

11.1.6 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

$$\min_x \phi(x), \qquad \phi(x) = \frac{1}{2}\|f(x)\|_2^2 = \frac{1}{2} f(x)^T f(x). \tag{11.1.24}$$

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 11.2.2.

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

$$x_{k+1} = x_k + \alpha_k h_k, \qquad J(x_k)h_k = -f(x_k). \tag{11.1.25}$$

where the step length α_k is computed by a **line search**. Ideally α_k should be chosen to be minimize the scalar function

$$\psi(\alpha) = \phi\|f(x_k + \alpha h_k)\|_2^2.$$

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 $1, \frac{1}{2}, \frac{1}{4}, \ldots$ 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 (11.1.25) 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 11.3.4. Here one takes $x_{k+1} = x_k + d_k$, where d_k solves the constrained linear least squares problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2^2, \text{ subject to } \|d_k\|_2 \leq \Delta_k,$$

where Δ_k is a parameter, which is updated recursively. If the constraint is binding this problem can be solved by introducing a Lagrange parameter λ and minimizing

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2^2 + \lambda\|d_k\|_2^2. \tag{11.1.26}$$

Here λ is determined by the **secular equation** $\|d_k(\lambda)\|_2 = \Delta_k$. Note that the problem (11.1.26) 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 (11.1.25), 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.

11.1.7 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**)

$$H(x,t) = tf(x) + (1-t)g(x), \tag{11.1.27}$$

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 (11.1.27) 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

$$H(x, t_{j+1}) = 0, \qquad j = 0, \ldots, p-1, \tag{11.1.28}$$

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 (11.1.28)

$$x_{k+1} = x_k - \left(\frac{\partial H(x_k, t_{j+1})}{\partial x}\right)^{-1} H(x_k, t_{j+1}).$$

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 (11.1.27) 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 (11.1.28) very coarsely and then locally use a Newton-type iterative method for solving (11.1.27) as a corrector. This has the advantage that one takes advantage of the fact that the solution curve consists of solutions of (11.1.28), and uses the resulting strong contractive properties of Newton's method. Such predictor corrector continuation methods have been very successful, see Allgower and Georg [1, 1990]. The following algorithm uses Euler's method for integration as a predictor step and Newton's method as a corrector:

Algorithm 11.1.1 *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 [1, 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 $h = 0, 0.01, \ldots, 0.10$. 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,

$$x^2 - 2x - y + 1 = 0$$
$$x^2 + y^2 - 1 = 0$$

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 Fig. 11.1.1. It can be shown that the angles are determined by the equations

$$\tan x_1 - k(2 \sin x_1 + \sin x_2) = 0,$$
$$\tan x_2 - 2k(\sin x_1 + \sin x_2) = 0.$$

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

$$k = 0.30, 0.31, \ldots, 0.35, 0.4, 0.5, \ldots, 0.9, 1, 2, 3, 4, 5, 10, 15, 20, \infty.$$

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

$$x_1 \approx x_2 \approx \sqrt{k - (1 - 1/\sqrt{2})}, \qquad 0 \leq k - (1 - 1/\sqrt{2}) \ll 1.$$

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.

11.2 Unconstrained Optimization

11.2.1 Optimality Conditions

Consider an unconstrained optimization problem of the form

$$\min_x \phi(x), \qquad x \in \mathbf{R}^n. \tag{11.2.1}$$

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 11.2.1.
A point x^* is said to be a **local minimum** *of ϕ if $\phi(x^*) \leq \phi(y)$ for all y in a sufficiently small neighborhood of x^*. If $\phi(x^*) < \phi(y)$ then x^* is a* **strong** *local minimum.*

Assume that the objective function ϕ is continuously differentiable at a point x 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 11.2.2.
A point x^* which satisfies $g(x) = \nabla\phi(x) = 0$ is called a **stationary point**.

Definition 11.2.3.
A function $\phi : \mathbf{R}^n \to \mathbf{R}$ is twice continuously differentiable at x, if

$$g_{ij} = \frac{\partial}{\partial x_i}\left(\frac{\partial\phi}{\partial x_j}\right) = \frac{\partial^2\phi}{\partial x_i \partial x_j}, \qquad 1 \leq i, j \leq n.$$

exist and are continuous. The square matrix $H(x)$ formed by these n^2 quantities is called the **Hessian** *of $\phi(x)$,*

$$H(x) = \nabla^2\phi(x) = \begin{pmatrix} \frac{\partial^2\phi}{\partial x_1^2} & \cdots & \frac{\partial^2\phi}{\partial x_n \partial x_1} \\ \vdots & & \vdots \\ \frac{\partial^2\phi}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2\phi}{\partial x_n^2} \end{pmatrix} \in \mathbf{R}^{n\times n}. \tag{11.2.2}$$

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 11.2.4.
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 $0 \leq \theta \leq 1$, ϵ is a scalar and d a vector. Assume that $g(x^*) \neq 0$ and choose d so that $d^T g(x^*) < 0$. Then for sufficiently small $\epsilon > 0$ the last term is negligible and $\phi(x^* + \epsilon d) < \phi(x^*)$. □

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 Fig. 11.2.1.

Figure 11.2.1. *A saddle point.*

11.2.2 Steepest Descent

In many iterative methods for minimizing a function $\phi(x) : \mathbf{R}^n \to \mathbf{R}$, a sequence of points $\{x_k\}$, $k = 0, 1, 2, \ldots$ are generated from

$$x_{(k+1)} = x_k + \alpha_k d_k, \tag{11.2.3}$$

where d_k is a **search direction** and α_k a **step length**. If we put

$$f(\alpha) = \phi(x_k + \alpha_k d_k), \tag{11.2.4}$$

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

$$d_k = -g(x_k)/||g(x_k)||_2. \tag{11.2.5}$$

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, see Fig. 11.4.1. Hence this method is usually used only as a starting step, or when other search directions fail.

Example 11.2.1.

If the steepest descent method is applied to a quadratic function

$$\phi(x) = b^T x + \frac{1}{2} x^T G x,$$

where G is a symmetric positive definite matrix. Then 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!

11.2.3 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** $\phi(x_k + s_k) \approx q_k(s_k)$,

$$q_k(s_k) = \phi(x_k) + g(x_k)^T s_k + \frac{1}{2} s_k^T H(x_k) s_k, \tag{11.2.6}$$

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

$$H(x_k) s_k = -g(x_k). \tag{11.2.7}$$

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 11.2.2 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

$$\phi(x_k + \lambda_k d_k) \leq \phi(x_k) + \mu\lambda_k g(x_k)^T d_k, \tag{11.2.8}$$

$$\left|g(x_k + \lambda_k d_k)^T d_k\right| \leq \eta \left|g(x_k)^T d_k\right|, \tag{11.2.9}$$

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 $g_k^T H(x_k)^{-1} g_k \leq 0$. This situation is not likely to occur in the vicinity of a local optimum x^*, because of the positive (or at least nonnegative) definiteness of $H(x^*)$. Far away from an optimal point, however, this can happen. This is the reason for admitting the gradient as an alternative search direction—especially since there is a danger that the Newton direction will lead to a saddle point.

In the quadratic model the term $s_k^T H(x_k) s_k$ can be interpreted as the curvature of the surface $\phi(x)$ at x_k along s_k. Often $H(x_k)$ is expensive to compute, and we want to approximate this term. Expanding the gradient function in a Taylor series about x_k along a direction s_k we have

$$g(x_k + s_k) = g_k + H(x_k)s_k + \ldots. \tag{11.2.10}$$

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.,

$$B_{k+1} s_k = \gamma_k, \qquad \gamma_k = g(x_{k+1}) - g(x_k), \tag{11.2.11}$$

where γ_k is the change in the gradient. The first equation in (11.2.11) is the analog of the secant equation (11.2.10) 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

$$B_{k+1} = B_k + \frac{r_k s_k^T + s_k r_k^T}{s_k^T s_k} - \frac{(r_k^T s_k) s_k s_k^T}{(s_k^T s_k)^2}, \tag{11.2.12}$$

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 (11.2.11).

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

$$B_k d_k = -g_k, \qquad k \geq 1, \tag{11.2.13}$$

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 (11.2.13) 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 (11.2.13), 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 (11.2.13) 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. [15, 1975]. Let $B = LDL^T$ be the Cholesky factorization of B, where $L = (l_{ij})$ is unit lower triangular and $D = \text{diag}\,(d_j) > 0$ diagonal. Let $\bar{B} = B \pm vv^T$ be a rank-one modification of B. Then we can write

$$\bar{B} = LDL^T \pm vv^T = L(D \pm pp^T)L^T,$$

where p is the solution of the triangular system $Lp = v$. The Cholesky factorization $D \pm pp^T = \hat{L}\bar{D}\hat{L}^T$ 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

$$\phi(x) = \frac{1}{2} x^T G x - b^T x + c,$$

   where $G \in \mathbf{R}^{n \times n}$ is a symmetric matrix and $b \in \mathbf{R}^n$ a column vector. Show that the gradient of ϕ is $g = Gx - b$ and the Hessian is G. Also show that if $g(x^*) = 0$, then

$$\phi(x) = \phi(x^*) + \frac{1}{2}(x - x^*)^T H (x - x^*).$$

   (b) Suppose that G is symmetric and nonsingular. Using the result from (a) show that Newton's method will find a stationary point of ϕ in one step from an arbitrary starting point x_0. Under what condition is this a minimum point?
2. Let $\psi(x)$ be quadratic with Hessian matrix G, which need not be positive definite.

   (a) Let $\psi(\lambda) = \phi(x_0 - \lambda d)$. Show using Taylor's formula that

$$\psi(\lambda) = \psi(0) - \lambda g^T d + \frac{1}{2}\lambda^2 d^T G d.$$

   Conclude that if $d^T G d > 0$ for a certain vector d then $\psi(\lambda)$ is minimized when $\lambda = g^T d / d^T G d$, and

$$\min_\lambda \psi(\lambda) = \psi(0) - \frac{1}{2}\frac{(d^T g)^2}{d^T G d}.$$

   (b) Using the result from (a) show that if $g^T G g > 0$ and $g^T G^{-1} g > 0$, then the steepest descent method $d = g$ with optimal λ gives a smaller reduction of ψ than Newton's method if $g^T G^{-1} g > (g^T g)^2 / g^T G g$. (The conclusion holds also if $\phi(x_0 - \lambda d)$ can be approximated by a quadratic function of λ reasonably well in the relevant intervals.)

   (c) Suppose that G is symmetric and nonsingular. Using the result from (b) show

that Newton's method will find a stationary point of ϕ in one step from an arbitrary starting point x_0. Under what condition is this a minimum point?

11.3 Nonlinear Least Squares Problems

11.3.1 Introduction

In this section we discuss the numerical solution of nonlinear least squares problem. Let $f: \mathbf{R}^n \to \mathbf{R}^m$, $m \geq n$, and consider the problem

$$\min_{x \in \mathbf{R}^n} \phi(x), \qquad \phi(x) = \tfrac{1}{2}\|f(x)\|_2^2 = \tfrac{1}{2} f(x)^T f(x). \tag{11.3.1}$$

This is a special case of the general optimization problem in $\mathbf{R}^n$ studied in Section 11.2. We will in the following mainly emphasize those aspects of the problem (11.3.1), which derive from the special form of $\phi(x)$. (Note that the nonlinear system $f(x) = 0$ is equivalent to (11.3.1) 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), $i = 1, \ldots, m$ 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,

$$r_i(x) = y_i - h(x, t_i), \qquad i = 1, \ldots, m,$$

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 (11.3.1) with $f(x) = r(x)$.

Example 11.3.1.

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), $i = 1, 2, \ldots, m$, 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 11.3.6.

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

$$g(x) = \nabla \phi(x) = J(x)^T f(x), \tag{11.3.2}$$

where $J(x) \in \mathbf{R}^{m \times n}$ is the Jacobian matrix of $f(x)$. The Hessian matrix is

$$H(x) = \nabla^2 \phi(x) = J(x)^T J(x) + Q(x), \quad Q(x) = \sum_{i=1}^{m} f_i(x) G_i(x), \tag{11.3.3}$$

where $G_i(x) \in \mathbf{R}^{n\times n}$, is the Hessian matrix of $f_i(x)$ with elements

$$G_i(x)_{jk} = \frac{\partial^2 f_i(x)}{\partial x_j \partial x_k}, \quad j,k = 1,\ldots,n, \quad i = 1,2,\ldots,m. \tag{11.3.4}$$

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

$$g(x^*) = J(x^*)^T f(x^*) = 0. \tag{11.3.5}$$

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 (11.3.1). 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

$$\tilde{f}(x) = f(x_k) + J(x_k)(x - x_k), \tag{11.3.6}$$

and use the solution p_k to the linear least squares problem

$$\min_p \|f(x_k) + J(x_k)p\|_2. \tag{11.3.7}$$

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 11.3.2.

In the second approach (11.3.1) is viewed as a special case of unconstrained optimization in $\mathbf{R^n}$. A quadratic model at a point x_k is used,

$$\tilde{\phi}_c(x) = \phi(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k), \tag{11.3.8}$$

where the gradient and Hessian of $\phi(x) = \frac{1}{2}f^T(x)f(x)$ are given by (11.3.2) and (11.3.3). The minimizer of $\tilde{\phi}_c(x)$ is given by $x_{k+1} = x_k + p_k$, where

$$p_k = -H(x_k)^{-1}J(x_k)^T f(x_k). \tag{11.3.9}$$

This method is equivalent to Newton's method applied to (11.3.1), 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 (11.3.4) 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 $G_i(x)$, $i = 1, \ldots, m$, 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.

11.3.2 Gauss–Newton-Type Methods

The Gauss–Newton method for problem (11.3.1) is based on a sequence of linear approximations of $f(x)$ of the form (11.3.6). If x_k denotes the current approximation then the Gauss–Newton step d_k is a solution to the linear least squares problem

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \qquad d_k \in \mathbf{R}^n. \tag{11.3.10}$$

and the new approximation is $x_{k+1} = x_k + d_k$. The solution d_k is unique if $\text{rank}(J(x_k)) = n$. Since $J(x_k)$ may be ill-conditioned or singular, d_k should be computed by a stable method using, e.g., the QR- 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

$$d_k^T g(x_k) = -f(x_k)^T J^\dagger(x_k)^T J(x_k)^T f(x_k) = -\|P_{J_k} f(x_k)\|_2^2, \tag{11.3.11}$$

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 11.3.2. (Gill, Murray and Wright [17, 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.

11.3.3 Convergence of Gauss–Newton-Type Methods

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. This is illustrated by the following example.

Example 11.3.3. (Fletcher [1980, p.94])
Consider the problem with $m = 2, n = 1$ given by

$$\begin{aligned} f_1(x) &= x + 1 \\ f_2(x) &= \lambda x^2 + x - 1 \end{aligned}$$

where λ is a parameter. The minimizer of $\|f(x)\|_2$ is $x^* = 0$, and $\|f(x^*)\|_2 = \sqrt{2}$. It can be shown that for the Gauss–Newton method

$$x_{k+1} = \lambda x_k + 0(x_k^2)$$

and therefore this method is not locally convergent when $|\lambda| > 1$. The second derivative term in the Hessian $Q(x) = 2\lambda f_2(x)$ for this problem is not small close to the solution!

To analyze the rate of convergence of Gauss–Newton type methods let $J^\dagger(x)$ denote the pseudoinverse of $J(x)$, and assume that $\operatorname{rank}(J(x)) = n$. Then $I = J^\dagger(x)J(x)$, and (11.3.3) can be written in the form

$$H(x) = J(x)^T (I - \gamma K(x)) J(x), \qquad K(x) = J^\dagger(x)^T G_w(x) J^\dagger(x). \qquad (11.3.12)$$

where $\gamma = \|f(x)\|_2 \neq 0$, and

$$G_w(x) = \sum_{i=1}^{m} w_i G_i(x), \qquad w(x) = -\frac{1}{\gamma} f(x). \tag{11.3.13}$$

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

$$1 - \gamma\kappa_1 > 0. \tag{11.3.14}$$

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^*.

Figure 11.3.1. *Geometry of the data fitting problem for $m = 2$, $n = 1$.*

Example 11.3.4.

The geometrical interpretation of the nonlinear least squares problem (11.3.1) is to find a point on the surface $f(x) \in \mathbf{R}^m$ 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 Fig. 11.4.1 for the simple case of $m = 2$ observations and a scalar parameter x. Since in the figure we have $\gamma = \|y - z(x^*)\|_2 < \rho$, it follows that $1 - \gamma\kappa_1 > 0$, which is consistent with the fact that x^* is a local minimum. In general the solution (if it exists) is given by an orthogonal

projection of y onto the surface $z(x)$. Compare the geometrical intepretation in Fig. 7.2.1 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 (11.3.12) 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. $|\kappa_i|$, $i = 1, \ldots, n$ 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 11.1.6 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

$$\rho_{\text{est}} = \|P_J(x_{k+1})r_{k+1}\|_2 / \|P_J(x_k)r_k\|_2 = \rho + O(\|x_k - x^*\|_2^2). \tag{11.3.15}$$

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.

11.3.4 Trust Region Methods

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 11.3.5) 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 [25, 1944] and Marquardt [23, 1963]. Here a search direction d_k is computed by solving the problem

$$\min_{d_k}\{\|f(x_k) + J(x_k)d_k\|_2^2 + \mu_k\|d_k\|_2^2\}, \tag{11.3.16}$$

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

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \quad \text{subject to} \quad \|d_k\|_2 \leq \delta_k, \tag{11.3.17}$$

where $\mu_k = 0$ if the constraint in (11.3.17) 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

$$\min_{d_k} \|f(x_k) + J(x_k)d_k\|_2, \quad \text{subject to} \quad \|D_k d_k\|_2 \leq \delta_k,$$

where D_k is a diagonal scaling matrix.

(b) Compute the ratio $\rho_k = \left(\|f(x_k)\|_2^2 - \|f(x_k + d_k)\|_2^2\right)/\psi_k(d_k)$, 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 11.3.5 are somewhat more efficient but also more complex than the Levenberg-Marquardt methods.

11.3.5 Newton-Type Methods

The analysis in the Section 11.3.2 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 (11.3.8), can be used to overcome these problems. The optimal point d_k of this quadratic model, satisfies the linear system

$$H(x_k)d_k = -J(x_k)^T f(x_k), \tag{11.3.18}$$

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 [11, 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

$$B_k s_k = y_k, \qquad s_k = x_k - x_{k-1}, \qquad y_k = g(x_k) - g(x_{k-1}), \tag{11.3.19}$$

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 (11.3.19) can now be written

$$S_k s_k = z_k, \qquad z_k = \big(J(x_k) - J(x_{k-1})\big)^T f(x_k), \tag{11.3.20}$$

where S_k is required to be symmetric. It can be shown that a solution to (11.3.20) which minimizes the change from S_{k-1} in a certain weighted Frobenius norm is given by the update formula

$$B_k = B_{k-1} + \frac{w_k y_k^T + y_k w_k^T}{y_k^T s_k} - \frac{w_k^T s_k y_k y_k^T}{y_k^T s_k^2}, \tag{11.3.21}$$

where $s_k = x_k - x_{k-1}$, and $w_k = z_k - B_{k-1}s_k$.

In some cases the updating (11.3.21) 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 [16], $J(x_k)^T J(x_k)$ 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 $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$. Then putting $Q_k = Q(x_k)$ the equations for the Newton direction $d_k = Vq$ can be written

$$(\Sigma^2 + V^T Q_k V)q = -\Sigma r_1, \qquad r_1 = (\, I_n \quad 0\,)\, U^T f(x_k). \tag{11.3.22}$$

We now split the singular values into two groups, $\Sigma = \operatorname{diag}(\Sigma_1, \Sigma_2)$, where $\Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$ are the "large" singular values. If we partition V, q and $\bar{r}$ conformally, then the first r equations in (11.3.22) 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 [16, 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

$$(\nabla r_i(x_k + hv_j) - \nabla r_i(x_k))/h = v_j^T G_i(x_k) + O(h).$$

The vector on the left hand side is the ith row of $(J(x_k + hv_j) - J(x_k))/h$. Multplying with $r_i(x_k)$ and adding we obtain

$$\begin{aligned} r(x_k)^T (J(x_k + hv_j) - J(x_k))/h &= v_j^T \sum_{i=1}^{m} r_i(x_k) G_i(x_k) + O(h) \\ &= v_j^T Q_k + O(h). \end{aligned}$$

Repeating this for all columns in V_2 we obtain an approximation for $V_2^T Q_k$ and we finally form $(V_2^T Q_k)V_2$.

11.3.6 Methods for Separable Problems

A nonlinear least squares problem is said to be **separable** if the parameter vector x can be partitioned as $x^T = (y^T, z^T)$, with the subproblem

$$\min_y \|r(y,z)\|_2, \quad y \in \mathbf{R}^p, \quad z \in \mathbf{R}^q, \tag{11.3.23}$$

easy to solve. In the following we restrict ourself to the particular case when $r(y,z)$ is linear in y i.e.

$$r(y,z) = f(z)y - g(z), \quad f(z) \in \mathbf{R}^{m\times p}. \tag{11.3.24}$$

Then the minimum norm solution to (11.3.22) is $y(z) = f^\dagger(z)g(z)$, where $f^\dagger(z)$ is the pseudoinverse of $f(z)$. The original problem can be written

$$\min_z \|g(z) - f(z)y(z)\|_2 = \min \|(I - P_{f(z)})g(z)\|_2 \tag{11.3.25}$$

where $P_{f(z)} = f(z)f(z)^\dagger$ is the orthogonal projector onto the range of $f(z)$. Algorithms based on (11.3.25) are often called **variable projection algorithms**.

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

$$r(y,z) = H(y)z - h(y), \qquad H(y) \in \mathbf{R}^{m\times q}.$$

Example 11.3.5.

Consider the exponential fitting problem

$$\min_{y,z} \sum_{i=1}^{m} (y_1 e^{z_1 t_i} + y_2 e^{z_2 t_i} - g_i)^2.$$

Here the model is nonlinear only in the parameters z_1 and z_2. Given values of z_1 and z_2 the subproblem (11.3.22) is easily solved.

We here describe a variable projection algorithm due to Kaufman [21, 1975], which uses a Gauss–Newton method applied to the problem (11.3.25). 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

$$\min_{\delta y_k} \big\| f(z_k)\delta y_k - \big(g(z_k) - f(z_k)y_k\big) \big\|_2, \tag{11.3.26}$$

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

$$\min_{d_k} \big\| C(x_{k+1/2})d_k + r(y_{k+1/2}, z_k) \big\|_2, \tag{11.3.27}$$

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 (11.3.27) we have used that by (11.3.23) 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

$$r_z(y_{k+1/2}, z_k) = B(z_k)y_{k+1/2} - g'(z_k), \qquad B(z)y = \left(\frac{\partial F}{\partial z_1}y, \ldots, \frac{\partial F}{\partial z_q}y\right),$$

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 (11.3.4) 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 11.3.4 and 11.3.5.

It can be shown that the Gauss–Newton algorithm applied to (11.3.25) 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 (11.3.26). 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.

11.3.7 Orthogonal Distance Regression

Consider the problem of fitting observations (y_i, t_i), $i = 1, \ldots, m$ to a mathematical model

$$y = g(x, t). \tag{11.3.28}$$

where y and t are scalar variables and $x \in \mathbf{R}^n$ 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 $y_i - g(x, t_i)$. In this section we consider the more general situation, when also the values t_i contain errors.

Figure 11.3.2. *Orthogonal distance fitting.*

Assume that y_i and t_i are subject to errors $\bar{\epsilon}_i$ and $\bar{\delta}_i$ respectively, so that

$$y_i + \bar{\epsilon}_i = g(x, t_i + \bar{\delta}_i), \qquad i = 1, \ldots, m,$$

where $\bar{\epsilon}_i$ and $\bar{\delta}_i$ are independent random variables with zero mean and variance σ^2. Then the parameters x should be chosen so that the sum of squares of the **orthogonal distances** from the observations (y_i, t_i) to the curve in (11.3.28) is minimized, cf. Fig. 11.4.2. Hence the parameters x should be chosen as the solution to

$$\min_{x,\epsilon,\delta} \sum_{i=1}^{m} (\epsilon_i^2 + \delta_i^2), \quad \text{subject to} \quad y_i + \epsilon_i = f(x, t_i + \delta_i), \quad i = 1, \ldots, m.$$

Eliminating ϵ_i using the constraints we arrive at the **orthogonal distance problem**

$$\min_{x,\delta} \sum_{i=1}^{m} \big(f(x, t_i + \delta_i) - y_i\big)^2 + \delta_i^2. \tag{11.3.29}$$

Note that (11.3.29) is a nonlinear least squares problem even if $f(x,t)$ is linear in x.

The problem (11.3.29) has $(m+n)$ unknowns x 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 $r(\delta, x) = (r_1^T(\delta, x), r_2^T(\delta))$ by

$$r_1^T(\delta, x)_i = f(x, t_i + \delta_i) - y_i, \qquad r_2^T(\delta) = \delta_i, \quad i = 1, \ldots, m,$$

the Jacobian matrix for problem (11.3.29) can be written in block form as

$$\tilde{J} = \begin{pmatrix} D_1 & J \\ \underbrace{I_n}_{m} & \underbrace{0}_{n} \end{pmatrix} \begin{matrix} \}m \\ \}m \end{matrix} \in \mathbf{R}^{2m \times (m+n)}, \tag{11.3.30}$$

where

$$D_1 = \operatorname{diag}(d_1, \ldots, d_m), d_i = \left(\frac{\partial f}{\partial t}\right)_{t=t_i},$$

$$J_{ij} = \frac{\partial f(x, t_i + \delta_i)}{\partial x_j}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Note that $\tilde{J}$ is sparse and highly structured. In the Gauss–Newton method we compute corrections $\Delta\delta_k$ and Δx_k to the current approximations which solve the linear least squares problem

$$\min_{\Delta\delta, \Delta x} \left\| \tilde{J} \begin{pmatrix} \Delta\delta \\ \Delta x \end{pmatrix} - \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \right\|_2, \tag{11.3.31}$$

where $\tilde{J}, r_1$, and r_2 are evaluated at the current estimates of δ and x. 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 $G_i = R_{i,i+m}$, $i = 1, 2, \ldots, m$, 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 (11.3.31) now decouples, and Δx_k is determined as the solution to

$$\min_{\Delta x} \|L\Delta x - s_2\|_2.$$

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

$$\Delta\delta_k = D_2^{-1}(s_2 - K\Delta x_k).$$

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

$$\min_{x,\delta} \sum_{i=1}^{m} \Big(\|f(x, t_i + \delta_i) - y_i\|_2^2 + \|\delta_i\|_2^2 \Big).$$

The structure in this more general problem can also be taken advantage of in a similar manner.

Schwetlik and Tiller [32, 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].

11.3.8 Least squares fit 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), $i = 1, \ldots, n$ 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. Algorithms for such a **geometric fitting** are described, e.g., in Gander, Golub, and Strebel [14, 1994].

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

$$p = (a, b_1, b_2, c)^T, \qquad s_i^T = (x_i^2 + y_i^2, x_i, y_i, 1). \tag{11.3.32}$$

The problem can now be formulated as $\min_p \|Sp\|_2^2$, subject to the constraint $\|p\|_2 = 1$. 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

$$z = -\frac{1}{2a}\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \qquad \rho = \frac{1}{2a}\sqrt{\|b\|_2^2 - 4ac}. \tag{11.3.33}$$

We now discuss the algebraic fitting of ellipses. An ellipse in the x-y plane can be represented algebraically by

$$f(x, y, p) = (x \ y)\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + (b_1 \ b_2)\begin{pmatrix} x \\ y \end{pmatrix} + c = 0. \tag{11.3.34}$$

It we define

$$p = (a_{11}, a_{12}, a_{22}, b_1, b_2, c)^T, \qquad s_i^T = (x_i^2, 2x_i y_i, y_i^2, x_i, y_i, 1), \tag{11.3.35}$$

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**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|p\|_2 = 1. \tag{11.3.36}$$

The solution of this constrained problem equals the right singular vector corresponding to the smallest singular value of S.

(b) **Linear constraint**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad p^T b = 1, \tag{11.3.37}$$

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 $SH^T = [s S_2]$ we arrive at the unconstrained problem

$$\min_q \|S_2 q + s\|_2^2, \tag{11.3.38}$$

which is a standard linear least squares problem.

(c) **Quadratic constraint**:

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|Bp\|_2 = 1. \tag{11.3.39}$$

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 $R_{11}p_1 = -R_{12}p_2$.

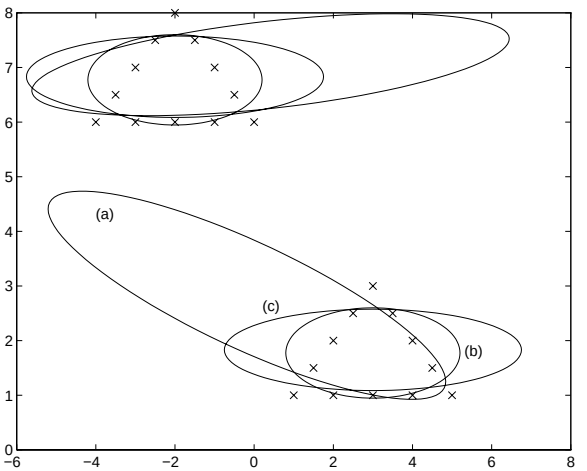


Figure 11.3.3. *Ellipse fits for triangle and shifted triangle data: (a) SVD constraint; (b) Linear constraint $\lambda_1 + \lambda_2 = 1$; (c) Bookstein constraint $\lambda_1^2 + \lambda_2^2 = 1$.*

It should be stressed that the different constraints above can lead to very different solutions, unless the errors in the fit are small. One desirable property of the fitting algorithm is that when the data is translated and rotated the fitted ellipse should be transformed in the same way. It can be seen that to lead to this kind of invariance the constraint must involve only symmetric functions of the eigenvalues of the matrix A.

The disadvantage of the SVD constraint is its non-invariance under translation and rotations. In case of a linear constraint the choice $b^T = (1\ 0\ 1\ 0\ 0\ 0)$, which corresponds to

$$\text{trace}(A) = a_{11} + a_{22} = \lambda_1 + \lambda_2 = 1. \tag{11.3.40}$$

gives the desired invariance. This constraint, attributed to Bookstein,

$$\|A\|_F^2 = a_{11}^2 + 2a_{12}^2 + a_{22}^2 = \lambda_1^2 + \lambda_2^2 = 1. \tag{11.3.41}$$

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 (11.3.34) to the parametric form

$$\begin{pmatrix} x(\theta) \\ y(\theta) \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + Q(\alpha) \begin{pmatrix} a\cos\theta \\ b\sin\theta \end{pmatrix}, \quad Q(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}. \tag{11.3.42}$$

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 (11.3.34) 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

$$\tau = (a_{22} - a_{11})/(2a_{12}), \qquad \tan\alpha = t = \operatorname{sign}(\tau)/\left(|\tau| + \sqrt{1+\tau^2}\right).$$

The elements in Q and Λ are then given by

$$\begin{aligned} \cos\alpha &= 1/\sqrt{1+t^2}, \qquad & \sin\alpha &= t\cos\alpha, \\ \lambda_1 &= a_{11} - ta_{12}, \qquad & \lambda_2 &= a_{22} + ta_{12}. \end{aligned}$$

If we introduce the new coordinates $z = Q\tilde{z} + s$ in the algebraic form (11.3.34) 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

$$s = \begin{pmatrix} x_c \\ y_c \end{pmatrix} = -\frac{1}{2}A^{-1}b = -\frac{1}{2}A^{-1}Q\Lambda^{-1}(Q^T b), \tag{11.3.43}$$

and the axis (a, b) of the ellips are given by

$$\begin{pmatrix} a \\ b \end{pmatrix} = \sqrt{-\tilde{c}}\,\operatorname{diag}\Lambda^{-1/2}, \qquad \tilde{c} = c + \frac{1}{2}b^T s = -\frac{1}{2}\tilde{b}^T\Lambda^{-1}\tilde{b}. \tag{11.3.44}$$

In geometric fitting of data (x_i, y_i), $i = 1, \ldots, m$ to a curve of the form $f(x, y, p) = 0$ the orthogonal distance $d_i(p)$ is first measured from each data point to the curve, where

$$d_i^2(p) = \min_{f(x,y,p)=0} \left((x - x_i)^2 + (y - y_i)^2\right).$$

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 11.3.7.

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 ellips the minimization problem for each point is only one-dimensional.

We will here only consider the orthogonal distance fitting of a circle written in parametric form

$$f(x, y, p) = \begin{pmatrix} x - x_c - r\cos\theta \\ y - y_c - r\sin\theta \end{pmatrix} = 0.$$

where $p = (x_c, y_c, r)^T$. The problem can be written as a nonlinear least squares problem

$$\min_{p,\theta_i} \|r(p, \theta_1, \ldots, \theta_m)\|_2^2, \tag{11.3.45}$$

where r is a vector of length $2m$, and we define

$$r_i = \begin{pmatrix} x_i - x_c + r\cos\theta_i \\ y_i - y_c + r\sin\theta_i \end{pmatrix}$$

We have

$$\frac{\partial r_i}{\partial \theta_i} = r\begin{pmatrix} \sin\theta_i \\ -\cos\theta_i \end{pmatrix}, \qquad \frac{\partial r_i}{\partial r} = -\begin{pmatrix} \cos\theta_i \\ \sin\theta_i \end{pmatrix},$$

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

$$J = \begin{pmatrix} rS & A \\ -rC & B \end{pmatrix}, \quad S = \operatorname{diag}(\sin\theta_i), \quad C = \operatorname{diag}(\cos\theta_i).$$

Here the first block column, which corresponds to the m parameters θ_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 $CA + SB$.

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), $i = 1, \ldots, m$ to the model $y = g(x, t)$, where y and t are scalar variables and $x \in \mathbf{R}^n$ 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), $i = 1, 2, \ldots, m$. The orthogonal distance from (x_i, y_i) to the circle

$$d_i(x_0, y_0, r) = r_i - r, \qquad r_i = \left((x_i - x_0)^2 + (y_i - y_0)^2\right)^{1/2},$$

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 $m < 3$.

11.4 Constrained Linear Optimization

11.4.1 Introduction.

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!

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

$$\begin{aligned} 5x_1 + x_2 &\leq 60 \quad \text{for } M_1, \\ 3x_1 + 4x_2 &\leq 60 \quad \text{for } M_2, \\ 4x_1 + 3x_2 &\leq 60 \quad \text{for } M_3. \end{aligned} \tag{11.4.1}$$

The problem is illustrated geometrically in Fig. 11.3.1. The first of the inequalities (11.4.2) 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.

11.4.2 Optimality for Inequality Constraints.

A linear programming (LP) problem can more generally be stated in the following form:

$$\begin{aligned} &\min_{x \in \mathbf{R}^n} c^T x \\ &\text{subject to} \quad Ax \geq b, \quad x \geq 0. \end{aligned} \tag{11.4.2}$$

Figure 11.4.1. *Geometric illustration of a linear programming problem.*

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 (11.4.2) is the set

$$\mathcal{F} = \{x \in \mathbf{R}^n \mid Ax \geq b\}. \tag{11.4.3}$$

An inequality constraint is said to be **redundant** if its removal does not alter the feasible region.

Obviously, a solution to the LP (11.4.2)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 (11.4.3) has this property, since

$$Az = (1-\alpha)Ax + \alpha Ay \geq (1-\alpha)b + \alpha b = b.$$

when $0 \leq \alpha \leq 1\}$.

The **active set** of the inequality constraints $Ax \geq b$ at a point x is the subset of constraints which are satisfied with equality at x. Hence the constraint $a_i^T x \geq b_i$ 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 $a_i^T x \geq b_i$ is active at x,

then all points $y = x+\alpha p$, $\alpha > 0$, will remain feasible with respect to this constraint if and only if $a_i^T p \geq 0$. It is not difficult to see that the feasible directions p are not affected by the inactive constraints at x. Hence p is a feasible directions at the point x if and only if $a_i^T p \geq 0$ for all active constraints at x.

Given a feasible point x the maximum step α that can be taken along a feasible direction p depends on the inactive constraints. We need to consider the set of inactive constraints i for which $a_i^T p < 0$. For these constraints, which are called decreasing constraints, $a_i^T(x+\alpha p) = a_i^T x + \alpha a_i^T p = b_i$, and thus the constraint i becomes active when

$$\alpha = \alpha_i = \frac{a_i^T x - b_i}{-a_i^T p}.$$

Hence the largest step we can take along p is $\max \alpha_i$ where we maximize over all decreasing constraints.

For 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 minimizing $c^T x$ subject to $Ax \geq b$. (We assume here that the constraints $x \geq 0$ are not present.) Then the following results hold:

(a) *If no points satisfy $Ax \geq b$, the LP has no solution;*

(b) *If there exists a point x^8 satisfying the conditions*

$$Ax^* \geq b, \qquad c = A_A^T \lambda_A^*, \quad \lambda_A^* \geq 0,$$

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 $Ax \geq b$ 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 Lemma**, a classical result published in 1902. For a proof we refer to [18, 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 minimizing $c^T x$ subject to $Ax \geq b$, $A \in \mathbf{R}^{m\times n}$. 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.

11.4.3 Standard Form LP.

It is convenient to adopt the following **standard form** of a linear programming problem:

$$\begin{aligned} &\min_{x\in\mathbf{R}^n} c^T x \\ &\text{subject to} \quad Ax = b, \quad x \geq 0. \end{aligned} \tag{11.4.4}$$

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 $a^T x \leq \beta$ is converted into an equality $a^T x + s = \beta$ by introducing a **slack variable** s subject to $s \geq 0$. A lower bound inequality of the form $a^T x \geq \beta$ can be changed to an upper bound inequality $(-a)^T x \leq -\beta$. 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 $Ax \leq b$, $A \in \mathbf{R}^{m\times n}$, then the matrix in the equivalent standard form will be $(A \quad I_m)$, and the number of variables 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 Fig. 11.4.1) in the five-dimensional space of x. Each side of the pentagon $OABCD$ has an equation of the form $x_i = 0$, $i = 1, \ldots, 5$. At a vertex two of the coordinates are zero, and the rest cannot be negative.

For completeness we note that, although this is seldom used in practice, equality constraints can be converted to inequality constraints. For example, $a_i^T x = b_i$ is equivalent to the two inequality constraints $a_i^T x > b_i$ and $-a_i^T x \geq -b_i$.

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 $x \geq 0$ for which feasible points exist. Then x^* is a minimizer if and only if x^* is a feasible point and

$$c = A^T \pi^* + \eta^*, \qquad \eta^* \geq 0, \qquad \eta_i^* x_i^* = 0, \quad i = 1, \ldots, n. \tag{11.4.5}$$

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

$$\sum_{j=1}^{J} x_{ij} = a_i, \quad i = 1, \ldots, I, \qquad \sum_{i=1}^{I} x_{ij} = b_j, \quad j = 1, \ldots, J.$$

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.

11.4.4 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 11.4.7), 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 $c^T x$. The first phase in the simplex method is to determine an initial basic feasible point (vertex). In some cases an initial vertex can be trivially found (see, e.g., Example 11.4.4 below). A systematic method which can be used in more difficult situations will be described later in Section 11.4.5.

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 (11.4.4). At a vertex $n-m$ variables are zero. We divide the index set $I = \{1, 2, \ldots, m\}$ into two disjoint sets

$$I = B \cup N, \quad B = \{j_1, \ldots, j_n\}, \quad N = \{i_1, \ldots, i_{n-m}\}, \tag{11.4.6}$$

such that N corresponds to the zero variables. We call x_B **basic variables** and x_N **nonbasic variables**. If the vector x and the columns of the matrix A are split in a corresponding way, we can write the system $Ax = b$ as

$$A_B x_B = b - A_N x_N. \tag{11.4.7}$$

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 Fig. 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 Fig. 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 (11.4.10). The optimality criterion is not satisfied. We take $x_r = x_2$ and solve $A_B b_2 = a_2$ to get $b_2 = (1/5)(1, 17, 11)^T$. We find $\theta = 5 \min(12/1, 24/17, 12/11) = 60/11$. Exchanging x_2 and x_5 we go from A to B in Fig. 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 (11.4.7) is nonsingular. (This will always be the case if rank $(A) = n$.) We can then express the basic variables in in terms of the nonbasic

$$x_B = \hat{x}_B - A_B^{-1}A_N x_N, \quad \hat{x}_B = A_B^{-1}b, \tag{11.4.8}$$

where $\hat{x}_B$ is obtained by solving the linear system $A_B\hat{x}_B = b$. If $\hat{x}_B \geq 0$ then $x_N = 0$ corresponds to a basic feasible point (vertex). The vector c is also split in two subvectors c_B and c_N, and using (11.4.8) we have

$$f = c^T x = c_B^T(\hat{x}_B - A_B^{-1}A_N x_N) + c_N^T x_N = c_B^T\hat{x}_B + \hat{c}_N^T x_N,$$

where

$$\hat{c}_N = c_N - A_N^T d, \qquad d = A_B^{-T} c_B. \tag{11.4.9}$$

Here d can be computed by solving the linear system

$$A_B^T d = c_B. \tag{11.4.10}$$

The components of $\hat{c}_N$ are known as the **reduced costs** for the nonbasic variables, and the process of computing them known as **pricing**. If $\hat{c}_N \geq 0$ *then* $x_N = 0$ *corresponds to an optimal point, since* f *cannot decrease when one gives one (or more) nonbasic variables positive values* (negative values are not permitted). Hence if the **optimality criterion** $\hat{c}_N \geq 0$ is satisfied, then the solution $x_B = \hat{x}_B$, $x_N = 0$ is optimal, and we can stop.

If the optimality criterion is *not* satisfied, then there is at least one non-basic variable x_r whose coefficient $\hat{c}_r$ in $\hat{c}_N$ is negative. We now determine the largest positive increment one can give x_r without making any of the basic variables negative, while holding the other non-basic variables equal to zero. Consider equation (11.4.8), and let b_r be the corresponding column of the matrix $A_B^{-1} A_N$. This column can be determined by solving the linear system

$$A_B b_r = a_r, \tag{11.4.11}$$

where a_r is the column in A_N corresponding to x_r. If we take $x_r = \theta > 0$, then $x_B = \hat{x}_B - \theta b_r$, and for any basic variable x_i we have $x_i = \hat{x}_i - \theta b_{ir}$. Hence if $b_{ir} > 0$, then x_i remains positive for $\theta = \theta_i \leq \hat{x}_i / b_{ir}$. The largest θ for which no basic variable becomes negative is given by

$$\theta = \min_i \theta_i, \qquad \theta_i = \begin{cases} \hat{x}_i / b_{ir} & \text{if } b_{ir} > 0; \\ +\infty & \text{if } b_{ir} \leq 0; \end{cases} \tag{11.4.12}$$

If $\theta = +\infty$, the object function is unbounded in the feasible region, and we stop. Otherwise there is at least one basic variable x_l that becomes zero for this value of θ. Such a variable is now interchanged with x_r, so x_r becomes a basic variable and x_l a non-basic variable. (Geometrically this corresponds to going to a neighboring vertex.) Note that the new values of the basic variables can easily be found by updating the old values using $x_i = x_i - \theta b_{ir}$, and $x_r = \theta$.

In case several components of the vector $\hat{c}_N$ are negative we have to specify which variable to choose. The so-called **textbook** strategy chooses r as the index of the most negative component in $\hat{c}_N$. This can be motivated by noting that c_r equals the reduction in the object function $f = c_B^T \hat{x}_B + \hat{c}_N^T x_N$, produced by a unit step along x_r Hence this choice leads to the largest reduction in the objective function assuming a fixed length of the step. A defect of this strategy is that it is not invariant under scalings of the matrix A. A scaling invariant strategy called the **steepest edge strategy** can lead to great gains in efficiency, see Gill, Murray, and Wright [18, 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 Fig. 11.3.2. 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 Fig. 11.3.2) 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 [18, 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

$$P_B A_B = LU \tag{11.4.13}$$

where P_B is a permutation matrix. The initial factorization (11.4.13) 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

$$A_B^T \hat{d} = c_B, \qquad A_B b_r = a_r, \tag{11.4.14}$$

to compute reduced costs and update the basic solution. These systems can be solved cheaply by computing a LU factorization of the matrix A_B 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".)

11.4.5 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 (11.4.4) 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 $x_i \geq 0$, $i = 1, \ldots, 5$, 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

$$\bar{f} = x_1 - x_2 - M x_6 = -2M + (1 + 2M)x_1 - (1 + M)x_2 - M x_3.$$

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 [2, 1992].

11.4.6 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

$$c = A^T y^* + \eta^*, \qquad \eta^* \geq 0, \qquad \eta_i^* x_i^* = 0, \quad i = 1, \ldots, n.$$

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:

$$\begin{aligned} &\max_{y \in \mathbf{R}^m} g = y^T b \\ &\text{subject to} \quad y^T A \leq c^T. \end{aligned} \tag{11.4.15}$$

Here y are the **dual variables**. The initial problem will be called the **primal problem** and x the **primal variables**. If y satisfies the inequality in (11.4.15) 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

$$\begin{aligned} &\max_{y \in \mathbf{R}^m} g = y^T b \\ &\text{subject to} \quad A^T y + z = c, \qquad z \geq 0, \end{aligned} \tag{11.4.16}$$

where z are the dual slack variables. The solution y^* to the dual problem is the Lagrange multiplier for the m linear equality constraints in the primal problem. The primal solution x^* is the Lagrange multiplier for the n linear equality constraints of the standard-form dual problem.

Let x and y be arbitrary feasible vectors for the primal and dual problems, respectively. Then

$$g(y) = y^T b = y^T A x \leq c^T x = f(x). \tag{11.4.17}$$

The nonnegative quantity

$$c^T x - y^T b = x^T z$$

is called the **duality gap**. We will show it is zero if and only if x and y are optimal for the primal and dual.

Theorem 11.4.5.

The optimal values of the primal and dual problem are equal, i.e.,

$$\max g(y) = \min f(x). \tag{11.4.18}$$

The minimum value is obtained at a point $\hat{y}$ which is the solution of the m simultaneous equations

$$\hat{y}^T a_i = c_i, \quad i \in S, \tag{11.4.19}$$

where the set S is the set of integers defined previously.

Proof. By (11.4.17) it holds that

$$\max g(y) \leq \min f(x). \tag{11.4.20}$$

We shall show that $\hat{y}$ as defined by (11.4.19) is a feasible vector. Since $Ax = b$, we may write

$$f(x) = c^T x - \hat{y}^T(Ax - b) = \hat{y}^T b + (c^T - \hat{y}^T A)x.$$

Hence by (11.4.19)

$$f(x) = y^T b + \sum_{j \notin S} (c_j - \hat{y}^T a_j) x_j. \tag{11.4.21}$$

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 11.4.4) that $c_j - \hat{y}^T a_j \geq 0$, $j \notin S$. This together with (11.4.19), shows that $\hat{y}$ is a feasible point for the dual. Moreover, since $\hat{x}_j = 0$, $j \notin S$, then by (11.4.21) $f(\hat{x}) = \hat{y}^T b = g(\hat{y})$. This is consistent with (11.4.20) only if $\max g(y) = g(\hat{y})$. Hence $\max g(y) = \min f(x)$, and the theorem is proved. □

A linear program initially given in the inequality form (11.4.15)–(11.4.17) can be converted to standard form by adding n slack variables. If the simplex method is used to solve this standard problem, each step involves solution of a linear system of sizes $n \times n$. If n is large it may be advantageous to switch instead to the primal problem, which is already in standard form. A simplex step for this problem involves solving linear systems of size $m \times m$, which may be much smaller size!

11.4.7 Interior Point Methods

Interior point methods for optimization problems were introduced by Fiacco and McCormick [1968]. They are characterized by the property that a sequence of approximation strictly inside the feasible region are generated. They work by introducing a nonlinear barrier function which makes the approximations stay away from the boundary. Interest in interior point methods for linear programming did not arise until the work by Karmarkar [20, 1984], since solving a *linear* problem by techniques from *nonlinear* optimization was not believed to be a competitive approach. Currently the best interior point methods are competitive with the Simplex method on most problems and superior for some.

The most promising interior point method for linear programming is the so called primal-dual logarithmic barrier method, which we sketch below. We add a logarithmic barrier to the dual problem (11.4.15) and consider

$$\text{maximize} \quad g = y^T b + \mu \sum_{j=1}^{n} \ln z_j, \tag{11.4.22}$$

$$\text{subject to} \quad y^T A \le c^T.$$

The first order optimality conditions for (11.4.22) can be shown to be

$$\begin{aligned} XZe &= \mu e, \\ Ax &= b \\ A^T y + z &= c, \end{aligned} \tag{11.4.23}$$

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

$$\begin{aligned} Z\delta x + X\delta z &= \mu e - XZe, \\ A\delta x &= b - Ax, \\ A^T\delta y + \delta z &= c - A^T y - z. \end{aligned} \tag{11.4.24}$$

If $Z > 0$ we can solve to get

$$\begin{aligned} AZ^{-1}XA^T\delta y &= -AZ^{-1}(\mu e - XZe) + AZ^{-1}r_D + r_P, \\ \delta z &= -A^T\delta y + r_D, \\ \delta x &= Z^{-1}(\mu e - XZe) - Z^{-1}X\delta z. \end{aligned}$$

A sparse Cholesky factorization of ADA^T, where $D = Z^{-1}X$ is a positive diagonal matrix, is the main computational cost for the solution. Note that we need not worry about feasibility. The idea is to follow a **central path** $y(\mu)$ when $\mu \to 0$.

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 $Ax = b$, $x \ge 0$. 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 $x_1 \ge 0$, $x_2 \ge 0$,

$$x_1 + x_2 \le 2, \qquad x_1 + 2x_2 \le 2.$$

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 $x_i \geq 0$, $i = 1, \ldots, 7$,

$$\begin{aligned} |x_1 + x_2 + x_3 - 4| &\leq 12 \\ 3x_1 + x_2 + 5x_4 &\leq 6 \\ x_1 + x_2 + 3x_3 &\geq 3 \\ |x_1 - x_2 + 5x_7| &\geq 1 \end{aligned}$$

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 11.4.3. (Note that there is no non-negativity condition on y.)

(b) Show that the dual problem of the dual problem is the primal problem.

11.5 Appendix: 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 [12]. Our presentation is also much influenced by Butcher [5, Chapter 1], whose purpose is rather similar to ours, but his discussion is stricter. In these books 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.

11.5.1 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 $(Px_1)^T Q x_2 + (Rx_3)^T S x_4$ 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

$$\sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \cdots \sum_{j_k=1}^{n_k} a^i_{j_1,j_2,\ldots,j_k} x_1^{j_1} x_2^{j_2} \ldots x_k^{j_k}, \quad x_\nu \in X_\nu. \tag{11.5.1}$$

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,

$$\|A(x_1, x_2, \ldots, x_k)\|_\infty \le \|A\|_\infty \|x_1\|_\infty \|x_2\|_\infty \ldots \|x_k\|_\infty,$$

where

$$\|A\|_\infty = \max_{i=1}^{m} \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \cdots \sum_{j_k=1}^{n_k} |a^i_{j_1,j_2,\ldots,j_k}|. \tag{11.5.2}$$

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 $f\colon X \to Y$, 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 $\|f(x') - f(x'')\| \leq \alpha\|x' - x''\|$ 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 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

$$df(x, y) = f_x dx + f_y dy, \quad \forall dx \in \mathbf{R}^k, \quad dy \in \mathbf{R}^l. \tag{11.5.3}$$

If x, y are functions of $s \in \mathbf{R}^n$, then a general version of the *chain rule* reads

$$f'(x(s), y(s)) = f_x x'(s) + f_y y'(s). \tag{11.5.4}$$

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 (11.5.4), $f_x dx + f_y dy = 0$. If $f_y(x_0, y_0)$ is a non-singular matrix, then, by the *implicit function theorem* (see Dieudonné [12, 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 $f''_{ij}(x_0) = f''_{ji}(x_0)$ 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 [12]). 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, $u \in X$, $f : X \to Y$,

$$f(a + u) = f(a) + f'(a)u + \tfrac{1}{2} f''(a)u^2 + \ldots + \frac{1}{k!} f^{(k)}(a)u^k + R_{k+1}, \quad (11.5.5)$$

$$R_{k+1} = \int_0^1 \frac{(1 - t)^k}{k!} f^{(k+1)}(a + ut) dt u^{k+1};$$

it follows that

$$\|R_{k+1}\| \leq \max_{0 \leq t \leq 1} \left\| f^{(k+1)}(a + ut) \right\| \frac{\|u\|^{k+1}}{(k + 1)!}.$$

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 $f^{(k)}$: $X \to L_k(X,Y)$ and the k-linear function $f^{(k)}(x_0)$: $X^k \to Y$. 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.

In the following subsections we show some relevant applications of these notions to numerical mathematics.

11.5.2 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,

$$g'(x_0)v = \frac{g(x_0+\eta v) - g(x_0-\eta v)}{2\eta} + R_T, \quad R_T \approx -\frac{\eta^2 g'''(x_0)v^3}{6}, \tag{11.5.6}$$

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:

$$\min_{\eta} B(\eta) = (3\epsilon)^{2/3}\|g\|\|v\|/(2\xi), \quad \eta\|v\| = (3\epsilon)^{1/3}\xi. \tag{11.5.7}$$

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) $\frac{1}{2}\|g\|/\|g'\|$.[3] 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

$$f''(x_0)v^2 = \frac{f(x_0+\eta v) - 2f(x_0) + f(x_0-\eta v)}{\eta^2} + R_T, \tag{11.5.8}$$

$$R_T \approx -\frac{\eta^2 f^{iv}(x_0)v^4}{12},$$

$$B(\eta) = \|f\|\frac{4\epsilon}{\eta^2} + \frac{\|f^{iv}\|\,\|v\|^4\eta^2}{12},$$

$$\xi = (\|f\|/\|f^{iv}|)^{1/4} \approx (\frac{1}{3}\|f\|/\|f''\|)^{1/2},$$

$$\min_\eta B(\eta) = 2(\epsilon/3)^{1/2}\|f\|\,\|v\|^2/\xi^2 \approx \epsilon^{1/2}\|f''\|\|v\|^2, \quad \eta\|v\| = (48\epsilon)^{1/4}\xi.$$

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 (11.5.6), 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 $f''(x_0)$.

$$f''(x_0)(u,v) = \frac{1}{4}\big(f''(x_0)(u+v)^2 - f''(x_0)(u-v)^2\big) \tag{11.5.9}$$

[3] The factor $\frac{1}{2}$ is a safety factor. So is the factor $\frac{1}{3}$ in the equation for ξ in the group (11.5.8).

11.5.3 Taylor Coefficients for the Solution of a System of Ordinary Differential Equations.

Let y be a function of the real variable t, that satisfies the autonomous differential system $\dot{y} = f(y)$, $f : Y \to Y$.[4] We shall derive recursion formulas for the derivatives of the solution $y(t)$ with respect to t. We use dots for differentiation with respect to t of order less than 3, and we set $\ddot{y} = z$.

By repeated application of the chain rule, the time derivatives of $y(t)$ are expressed in terms of the derivatives of f with respect to the vector y. In the tables below the results are given first in the multilinear mapping notation with primes for differentiation with respect to y (as above). In the last line of the tables, the same vectors are expressed in tensor notation.

$\dot{y}$	$z = \ddot{y}$	$\dot{z} = y^{(3)}$
$f(y)$	$f'(y)\dot{y}$	$f''(y)\dot{y}^2 + f'(y)\ddot{y}$
f	$f'f$	$f''f^2 + (f')^2 f$
f^j	$f^j_k f^k$	$f^j_{kl} f^k f^l + f^j_k f^k_l f^l$

$\ddot{z} = y^{(4)}$
$f'''(y)\dot{y}^3 + 3f''(y)(\ddot{y}, \dot{y}) + f'(y)\dot{z}$
$f'''f^3 + 3f''(f'f, f) + f'f''f^2 + (f')^3 f$
$f^j_{klm} f^k f^l f^m + 3f^j_{km} f^k_l f^l f^m + f^j_k f^k_{lm} f^l f^m + f^j_k f^k_l f^l_m f^m$

Note that, at some places, we have here omitted the *obvious* argument y. We often do so when there is no doubt about the argument.

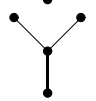
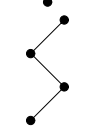
The individual terms on the third and fourth lines of these tables are called **elementary differentials**. The qth order derivative of y is a linear combination of the qth order elementary differentials with *integer coefficients*. They are fundamental in the theory of one-step methods for ordinary differential equations; see Section 13.3.

These matters can easily become rather messy. J. Butcher and others have made the analysis more transparent by employing an one-to-one correspondence between the qth order elementary differentials and a **rooted tree** with q vertices. We denote a rooted tree by $\mathbf{t}$; its order, that is the number of vertices, is denoted $\rho(\mathbf{t})$, and the corresponding elementary differential is denoted $F(\mathbf{t})$. The qth order trees are denoted $\mathbf{t}_{q1}$, $\mathbf{t}_{q2}, \ldots$.

Table 11.5.1 displays up to order 4 the elementary differentials and trees. (analogous to the tree $\mathbf{t_{32}}$). It corresponds to the elementary differential $(f')^3 f$. Study the table, and see Problem 7. Note the monotonic ordering of the labels along the branches, and see how well the tensor notation corresponds to this labeling. $F(\mathbf{t})$ denotes the elementary differential, which corresponds to the tree $\mathbf{t}$, e.g., $F(\mathbf{t_{21}}) = f'f$. A tree $\mathbf{t}$ can be labeled in several ways. A parameter named $\alpha(\mathbf{t})$ equals, in a certain sense, the number of essentially different monotonic labelings of $\mathbf{t}$; $\alpha(\mathbf{t}) = 1$ for all trees in the figure, except for $\alpha(\mathbf{t_{42}}) = 3$. (Pure permutation of the labels

[4] A differential system of equations is said to be autonomous if it does not explicitely contain the independent variable.

Table 11.5.1. *Elementary differentials and the corresponding trees up to order* $\rho(\mathbf{t}) = 4$.

order	$\mathbf{t}$	graph	$F(\mathbf{t})$	
1	$\mathbf{t}_{11}$		f	f^j
2	$\mathbf{t}_{21}$		$f'f$	$f^j_k f^k$
3	$\mathbf{t}_{31}$		$f''f^2$	$f^j_{kl} f^k f^l$
	$\mathbf{t}_{32}$		$(f')^2 f$	$f^j_k f^k_l f^l$
4	$\mathbf{t}_{41}$		$f'''f^3$	$f^j_{klm} f^k f^l f^m$
	$\mathbf{t}_{42}$		$f''(f'f, f)$	$f^j_{km} f^k_l f^l f^m$
	$\mathbf{t}_{43}$		$f'f''f^2$	$f^j_{km} f^k_l f^l f^m$
	$\mathbf{t}_{44}$		$(f')^3 f$	$f^j_k f^k_{lm} f^l f^m$

of leaves on the same branch is not "essential".) The precise definition of $\alpha(\mathbf{t})$ is rather subtle, and we refer to Hairer, Nørsett and Wanner [1993, Ch.2] or Butcher loc.cit. for more detailed information. We give in §13.3.1 a table with $\alpha(\mathbf{t})$ and some other data for $\rho(\mathbf{t}) \leq 5$.

With these notations, the formal Taylor expansion of the solution $y(t)$ around $t = t_0$ reads

$$\begin{aligned} y(t_0 + h) &= y(t_0) + y'(t_0)h + \frac{1}{2!}y''(t_0)h^2 + \frac{1}{3!}y'''(t_0)h^3 + \dots \\ &= y(t_0) + hfy(t_0) + \frac{h^2}{2!}h^2 f'fy(t_0) + \frac{h^3}{3!}(f''f^2 + (f')^2 f)y(t_0) + \dots \\ &= y(t_0) + \Big(hF(\mathbf{t_{11}}) + \frac{h^2}{2!}F(\mathbf{t_{21}}) + \frac{h^3}{3!}\big(F(\mathbf{t_{31}}) + F(\mathbf{t_{32}})\big) + \dots\Big)y(t_0). \end{aligned}$$

More generally, the Taylor expansion becomes,

$$y(t_0 + h) = y(t_0) + \sum_{\mathbf{t}} \frac{h^{\rho(\mathbf{t})}}{\rho(\mathbf{t})!}\alpha(\mathbf{t})F(\mathbf{t})y(t_0), \quad \rho(\mathbf{t}) \geq 1. \tag{11.5.10}$$

This expression is useful for the design and analysis of numerical methods. If you want to use a Taylor expansion for computing the numerical solution of a system, however, you had better use the techniques of *automatic differentiation*, see Section 3.1 and Section 13.3.

The number of elementary differentials for $q = 1 : 10$ are as follows:

$$\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 1 & 2 & 4 & 9 & 20 & 48 & 115 & 286 & 719 \end{array} \tag{11.5.11}$$

Much more about this can be found in Butcher, loc.cit., and Hairer, Nørsett and Wanner, loc.cit..

The formulas for an autonomous system, $\dot{y} = f(y)$, include also the non-autonomous case, i.e. a system of the form $\dot{y} = f(t, y)$, for if we add the trivial equation $\dot{t} = 1$ to the latter system, then we obtain an autonomous system for the vector (t, y), (written as a column). Nevertheless, since the variable t plays a special role, it is sometimes interesting to see the formulas for the non-autonomous system more explicitly. Recall that $f_{ty} = f_{yt}$.

$$\begin{aligned} \dot{y} &= f(t, y) \\ z &= \ddot{y} = df(t, y(t))/dt = f_t + f_y\dot{y} = f_t + f_y f \\ \dot{z} &= (f_t + f_y f)_t + (f_t + f_y f)_y f = f_{tt} + f_y f_t + 2f_{yt} f + f_{yy} f^2 + f_y f_y f, \end{aligned}$$

PROBLEMS

1. (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.

2. 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, $g(x) = e^{\alpha x}$, $g(x) = x^{-k}$.

3. 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 [7]

4. Consider the multilinear operator A defined by (11.5.1), and suppose that $X_\nu = \mathbf{R}^n$, $\forall\nu$. What is $\|A\|$ if a weighted max-norm is used in $\mathbf{R}^n$?

5. Write a program for the approximate computation of the Hessian of a real-valued function, by central differences.

6. Consider an autonomous system, $\dot{y} = f(y)$, $f : \mathbf{R}^s \to \mathbf{R}^s$. Such a system has an infinity of solutions $y(t)$, but we shall see in Section 13.1 that, for given $\tau \in \mathbf{R}$, $\eta \in \mathbf{R}^s$, there is, under very general conditions on the function f, only one solution for which $y = \eta$ for $t = \tau$. Denote this solution by $y(t; \tau, \eta)$. Runge's 2nd order method, introduced in Section 1.3, reads $k_1 = hf(y_n)$, $k_2 = hf\left(y_n + \frac{1}{2}k_1\right)$, $y_{n+1} = y_n + k_2$. Show that

$$y_{n+1} - y(t_n + h; t_n, y_n) = h^3\left(\tfrac{1}{8}f''\dot{y}^2 - \tfrac{1}{6}y'''\right) + O(h^4).$$

(This is called the local error.) Also show that $k_2 - k_1 = \frac{1}{2}h^2\ddot{y} + O(h^3)$. (The vector $k_2 - k_1$ is used for the choice of step size in the algorithm of Section 1.3. See also Section 13.2.)

7. (a) Draw the tree $\mathbf{t}_{44}$, and write down the corresponding elementary differential in multilinear mapping notation and in tensor notation.

(b) Given all trees of order $q-1$, two ways of producing (different) trees of order q are as follows. You can *either* put one more vertex on the first level above the root (and label it with the next character in the alphabet), *or* you can create a new root (labeled j) below the old one and change the other labels. Note that for $q = 3$ and $q = 4$ these operations yield all trees. Find the rules, how the elementary differentials are modified at these tree operations.

(c) For $q = 5$, however, the operations in (b) produce together 8 trees, instead of 9, according to the table in Example 11.5.3. What does the missing tree look like? Find the corresponding elementary differential.

Comment: There is more material about this in Section 13.3.

8. Consider a function $f : X \to X$, $\dim X > 1$. Do expressions like $f''(x_0)f''(x_0)$ and $f''(x_0)f''(x_0)f''(x_0)$ ever make sense?

Notes

The numerical solution of nonlinear equations by the methods of Newton, Brown and Brent is discussed by Moré and Cosnard [26, 1979]. An evaluation of numerical software that solves systems of nonlinear equations is given by Hiebert [19, 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 [30, 1970]. For a modern treatment of continuation methods see Allgower and Georg [1, 1990].

As the name implies the Gauss-Newton method was used already by Gauss [1809].

A standard text-book on numerical methods for unconstrained optimization, nonlinear systems, and nonlinear least squares is Dennis and Schnabel [11, 1983].

A classical reference on linear optimization is Danzig [8, 1965]. For nonlinear optimization the book by Luenberger [24, 1979] is a good introduction. Excellent textbooks on optimization are Gill, Murray and Wright [17, 1981], and Fletcher [13, 1987]. A very useful reference on software packages for large scale optimization is Moré and Wright [27, 1993].

References

[1] E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction.* Springer Series in Computational Mathematics, Vol. 13. Springer-Verlag, Berlin, 1990.

[2] R. E. Bixby. Implementing the simplex method: The initial basis. *ORSA J. Comput.*, 4:267–284, 1992.

[3] P. T. Boggs, R. H. Byrd, and R. B. Schnabel. A stable and efficient algorithm for nonlinear orthogonal regression. *SIAM J. Sci. Statist. Comput.*, 8:1052–1078.

[4] R. P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[5] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods.* Wiley-Interscience, Chichester, 1987.

[6] Y. Censor and S. A. Zenios. *Parallel Optimization. Theory, Algorithms, and Applications.* Oxford University Press, Oxford, 1997.

[7] A. 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.

[8] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, second edition, 1965.

[9] J. E. Dennis, D. M. Gay, and R. E. Welsh. An adaptive nonlinear least-squares algorithm. *ACM. Trans. Math. Software*, 7:348–368, 1981.

[10] J. E. Dennis and J. J. Moré. Quasi-newton methods, motivation and theory. *SIAM Review*, 19:46–89, 1977.

[11] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Classics in Applied Mathematics 16. SIAM, Philadelphia, PA, 1995.

[12] J. Dieudonné. *Foundations of Modern Analysis.* Academic Press, New York, NY, 1961.

[13] R. Fletcher. *Practical Methods of Optimization.* John Wiley, New York, second edition, 1987.

[14] W. Gander, G. H. Golub, and R. Strebel. Least squares fitting of circles and ellipses. *BIT*, 34:4:558–578, 1994.

[15] P. E. Gill, G. H. Golub, W. Murray, and M. Saunders. Methods for modifying matrix factorizations. *Math. Comp*, 28:505–535, 1974.

[16] P. E. Gill and W. Murray. Algorithms for the solution of the nonlinear least squares problem. *SIAM J. Numer. Anal.*, 15:977–992, 1978.

[17] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London, UK, 1981.

[18] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization.* Number 1. Addison-Wesley, London, UK, 1991.

[19] K. L. Hiebert. An evaluation of mathematical software that solves systems of nonlinear equations. *ACM Trans. Math. Software*, 8:5–20, 1982.

[20] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[21] L. Kaufman. Variable projection methods for solving separable nonlinear least squares problems. *BIT*, 15:49–57, 1975.

[22] H. P. Künzi, H. G. Tzschach, and C. A. Zehnder. *Numerical Methods of Mathematical Optimization.* Academic Press, New York, 1971.

[23] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[24] D. G. Luenberger. *Introduction to Dynamic Systems. Theory, Models and Applications.* Wiley, New York, 1979.

[25] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11:431–441, 1963.

[26] J. J. Moré and M. Y. Cosnard. Numerical solution of nonlinear equations. *ACM. Trans. Math. Software*, 5:64–85, 1979.

[27] J. J. Moré and S. J. Wright. *Optimization Software Guide.* SIAM, Philadelphia, PA, 1993.

[28] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[29] A. M. Ostrowski. *Solution of Equations in Euclidian and Banach Spaces.* Academic Press, New York, 1973.

[30] 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.

[31] P. Rabinowitz. *Numerical Methods for Non-Linear Algebraic Equations.* Gordon and Breach, London, UK, 1970.

[32] H. Schwetlick and V. Tiller. Numerical methods for estimating parameters in nonlinear models with errors in the variables. *Technometrics*, 27:17–24, 1985.

[33] S. J. Wright. *Primal-Dual Interior Point Methods.* SIAM, Philadelphia, PA, 1997.

Appendix. Guide to Literature in Linear Algebra

The literature on linear algbera is very extensive. For a theoretical treatise a classical source is Gantmacher [18, 1959]. Several nonstandard topics are covered in Lancaster and Tismenetsky [33, 1985] and in two excellent volumes by Horn and Johnson [29, 1985] and [30, 1991]. A very complete and useful book on and perturbation theory and related topics is Stewart and Sun [49, 1990]. Analytical aspects are emphasized in Lax [35, 1997].

An interesting survey of classical numerical methods in linear algebra can be found in Faddeev and Faddeeva [15, 1963], although many of the methods treated are now dated. A compact, lucid and still modern presentation is given by Householder [31, 1964]. Bellman [6, 1970] is an original and readable complementary text.

An excellent textbook on matrix computation are Stewart [46, 1973]. The recent book [47, 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 [23, 1996], which has become a standard reference. The book by Higham [28, 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 [7, 1996].

Two classic texts on iterative methods for linear systems are Varga [54, 1962] and Young [59, 1971]. The more recent book by Axelsson [2, 1994], also covers conjugate gradient methods. Barret et al. [5, 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 [44, 1996].

A still unsurpassed text on computational methods for the eigenvalue problem is Wilkinson [57, 1965]. Wilkinson and Reinsch [58, 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 [?, 1980]. Large scale eigenvalue problems are treated by Saad [43, 1992]. For an introduction to the implementation of algorithms for vector and parallel computers, see also Dongarra et al. [13, 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. [12, 1979] and Smith et al. [45, 1976]. Direct methods for sparse symmetric positive definite systems are covered in George and Liu [20, 1981], while a more general treatise is given by Duff et al. [14, 1986].

LAPACK95 is a Fortran 95 interface to the Fortran 77 LAPACK library documented in [1, 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 [4, 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/.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, editors. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, 1999.

[2] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge, UK, 1994.

[3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. A. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, PA, 2000.

[4] V. A. Barker, L. S. Blackford, J. J. Dongarra, S. Hammarling J. Du Croz, M. Marinova, J. Was'niewski, and P. Yalamov, editors. *LAPACK 95 Users' Guide*. SIAM, Philadelphia, PA, 2001.

[5] R. Barret, M. W. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, editors. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 1993.

[6] R. Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, New York, 1970.

[7] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA, 1996.

[8] C. Brezinski. *Projection Methods for System of Equations*. Elsevier, Amsterdam, 1997.

[9] F. Chatelin. *Eigenvalues of Matrices*. Wiley, Chichester, 1993.

[10] P. G. Ciarlet. *Introduction to Numerical Linear Algebra and Optimization*. Cambridge University Press, Cambridge, UK, 1989.

[11] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[12] J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. SIAM, Philadelphia, PA, 1979.

[13] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. Van der Vorst. *Numerical Linear Algebra for High-Performance Computers*. SIAM, Philadelphia, PA, 1998.

[14] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, London, 1986.

[15] D. K. Faddeev and V. N. Faddeeva. *Computational Methods of Linear Algebra*. W. H. Freeman, San Francisco, CA, 1963.

[16] L. Fox. *Introduction to Numerical Linear Algebra.* Clarendon Press, Oxford, 1964.

[17] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1992.

[18] F. R. Gantmacher. *The Theory of Matrices. Vols. I and II.* Chelsea Publishing Co, New York, 1959.

[19] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and G. W. Stewart. *Matrix Eigensystems Routines: EISPACK Guide Extension.* Springer-Verlag, New York, 1977.

[20] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[21] G. H. Golub and D. O'Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Review*, 31:50–102, 1989.

[22] G. H. Golub and H. 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.

[23] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[24] A. Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, Philadelphia, PA, 1997.

[25] R. T. Gregory and L. K. Karney. *A Collection of Matrices for Testing Computational Algorithms.* Wiley, New York, 1969.

[26] L. A. Hageman and D. M. Young. *Applied Iterative Methods.* Academic Press, New York, 1981.

[27] W. W. Hager. *Applied Numerical Linear Algebra.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[28] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA, 2002.

[29] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, UK, 1985.

[30] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, Cambridge, UK, 1991.

[31] A. S. Householder. *The Theory of Matrices in Numerical Analysis.* Dover, New York, 1964.

[32] A. Jennings and J. J. McKeown. *Matrix Computation.* Wiley, New York, second edition, 1992.

[33] P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, New York, 1985.

[34] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1974. Reprinted by SIAM, Philadelphia, PA, 1995.

[35] P. D. Lax. *Linear Algebra.* Wiley, New York, 1997.

[36] S. J. Leon. *Linear Algebra with Applications.* Macmillan, New York, fourth edition, 1994.

[37] M. Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities.* Allyn and Bacon, Boston, MA, 1964.

[38] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra.* SIAM, Philadelphia, PA, 2000.

[39] B. Noble and J. W. Daniel. *Applied Linear Algebra.* Prentice-Hall, Englewood Cliffs, second edition, 1977.

[40] J. M. Ortega. *Matrix Theory. A Second Course.* Plenum Publ., New York, 1987.

[41] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Classsics in Applied Mathematics 20. SIAM, Philadelphia, PA, 1998.

[42] J. R. Rice. *Matrix Computation and Mathematical Software.* Mc Graw-Hill, New York, 1981.

[43] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Halstead Press, New York, 1992.

[44] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, MA, 1996.

[45] 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.

[46] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, New York, 1973.

[47] G. W. Stewart. *Matrix Algorithms Volume I: Basic Decompositions.* SIAM, Philadelphia, PA, 1998.

[48] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems.* SIAM, Philadelphia, PA, 2001.

[49] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, Boston, MA, 1990.

[50] G. Strang. *Linear Algebra and Its Applications.* Academic Press, New York, third edition, 1988.

[51] L. N. Trefethen. Pseudospectra of linear operators. *SIAM Review*, 39:383–406, 1997.

[52] L. N. Trefethen and III D. Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[53] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem; Computational Aspects and Analysis.* SIAM, Philadelphia, PA, 1991.

[54] R. S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, 1962.

[55] D. S. Watkins. *Fundamentals of Matrix Computation.* Wiley-InterScience, New York, 2002.

[56] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. ACM*, 8:281–330, 1961.

[57] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[58] J. H. Wilkinson and C. Reinsch, editors. *Handbook for Automatic Computation. Vol. II, Linear Algebra.* Springer-Verlag, New York, 1971.

[59] D. M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.

Index

Chapter 13

Ordinary Differential Equations

13.1 Initial Value Problems for ODEs. Theoretical Background

13.1.1 Introduction

To start with, we shall study the following problem: given a function $f(t, y)$, find a function $y(t)$ which for $a \leq t \leq b$ is an approximate solution to the **initial value problem** for the ordinary differential equation or, with an established abbreviation, the ODE,

$$\frac{dy}{dt} = f(t, y), \qquad y(a) = c. \tag{13.1.1}$$

In mathematics courses, one learns how to determine exact solutions to this problem for certain special functions f. An important special case is when f is a linear function of y. However, for most differential equations, one must be content with approximate solutions.

Many problems in science and technology lead to differential equations. Often, the variable t means time, and the differential equations expresses the rule or law of nature which governs the change in the system being studied. In order to simplify the language in our discussions, we consider t as the time, and we use terms like time step, velocity etc. However, t can be a spatial coordinate in some applications.

As a rule, one has a **system of first-order differential equations** and initial conditions for several unknown functions (often more than ten), say $y_1, y_2, \ldots, y_s$, where

$$\frac{dy_i}{dt} = f_i(t, y_1, \ldots, y_s), \qquad y_i(a) = c_i, \qquad i = 1, 2, \ldots, s. \tag{13.1.2}$$

It is convenient to write such a system in vector form

$$\frac{dy}{dt} = f(t, y), \qquad y(a) = c,$$

where now

$$y = (y_1, \ldots, y_s)^T, \quad c = (c_1, \ldots, c_s)^T, \quad f = (f_1, \ldots, f_s)^T$$

are vectors. When the vector form is used, it is just as easy to describe numerical methods for systems as it is for just one single equation.

Often it is convenient to assume that the system is given in **autonomous form**—

$$\frac{dy}{dt} = f(y), \qquad y(a) = c \tag{13.1.3}$$

—i.e., f does not depend explicitly on t. A non-autonomous system is easily transformed into autonomous form by the addition of the trivial extra equation,

$$\frac{dy_{s+1}}{dt} = 1, \qquad y_{s+1}(a) = a,$$

which has the solution $y_{s+1} = t$. (It is sometimes more convenient to call the extra variable y_0 instead of y_{s+1}.)

We shall, with one exception, only consider numerical methods that produce identical results (in exact arithmetic) for a non-autonomous system and for the equivalent autonomous system. The use of the autonomous form in the analysis and in the description of numerical methods is usually no restriction. Wherever it is necessary or convenient, we shall return to the non-autonomous formulation.

Also higher-order differential equations can be rewritten as a system of first-order equations.

Example 13.1.1.

The differential equation

$$\frac{d^3y}{dt^3} = g\Big(t, y, \frac{dy}{dt}, \frac{d^2y}{dt^2}\Big)$$

with initial conditions

$$y(0) = c_1, \quad y'(0) = c_2, \quad y''(0) = c_3,$$

is by the substitution

$$y_1 = y, \qquad y_2 = y', \qquad y_3 = y'',$$

transformed into the system

$$y' = \begin{pmatrix} y_2 \\ y_3 \\ g(t, y_1, y_2, y_3) \end{pmatrix}, \qquad c = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}.$$

Most programs for initial value problems for ODEs are written for non-autonomous first order systems. So, this way of rewriting a higher order system is of practical

importance. The transformation to autonomous form mentioned above is, however, rarely needed in practice, but it gives a conceptual simplification in the description and the discussion of numerical methods.

If t denotes *time*, then the differential equation (13.1.3), determines the velocity vector of a particle as a function of time and the position vector y. Thus, the differential equation determines a **velocity field**, and its solution describes a *motion* in this field along an **orbit** in $\mathbf{R}^s$. For a non-autonomous system, the velocity field changes with time. You need time as an extra coordinate for visualizing all the velocity fields, just like the standard device mentioned above for making a non-autonomous system autonomous.

Example 13.1.2.

The velocity field of the two-dimensional system

$$\begin{aligned} y_1' &= -y_1 - y_2, \qquad y_1(0) = c_1, \\ y_2' &= y_1 - y_2, \qquad y_2(0) = c_2, \end{aligned}$$

(where f is independent of t) is shown in Fig. 13.1.1. The solution of the equation describes the motion of a particle in that field. For various initial conditions, we get a whole *family of solution curves*. Several such curves are shown, for

$$(c_1, c_2) = (L, 0), (L, L/2), (L, L), (L/2, L), (0, L).$$

This interpretation is directly generalizable to three dimensions, and these geometric ideas are also suggestive for systems of more than three dimensions.

Figure 13.1.1. *Velocity field of the two-dimensional system.*

It is useful to bear in mind the simple observation that every point of an orbit gives the initial value for the rest of the orbit. "Today is the first day of the rest of

your life." Also note that the origin of time is arbitrary for an *autonomous system*: if $y(t)$ is one solution, then $y(t+k)$ is also a solution for any constant k.

It seems plausible that the motion in a given velocity field is uniquely determined by its initial position, provided that the velocity field is sufficiently well-behaved. This statement will be made more precise below (see Theorem 13.1.1). In other words: *under very general conditions, the initial value problem defined by* (13.1.3) *has exactly one solution.* The picture also suggests that if one chooses sufficiently small step size, the rule,

$$\text{Displacement} = \text{Step size} \times \text{Mean velocity over a time step},$$

can be used for a step-by-step construction of an approximate solution. In fact, this is the basic idea of most methods for the numerical integration of ODEs.

More or less sophisticated constructions of the "mean velocity" over time step yield different methods, sometimes presented under the name of *dynamic simulation*. The simplest one is **Euler's method** that was described already in Chapter 1. We assume that the reader has a clear idea of this method. The methods we shall treat are such that one proceeds step by step to the times $t_1, t_2, \ldots$, and computes approximate values, $y_1, y_2, \ldots$, to $y(t_1), y(t_2), \ldots$. We shall distinguish two classes of methods:

a) **One-step methods**, where y_n is the only input data to the step in which y_{n+1} is to be computed. Values of the function $f(y)$ (and perhaps also some of its total time derivatives) are computed at a few points close to the orbit. They are used in the construction of an estimate of y_{n+1}, usually much more accurate than Euler's method would have given. Or, the accuracy requested is as a rule obtained with much longer steps and less work than Euler's method needs. We shall see, however, that there are exceptions from this rule, due to the possibility of numerical instability.

b) **Multistep methods**, where longer time steps are made possible by the use of several values from the past: $y_n, y_{n-1}, \ldots, y_{n-k+1}$. This requires some special arrangement at the start. One can e.g. use a one step method to provide k initial values. Another possibility is to use the procedure for the automatic variation of k and step size that is contained in most codes for multistep methods. The computations can therefore start with $k = 1$ and a very small step size. Then these quantities are gradually increased. Numerical stability is an important issue also for these methods.

It is important to distinguish between **global and local error**. The **global error** at $t = t_{n+1}$ is $y_{n+1} - y(t_{n+1})$, where $y(t)$ is the exact solution to the initial-value problem and y_{n+1} the computed value. The **local error** at at t_{n+1} is the difference between y_{n+1} and the value at t_{n+1} of that solution of the differential equation which passes through the point (t_n, y_n). In other words, the local errors are the jumps in the staircase curve in Fig. 13.1.3.

The distinction between the global and local error is made already at the computation of integrals (Chapter 8), which is, of course, a special case of the differential equation problem, where f is independent of y. There the global error

is simply the sum of the local errors. We saw, e.g., in Sec. 8.2.1, that the global error of the trapezoidal rule is $O(h^2)$, while its local error is $O(h^3)$, although this terminology was not used there. The reason for the difference in order of magnitude of the two errors is that the number of local errors which are added is inversely proportional to h.

For differential equations, the error propagation mechanism is, as we shall see, more complicated; but even here it holds that *the difference between the exponents in the asymptotic dependence on h of the local and global errors is equal to 1*, for first order differential systems.

We say that the **order of accuracy** is equal to p, if the *global error* is approximately proportional to h^p. For example, Euler's method is 1st order accurate. Although this definition is based on the application of a method with constant step size, it makes sense to use the same value also for computations with variable step size.

An example of a one-step method was given in Sec. 1.4, namely Runge's 2nd order method, with two function evaluations per step:

$$k_1 = hf(t_n, y_n); \quad k_2 = hf(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1); \quad y_{n+1} = y_n + k_2. \tag{13.1.4}$$

The step size is chosen so that $k_2 - k_1 \approx 3\,TOL$, where TOL is a tolerance chosen by the user. The difference $k_2 - k_1$ is approximately proportional to h^2. For details, see Sec. 1.4 and Sec. 13.6.

We postpone the discussion of multistep methods to Sec. 13.4, where there is a more systematic discussion of such methods. One-step methods, in particular Runge-Kutta methods will be treated in Sec. 13.3. Other methods, in particular extrapolation methods, are discussed in Sec. 13.5. Special methods for second order equations and methods for boundary and eigenvalue problems are then presented in Sec. 13.6.

In the other sections, ideas, concepts and results will be presented, which are relatively independent of the particular numerical methods, although references are made to Euler's method, Runge's 2nd order method and the analogous implicit methods for the illumination of the general theory. In Sec. 13.1 we treat existence and uniqueness for intial value problems, error propagation an logaritmic norms with applications. The headlines of Sec. 13.2 are control of step size, local time scale and scale functions, introductions to numerical stability, and finally general questions related to implicit methods, with applications to stiff and differential algebraic systems. In Sec. 13.7 some useful ideas from the qualitative theory of differential equations are collected, together with more advanced theory and applications of logarithmic norms. Finally, Sec. 13.8 is devoted to systems of difference equations, matrix power boundedness and some other topics like algorithms and graphical techniques for the investigation of numerical stability.

13.1.2 Existence and Uniqueness for Initial Value Problems

We shall consider initial value problems for the autonomous system

$$dy/dt = f(y), \qquad y(a) = c, \tag{13.1.5}$$

where $f : \mathbf{R}^s \to \mathbf{R}^s$. In this subsection we shall use a single bar $|\cdot|$ to denote a norm in $\mathbf{R}^s$ or the absolute value of a number, while a double bar $\|\cdot\|$ is used for the max-norm in a Banach space $\mathbf{B}$ of continuous vectorvalued functions over an interval $I = [a-d, a+d]$, i.e.,

$$\|u\| = \max_{t\in I} |u(t)|.$$

These notations are also used for the corresponding operator norms. Let $D \subseteq \mathbf{R}^s$ be a closed region. We recall, see Sec. 12.2, that f satisfies a Lipschitz condition in D, with the Lipschitz constant L, if

$$|f(y) - f(z)| \le L|y - z|, \quad \forall y, z \in D. \tag{13.1.6}$$

By Lemma 11.2.2, $\max |f'(y)|$, $y \in D$, is a Lipschitz constant, if f is differentiable and D is convex. A point, where a local Lipschitz condition is not satisfied is called a **singular point** of the system (13.1.5).

Theorem 13.1.1.

If f satisfies a Lipschitz condition in the whole of $\mathbf{R}^s$, then the initial value problem (13.1.5) has precisely one solution for each initial vector c. The solution has a continuous first derivative for all t.

If the Lipschitz condition holds in a subset D of $\mathbf{R}^s$ only, then existence and uniqueness holds as long as the orbit stays in D.

Proof. We shall sketch a proof of this fundamental theorem, when $D = \mathbf{R}^s$, based on an iterative construction named after Picard. We define an operator F (usually nonlinear) that maps the Banach space $\mathbf{B}$ into itself:

$$F(y)(t) = c + \int_a^t f(y(x))\,dx.$$

Note that the equation $y = F(y)$ is equivalent to the initial value problem (13.1.5) on some interval $[a-d, a+d]$, and consider the iteration, $y_0 = c$ (for example),

$$y_{n+1} = F(y_n).$$

For any pair y, z of elements in $\mathbf{B}$, we have,

$$\begin{aligned}\|F(y) - F(z)\| &\le \int_a^{a+d} |f\big(y(t)\big) - f\big(z(t)\big)| \cdot |dt| \\ &\le \int_a^{a+d} L|y(t) - z(t)| \cdot |dt| \le Ld\|y - z\|.\end{aligned}$$

It follows that Ld is a Lipschitz constant of the operator F. If $d < 1/L$, F is a contraction, and it follows from the Contraction Mapping (Theorem 11.2.1) that the equation $y = F(y)$ has a unique solution. For the initial value problem (13.1.5) it follows that there exists precisely one solution, as long as $|t-a| \le d$. This solution

can then be continued to any time by a step by step procedure, for $a + d$ can be chosen as a new starting time and substituted for a in the proof. In this way we extend the solution to $a + 2d$, then to $a + 3d$, $a + 4d$ etc. (or backwards to $a - 2d$, $a - 3d$, etc.). □

Note that this proof is based on two ideas of great importance to numerical analysis: *iteration* and the *step-by-step construction*. (There is an alternative proof that avoids the step-by-step construction, see, e.g., Coddington and Levinson, [2, 1955, p. 12]). A few points to note are:

A. For the *existence* of a solution, it is *sufficient that f is continuous*, (the existence theorem of Cauchy and Peano, see, e.g., Coddington and Levinson [2, 1955, p.6])). That *continuity is not sufficient for uniqueness* can be seen by the following simple initial value problem,

$$y' = 2|y|^{1/2}, \qquad y(0) = 0,$$

which has an infinity of solutions for $t > 0$, namely $y(t) = 0$, or, for any non-negative number k,

$$y(t) = \begin{cases} 0, & \text{if } t \le k; \\ (t-k)^2, & \text{otherwise.} \end{cases}$$

B. The theorem is *extended to non-autonomous systems* by the usual device for making a non-autonomous system autonomous (see Sec. 13.1.1).

C. If the Lipschitz condition holds only in a subset D, then the ideas of the proof can be extended to guarantee existence and uniqueness, *as long as the orbit stays in D*. Let M be an upper bound of $|f(y)|$ in D, and let r be the shortest distance from c to the boundary of D. Since

$$|y(t) - c| = |\int_a^t f\big(y(x)\big)dx| \le M|t - a|,$$

we see that there will be no trouble as long as $|t - a| < r/M$, at least. (This is usually a pessimistic underestimate.) On the other hand, the example

$$y' = y^2, \qquad y(0) = c > 0,$$

which has the solution $y(t) = c/(1 - ct)$, shows that the solution can cease to exist for a finite t (namely for $t = 1/c$), even if $f(y)$ is differentiable for all y. Since $f'(y) = 2y$, the Lipschitz condition is guaranteed only as long as $2y < L$. In this example, such a condition cannot hold forever, no matter how large L has been chosen.

D. On the other side: the solution of a *linear* non-autonomous system, where the data (i.e. the coefficient matrix and the right hand side) are *analytic* functions in some domain of the complex plane, cannot have other singular points than the data, in the sense of complex analysis.

E. Isolated *jump discontinuities* in the function f offer no difficulties, if the problem after a discontinuity can be considered as a new initial value problem that satisfies a Lipschitz condition. For example, *in a non-autonomous problem of the form*

$$y' = f(y) + r(t),$$

existence an uniqueness holds, even if the driving function $r(t)$ is only piecewise continuous. In this case $y'(t)$ is discontinuous, only when $r(t)$ is so, hence $y(t)$ is continuous. There exist, however, more nasty discontinuities, where existence and uniqueness are not obvious, see Problem 3.

F. A point y^* where $f(y^*) = 0$ is called a **critical point** of the autonomous system. (It is usually not a singular point.) If $y(t_1) = y^*$ at some time t_1, the theorem tells that $y(t) = y^*$ is the only solution for all t, forwards as well as backwards. It follows that a solution that does not start at y^* cannot reach y^* exactly in finite time, but it can converge very fast towards y^*.

Note that this does not hold for a non-autonomous system, at a point where $f(t_1, y(t_1)) = 0$, as is shown by the simple example $y' = t$, $y(0) = 0$, for which $y(t) = \frac{1}{2}t^2 \neq 0$ when $t \neq 0$. *For a non-autonomous system $y' = f(t, y)$, a critical point is instead defined as a point y^*, such that $f(t, y^*) = 0$, $\forall t \geq a$.* Then it is true that $y(t) = y^*, \forall t \geq a$, if $y(a) = y^*$.

13.1.3 Variational Equations and Error Propagation

We first discuss the propagation of disturbances (for example numerical errors) in an ODE system. It is a useful model for the error propagation in the application of one step methods, i.e. if y_n is the only input data to the step, where y_{n+1} is computed.

Figure 13.1.2. *Two families of solution curves $y(t; c)$.*

The solution of the initial-value problem, (13.1.3), can be considered as a function $y(t; c)$, where c is the vector of initial conditions. Here again, one can

visualize a *family of solution curves*, this time in the (t, y)-space, one curve for each initial value, $y(a; c) = c$. For the case $s = 1$, the family of solutions can, for example, look one of the two set of curves in Fig. 13.1.2a,b. The dependence of the solution $y(t; c)$ on c is often of great interest both for the technical and scientific context it appears in and for the numerical treatment of the differential equations.

A disturbance in the initial condition—e.g., a round-off error in the value of c—means that $y(t)$ is forced to follow "another track" in the family of solutions. Later, there is a small disturbance at each step,—truncation error and/or rounding error—which produces a similar transition to "another track" in the family of solution curves. In Fig. 13.1.3, we give a greatly exaggerated view of what normally happens. One can compare the above process of error propagation to an *interest process*; in each stage there is "interest" on previously committed errors. At the same time, a new "error capital" (local error) is put in. In Fig. 13.1.3, the local errors are the jumps in the staircase curve. The "interest rate" can, however, be negative (see Fig. 13.1.2b), an advantage in this context. If the curves in the family

Figure 13.1.3. *Propagation of truncation error.*

of solutions depart from each other quickly, then the initial value problem is ill-conditioned; otherwise, it is well-conditioned. (It is possible that a computational method which is not a one-step method can introduce other characteristics in the error propagation mechanism which are *not* inherent in the differential equation itself. *So our discussion in this section are valid, only if the method has adequate stability properties* , for the step size sequence chosen. We shall make this assumption more clear later, e.g in Sec. 13.2.2 and Sec. 13.4. For the two methods mentioned so far, the discussion is relevant (for example) as long as $\|hf'(y)\|^2 \ll 1$.

We can look at the error propagation more quantitatively, to begin with in *the scalar case*. Consider the function $u = \partial y(t; c)/\partial c$. It satisfies a linear differential equation, the **linearized variational equation**

$$\frac{\partial u}{\partial t} = J(t)\, u, \qquad J(t) = \Big(\frac{\partial f}{\partial y}\Big)_{y=y(t;c)}, \tag{13.1.7}$$

since

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial t}\Big(\frac{\partial y}{\partial c}\Big) = \frac{\partial}{\partial c}\Big(\frac{\partial y}{\partial t}\Big) = \frac{\partial}{\partial c}\big(f(t, y(t;c))\big) = \frac{\partial f}{\partial y}\frac{\partial y}{\partial c}.$$

Note that the variational equation is usually non-autonomous, even if the underlying ODE is autonomous. We can derive many results from the above, since

$$y(t, c+\delta c) - y(t;c) \approx \frac{\partial y}{\partial c}\delta c = u(t;c)\delta c.$$

We rewrite Eqn. (13.1.7) in the form, $\dfrac{\partial \ln|u|}{\partial t} = J(t)$.

Proposition 13.1.2.

Closely lying curves in the family of solutions approach each other, as t increases, if $\partial f/\partial y < 0$ and depart from each other if $\partial f/\partial y > 0$.

$\partial f/\partial y$ corresponds to the "rate of interest" mentioned previously. In the following *we assume that* $\partial f/\partial y < \mu^*$ for all y in some interval D, that contains the range of $y(t;c)$, $(a < t < b)$. Hence $\partial \ln|u|/\partial t \le \mu^*$. The following propositions are obtained by the integration of this inequality.

Proposition 13.1.3.

For $u(t) = \partial y/\partial c$ it holds, even if μ^* is negative, that

$$|u(t)| \le |u(a)|e^{\mu^*(t-a)}, \qquad a \le t \le b.$$

Proposition 13.1.4.

Let $y(t_n)$ be perturbed by a quantity ϵ_n. The effect of this perturbation on $y(t)$, $t > t_n$, will not exceed

$$|\epsilon_n|e^{\mu^*(t-t_n)} \tag{13.1.8}$$

as long as this bound guarantees that the perturbed solution curve remains in D.

Various bounds for the global error can be obtained by adding such local contributions. Assume, e.g., that there is no initial error at $t = t_0$, that the sequence $\{t_n\}$ is increasing, and that the local *error per unit of time* is less than ϵ, i.e.,

$$|\epsilon_n| \le \epsilon(t_{n+1} - t_n).$$

Substituting in (13.1.8) and summing all contributions for $t_{n+1} \le t$, an approximate bound for the global error at the time t can thus be obtained:

$$\epsilon \sum_{t_{n+1} \le t} (t_{n+1} - t_n)e^{\mu^*(t-t_n)} \approx \epsilon \int_{t_0}^{t} e^{\mu^*(t-x)}dx,$$

hence

$$|\text{Global Error}| \le \begin{cases} \frac{\epsilon}{\mu^*}(e^{\mu^*(t-t_0)} - 1), & \text{if } \mu^* \ne 0;. \\ \epsilon(t - t_0), & \text{if } \mu^* = 0; \end{cases} \tag{13.1.9}$$

Figure 13.1.4. *The global error bound, divided by ϵ, versus $t - t_0$ for $\mu^* = \pm 1,\ \pm 0.5,\ 0$.*

Note that, if $\mu^* < 0$, the error is bounded by $\epsilon/|\mu^*|$, for all $t > t_0$. The global error bound, divided by ϵ, is shown in Fig. 13.1.4 for $\mu^* = \pm 1,\ \pm 0.5,\ 0$.

We shall see that Fig. 13.1.4 and the inequalities of (13.1.9), with a different interpretation, are typical for the error propagation under much more general and realistic assumptions than those made here. More general versions of Propositions 2 and 3 will be given below.

Now the concept of variational equation will be generalized to *systems* of ODEs. Now $u(t;c)$ and $J(t;c)$ become matrix-valued. We therefore write U instead of u. J and U are Jacobian matrices,to be evaluated at $y = y(t;c)$,

$$J = \frac{\partial f}{\partial y}, \qquad U = \frac{\partial y}{\partial c}. \tag{13.1.10}$$

We first discuss linear systems with variable coefficients in general, and drop the parameter c in $U(t;c), J(t;c)$. It is important to distinguish c from the initial value of this *linear* system.

(13.1.7) becomes a *matrix differential equation,*

$$\frac{dU}{dt} = J(t)U, \qquad U(a) = I. \tag{13.1.11}$$

A matrix differential equation is simply a collection of vector differential equations. The jth column of U is $u_j = \partial y/\partial c_j$. Then $u = u_j(t)$ is the solution of the vector differential equation

$$\frac{du}{dt} = J(t)u, \tag{13.1.12}$$

with the initial condition $u(a) = e_j$. The solution of (13.1.12) with a general initial condition at $t = a$ reads,

$$u(t; u(a)) = U(t)\,u(a). \tag{13.1.13}$$

More generally, the solution with a condition at $t = x$ reads,

$$u(t) = U(t)(U(x))^{-1}u(x). \tag{13.1.14}$$

$U(t)$ is called a **fundamental matrix** solution for (13.1.12). We summarize and extend this in a theorem.

Theorem 13.1.5.

The solution at time t of a homogeneous linear ODE system with variable coefficients is a linear function of its initial vector. For the system in (13.1.11) this function is represented by the fundamental matrix $U(t)$ defined by (13.1.11).

The solution of the inhomogeneous problem,

$$\frac{du}{dt} = J(t)u + r(t),$$

reads,

$$u(t; u(a)) = U(t)u(a) + \int_a^t U(t)(U(x))^{-1}r(x)dx. \tag{13.1.15}$$

For a fixed t, $u(t)$ is thus an affine function of $u(a)$.

Proof.

The verification of (13.1.15) is left as an exercise (Problem 7). □

If J does not depend on t,

$$U(t) = e^{tJ}, \qquad U(t)(U(x))^{-1} = U(t-x). \tag{13.1.16}$$

The matrix exponential is defined in see Sec. 10.2.3. More generally, the fundamental matrix can be expressed in terms of matrix exponentials, if $J(t)$ commutes with its time derivative for all t. In other cases *the fundamental matrix cannot be expressed in terms of matrix exponentials*, and $U(t)(U(x))^{-1} \neq U(t-x)$.

Example 13.1.3.

Fig. 13.1.5 shows the orbits from 40 starting points on the boundary of the square with corners at $y_1 = \pm 1, y_2 = \pm 1$, for the linear system of Example 13.1.2, i.e., $y_1' = -y_1 - y_2, \quad y_2' = y_1 - y_2$. For some values of t, the points reached at time t are joined by straight lines. These are the maps at time t of the square boundary that contains the initial points. That these maps also become squares is due to this special example. The theorem tells, however, that for any linear system, also with variable coefficients, they would have become parallelograms, at least.

Example 13.1.4.

Fig. 13.1.6 shows periodic orbits that start at $t = 0$ from 12 starting points on the y_1-axis, for the famous Volterra **Predator-Prey Model**, see Braun (1975),

$$\frac{dy_1}{dt} = ay_1 - by_1y_2, \qquad \frac{dy_2}{dt} = -cy_2 + dy_1y_2.$$

Figure 13.1.5. *Orbits from 40 starting points.*

Figure 13.1.6. *Orbits for the Predator-Prey Model.*

Parameter values: $a = b = c = d = 1$. In the second Fig. 13.1.6 points reached at the same time t are joined, for a large number of equidistant times. The mappings of the straight line, which contain the initial points, are no straight lines here, since the problem is nonlinear. On a "microscopic" scale the mapping is approximately affine: small (approximate) parallelograms are mapped onto small (approximate) parallelograms, by the matrix $U(t, y(0))$.

In order to avoid overlaps, the computation of Fig. 13.1.6 stopped at $t = 6.4$, just a little bit above the period of small orbits around the critical point $(1, 1)$. (Recall that a critical point is a point, where $f(y) = 0$.) A "motion" that starts exactly at a critical point, remains there forever. The period of of small motions

around $(1,1)$ is in the limit $2\pi = 6.28$. (This is obtained from the eigenvalues of the Jacobian at the critical point.) See Sec. 13.2.2.)

13.1.4 The Logarithmic Norm, Properties and Applications

Propositions 13.1.3 and 13.1.4 are (approximately) valid also for systems of ODEs, if μ^* denotes a bound for the **logarithmic norm**, $\mu(J(t))$.

Definition 13.1.6.

Let $\|\cdot\|$ denote some vector norm and its subordinate matrix norm. Then the subordinate **logarithmic norm** *of the matrix A is given by*

$$\mu(A) = \lim_{\epsilon \to +0} \frac{\|I + \epsilon A\| - 1}{\epsilon} \tag{13.1.17}$$

This limit exists, for any choice of vector norm, since one can show, by the triangle inequality, that, as $\epsilon \to +0$, the right hand side decreases monotonically and is bounded below by $-\|A\|$.

Just like the ordinary norm of a matrix is a generalization of the modulus of of a complex number, the logarithmic norm corresponds to the real part. The logarithmic norm is a real number, and $\mu(A) \leq \|A\|$. It can even be negative, which is very favorable for the estimates that we are interested to make. We shall here only use the logarithmic norm subordinate to the maximum and the l_1 vector norms.

Theorem 13.1.7.

The logarithmic norm subordinate to the max-norm reads,

$$\mu_\infty(A) = \max_i \big(\Re(a_{ii}) + \sum_{j \neq i} |a_{ij}|\big). \tag{13.1.18}$$

If all diagonal elements are real and larger than $-1/\epsilon$, then $\|I+\epsilon A\|_\infty = 1+\epsilon\mu_\infty(A)$. Similarly, the logarithmic norm subordinate to the l_1-norm reads,

$$\mu_1(A) = \max_j \big(\Re(a_{jj}) + \sum_{i \neq i} |a_{ij}|\big). \tag{13.1.19}$$

Proof. First note that, if $a \in \mathbf{C}$ then, $\lim_{\epsilon \to +0} \frac{|1+\epsilon a|-1}{\epsilon} = \Re a$. Set $s_i = \sum_{j, j \neq i} |a_{ij}|$. By (6.2.16),

$$\|I + \epsilon A\|_\infty = \max_i (|1 + \epsilon a_{ii}| + \epsilon s_i),$$

and hence

$$\frac{\|I + \epsilon A\|_\infty - 1}{\epsilon} = \max_i \left(\frac{|1 + \epsilon a_{ii}| - 1}{\epsilon} + s_i \right) \to \max_i (\Re a_{ii} + s_i).$$

The second result then follows, for if $\epsilon a_{ii} \geq -1 \quad \forall i$, then

$$\|I + \epsilon A\|_\infty = \max_i (1 + \epsilon a_{ii} + \epsilon s_i) = 1 + \epsilon \mu_\infty(A).$$

The expression for $\mu_1(A)$ is derived in the same way. □

The most important property of this concept is given in the following theorem that will be proved in Sec. 13.2.3, and is useful for generalization to non-linear systems (Theorem 13.7.3). If you are impatient, please look ahead!

Theorem 13.1.8.
The solutions of a "pseudo-linear" system,

$$\frac{du}{dt} = J(t,u)u + r(t,u), \tag{13.1.20}$$

satisfy the inequality,

$$\frac{d\|u\|}{dt} \leq \mu(J(t,u))\|u\| + \|r(t,u)\|. \tag{13.1.21}$$

Let D_t be a convex domain in $\mathbf{R}^s$, and assume that

$$\mu(J(t,w) \leq \mu^*(t), \qquad \|r(t,w)\| \leq \epsilon(t), \quad \forall w \in D_t.$$

Then, $\|u(t)\| \leq \psi(t)$, where $\psi(t)$ is a solution of a single differential equation,

$$\frac{d\psi}{dt} = \mu^*(t)\psi + \epsilon(t), \qquad \psi(a) = \|u(a)\| \tag{13.1.22}$$

as long as the bound derived from this guarantees that $u(t) \in D_t$.

If μ^*, ϵ are chosen to be independent of t, this leads exactly to the bounds (13.1.9), and the behavior of $\psi(t)$ is illustrated by Fig. 13.1.4.

Comments:

i. *This theorem is formulated, and will be proved, for a general vector norm, and its subordinate logarithmic norm.* The bounds may become much sharper if another norm than the max-norm is used, but we postpone such technicalities to Sec. 13.7.2.

ii. The fact that (13.1.20) leads exactly to same bound as the previous model with discrete error sources illustrated by the staircase curve of Fig. 13.1.3 shows that we can just as well work with a **continuous model for the error propagation**, with functions that interpolate the local error, the step size etc. The continuous model is in several ways more convenient, and we shall see in Sec. 13.4 that, with an amendment that takes into account the numerical stability of the particular method, it also makes sense in more general cases, such as multistep methods, stiff problems etc. We recall the assumptions previously made for the two methods mentioned so far, namely that $\|hf'(y)\|^2 \ll 1$.

We now prove some important properties of the logarithmic norm.

Theorem 13.1.9.
The logarithmic norm has the following properties:

A. $-\|A\| \leq \mu(A) \leq \|A\|$.

B. $\mu(\alpha A + \beta B) \leq \alpha\mu(A) + \beta\mu(B),\ if\ \alpha \geq 0,\ \beta \geq 0.$

C. $\mu(\alpha A + \gamma I) = \alpha\mu(A) + \Re\gamma, \quad if\ \alpha \geq 0,\ \gamma \in \mathbf{C}.$

Proof. Statement A follows from the application of the triangle inequality to the definition of $\mu(A)$. We then note that, for $\alpha \geq 0$,

$$\mu(\alpha A) = \lim_{\epsilon \to +0} \frac{\|I + \epsilon\alpha A\| - 1}{\epsilon} = \alpha \lim_{\epsilon \to +0} \frac{\|I + (\epsilon\alpha)A\| - 1}{(\epsilon\alpha)} = \alpha\mu(A).$$

We can then, without loss of generality, put $\alpha = \beta = 1$ in the rest of the proof. By the triangle inequality,

$$\left\|I + \frac{\epsilon}{2}(A+B)\right\| - 1 = \left\|\tfrac{1}{2}(I+\epsilon A) + \tfrac{1}{2}(I+\epsilon B)\right\| \leq \frac{1}{2}\big((\|I+\epsilon A\|-1) + (\|I+\epsilon B\|-1)\big).$$

Divide the first and the last expression by $\frac{1}{2}\epsilon$, and let $\epsilon \to +0$, and statement B follows (for $\alpha = \beta = 1$).

In order to prove statement C, we consider the identity,

$$\|(1+\epsilon\gamma)(I+\epsilon A)\| - 1 = |1+\epsilon\gamma|(\|I+\epsilon A)\| - 1) + (|1+\epsilon\gamma| - 1).$$

After division by ϵ and passage to the limit, the right hand side becomes $\mu(A) + \Re\gamma$. The left hand side can be written,

$$\|I + \epsilon\gamma I + \epsilon A + O(\epsilon^2)\| - 1 = \|I + \epsilon(\gamma I + A)\| - 1 + O(\epsilon^2),$$

where the triangle inequality was used in the last step. After division by ϵ and passage to the limit, this becomes $\mu(\gamma I + A)$. □

Remark 13.1.1. *Statement B is called* subadditivity. *The non-negativity of α and ηb is important. In general, $\mu(-A) \neq -\mu(A)$. Actually, $\mu(-A) \geq -\mu(A)$, since by the subadditivity,*

$$\mu(A) + \mu(-A) \geq \mu(A - A) = 0.$$

By induction, the subadditivity can be extended to any number of terms *and, by a passage to the limit,* also to infinite sums and integrals. *In particular we have, for the neighborhood average $J(t,u)$ defined by (13.8.12),*

$$\mu(J(t,u)) = \mu\left(\int_0^1 f'(y+\theta u)d\theta\right) \leq \int_0^1 \mu(f'(y+\theta u)d\theta) \leq \max\mu(f'(z)), \tag{13.1.23}$$

where the domain of z is the line segment between y and $y+u$.

Next, we shall prove Theorem 13.1.8 and apply it in the derivation of an inequality for a general nonlinear system of ODEs. Let $u(t)$ be the solution of the system,

$$\frac{du}{dt} = J(t,u)u + r(t,u). \tag{13.1.24}$$

Such a system is called "pseudo-linear", since $J(t,u)$ is a square matrix, the elements of which may depend on u. By Taylor's theorem,

$$u(t+h) = u(t) + hJ(t,u)u(t) + hr(t,u) + o(h), \qquad (h>0),$$

$$\|u(t+h)\| \le \|(I+hJ)u(t)\| + h\|r\| + o(h) \le \|I+hJ\| \cdot \|u(t)\| + h\|r\| + o(h).$$

Subtract $\|u(t)\|$ from the first and the last side, and divide by h.

$$\frac{\|u(t+h)\| - \|u(t)\|}{h} \le \frac{\|I+hJ\|-1}{h}\|u(t)\| + \|r\| + o(1).$$

As $h \to +0$, the left hand side tends to the right-hand derivative of $\|u(t)\|$, and we obtain the result,

$$\frac{d\|u\|}{dt} \le \mu(J(t,u))\|u\| + \|r(t,u)\|. \tag{13.1.25}$$

It is assumed that D_t is a convex domain in $\mathbf{R}^s$, such that for all $w \in D_t$,

$$\mu(J(t,w)) \le \mu^*(t), \qquad \|r(t,w)\| \le \epsilon(t). \tag{13.1.26}$$

Then, by the Comparison Theorem of Sec. 13.2.1,

$$\|u(t)\| \le \psi(t),$$

where $\psi(t)$ is the solution of the scalar differential equation,

$$\frac{d\psi}{dt} = \mu^*(t)\psi + \epsilon(t), \qquad \psi(a) \ge \|u(a]\|, \tag{13.1.27}$$

as long as the bound derived from this guarantees that $u(t) \in D_t$.

For example, if $u(a) = 0$ and if μ^* and ϵ do not depend on t, then $\psi(t) = \epsilon(t-t_0)$, if $\mu^* = 0$, and otherwise

$$\psi(t) = \frac{\epsilon}{\mu^*}(e^{\mu^*(t-t_0)} - 1), \tag{13.1.28}$$

i.e., the same bounds as in (13.1.9). □

Theorem 13.1.10.

Let $z : \mathbf{R} \to \mathbf{R}^s$ be a known function that satisfies the differential inequality,

$$\|z'(t) - f(z(t))\| \le \epsilon(t), \qquad a \le t \le b, \tag{13.1.29}$$

and let $y(t)$ be a solution of the differential system,

$$\frac{dy}{dt} - f(y) = 0, \qquad a \le t \le b.$$

Assume that, for every $t \in [a,b]$, there exists a real-valued function $\mu^*(t)$, and a convex domain $D_t \subseteq \mathbf{R}^s$ that contains the point $z(t)$, such that

$$\mu(f'(y)) \le \mu^*(t), \qquad \forall y \in D_t.$$

Then $\|z(t)-y(t)\| \le \psi(t)$, where $\psi(t)$ is a solution of the scalar differential equation,

$$\frac{d\psi}{dt} = \mu^*(t)\psi + \epsilon(t), \qquad \psi(a) \ge \|z(a) - y(a)\|,$$

as long as $\{w : \|w - z(t)\| \le \psi(t)\} \subseteq D_t, \quad t \in [a,b]$, i.e. as long as the bounds obtained from this, e.g. (13.1.28), guarantee that $y(t)$ stays in D_t.
(The union of the sets D_t is to be thought of as a hose or "a French horn" enclosing the orbit.)

Proof. Set $u(t) = z(t)-y(t)$. Note that we can write $u'(t) = f(z(t)) - f(y(t)+r(t)$, where $\|r(t)\| \le \epsilon(t)$. By Lemma 12.2.1,

$$f(z(t)) - f(y(t)) = \int_0^1 f'(z(t) + \theta w)dw(z(t) - y(t)) = J(t, u(t))u(t).$$

By (13.1.23), $\mu(J(t,u(t)) \le \max_y \mu(f'(y)) \le \mu^*(t)$, $y \in D_t$. Hence, $u(t)$ satisfies a non-linear variational equation of the form, $u'(t) = J(t,u)u + r(t)$, where $\|r(t)\| \le \epsilon(t)$. The result then follows from Theorem 13.1.8. □

Theorem 13.1.10 is, for the sake of simplicity, formulated for an autonomous system. It is, mutatis mutandis, *valid also for a non-autonomous system $y' = f(t,y)$.*

Theorem 13.1.11.

A. $\|e^{At}\| \le e^{\mu(A)t}$, *if* $t \ge 0$.

B. *The real part of an eigenvalue of A cannot exceed $\mu(A)$.*

C. $\|Au\| \ge |\mu(A)|\ \|u\|$, *if* $\mu(A) < 0$.

D. $\|A^{-1}\| \le |\mu(A)|^{-1}$, *if* $\mu(A) < 0$.

Demonstration: The system $u' = Au$ has the solution $u(t) = e^{At}u(0)$. Then, by Theorem 13.1.8, $\|e^{At}u(0)\| \le e^{\mu(A)t}\|u(0)\|$, if $t \ge 0$. Since this is true for every vector $u(0)$, statement A follows.

In order to prove statement B, note that if $Av = \lambda v$ then $\|e^A v\| = \|e^\lambda v\| = e^{\Re\lambda}\|v\|$. Then, by statement A, $e^{\mu(A)}\|v\| \ge \|e^A v\| = e^{\Re\lambda}\|v\|$. This proves statement B.

By the definition of the logarithmic norm, we have, as $\epsilon \to +0$,

$$\mu(A) \geq \frac{\|u + \epsilon Au\| - \|u\|}{\epsilon \|u\|} + o(1) \geq -\frac{\|Au\|}{\|u\|} + o(1) \quad \forall u.$$

The triangle inequality was used in the last step. Statement C follows.

By the last formula,

$$-\mu(A) \leq \inf_u \frac{\|Au\|}{\|u\|} = \inf_v \frac{\|v\|}{\|A^{-1}v\|} = \frac{1}{\|A^{-1}\|}.$$

Since $\mu(A) < 0$, this proves statement D. A completely different proof is indicated by the hints of Problem 7 of Sec. 13.1. □

Review Questions

1. Explain, with an example, how a differential equation of higher order can be written as a system of first-order differential equations.
2. Define local and global error, and explain with a figure the error propagation for a one-step method for a scalar differential equation. What is meant by order of accuracy?
3. (a) Formulate the basic existence and uniqueness theorem for initial value problems for the system $y' = f(y)$. Set up the iteration formula used in the demonstration.

   (b) Demonstrate the existence and uniqueness theorem. What is the relation between the Lipschitz constants of $f(y)$ and the iteration operator?

   (c) Give a simple example of a differential equation, where the solution escapes to ∞ in finite time. Explain why this does not contradict the existence and uniqueness theorem.

   (d) Show that the initial value problem $dy/dt = y^{2/3}$, $y(0) = 0$, has an infinite number of solutions. Explain why this does not contradict the existence and uniqueness theorem.

   (e) What is a critical point p of an autonomous system $y' = f(y)$? Let p be a critical point. Prove the following statement, under appropriate conditions:
   If $y(t_1) = p$, at some time t_1, then $y(t) = p$ for all times x.
   Can an orbit reach a critical point in finite time, if it is not there from the beginning? (Motivate the answer.)

   Hint: Use the uniqueness theorem.
4. Define and derive the variational equation, and state a few results that can be derived from it. In Example 13.1.4 (Predator-Prey Problem), it is stated that "on a microscopic scale the mapping is approximately affine". Explain this statement, and give a motivation for it.

5. (a) Give the general definition of the logarithmic norm of a matrix.

(b) Describe a formula or algorithm for the computation of the logarithmic norm subordinate to the l_p-norm, for $p = 1, 2, \infty$. Derive it, assumed that the corresponding formula or algorithm for the usual matrix norm is given.

(c) Show that if $u' = J(t)u + r(t)$, $t \geq a$, then $\|u\|'(t) \leq \mu(J(t))\|u(t)\| + \|r(t)\|$. Also show that $\|y(t)\| \leq \psi(t)$, where $\psi(t)$ satisfies an ODE of the form $\psi' = \mu^*(t)\psi + r^*(t)$. Give the conditions that $\psi(a), \mu^*(t), r^*(t)$ satisfy. Solve this equation, if μ^* and r^* are constant, and compute $\lim_{t\to\infty} \psi(t)$. Sketch the behavior of $\psi(t)$.

(d) Generalize the previous question to an approximate bound for $z(t) - y(t)$, where $y(t), z(t)$, satisfy the ODEs, $y' = f(y)$, $z' = f(z) + r(t)$, under appropriate conditions that include the conditions of the previous question as particular cases.

(e) Tell five of the most important general properties of the logarithmic norm, in particular the subadditivity.

(f) Let $\mu(\cdot)$ be the logarithmic norm subordinate to a vector norm $\|\cdot\|$. Define a new vector norm $\|\cdot\|_T$ by the relation $\|y|_T = \|T^{-1}y\|$, and let $\mu_T(\cdot)$ be the subordinate logarithmic norm. What is the relationship between $\mu_T(\cdot)$ and $\mu(\cdot)$?

Problems

1. Rewrite the system

$$y'' = t^2 - y' - z^2,$$
$$z'' = t + z' + y^3,$$

with initial conditions $y(0) = 0$, $y'(0) = 1$, $z(0) = 1$, $z'(0) = 0$, as an initial value problem for a system of first-order equations.

2. (a) Find the general solution to the differential equation $y' = y/t$. Where is the Lipschitz condition not satisfied? Study the behavior of orbits in the neighborhood of such singular points.
Also study the system $y_1' = y_1, \quad y_2' = y_2$. Note that the singular point of the single differential equation corresponds to a critical point of the system.

(b) Study in a similar way the differential equation $y' = at/y$, and the system $y_1' = ay_2$, $y_2' = y_1$. How does the character of the solution manifold depend on the parameter a?

3. (C. Moler, personal communication.) Consider the initial value problem, $dy/dt = \sqrt{1 - y^2}$, $y(0) = 0$. Show that the Lipschitz condition is not satisfied at $y = 1$, but nevertheless the problem has a unique *real* solution for positive

values of t, namely:

$$y(t) = \begin{cases} \sin t, & \text{if } 0 \le t \le \frac{1}{2}\pi \ , \\ 1, & \text{if } t > \frac{1}{2}\pi. \end{cases}$$

Hint: Derive contradictions from the assumptions that $y(t) > 1$ or $y(t) < 1$ for some $t > \frac{1}{2}\pi$.

4. (a) Study the differential equation, $y' = -1 - 2\text{sgn}(y)$, with initial condition $y(0) = 1$. If we insist on the convention that $\text{sgn}(0) = 0$, show that there is no solution for $t > \frac{1}{3}$. Why does not this contradict the existence and uniqueness theorem?
Find out, with paper and pencil or with a computer, what happens if you apply Euler's method with constant step size to this equation.

(b) This problem is related to the study of the motion of particle in a slope with Coulomb friction (dry friction).
Notations: the friction coefficient is γ, the angle between the plane and the horizontal plane is α, $0 < \alpha < \pi/2$, and the acceleration of gravity is g.
Let the velocity at time t be $v(t)$. $v(0) > 0$ is given. The positive direction is uphill. The equation of motion then reads:

$$\frac{dv}{dt} = \begin{cases} 0, & \text{if } v = 0 \text{ and } \gamma \ge \tan(\alpha); \\ -g\sin(\alpha) - \gamma g \cos(\alpha)\text{sgn}(v), & \text{otherwise.} \end{cases}$$

Is it true that this problem has precisely one solution for all $t > 0$?
What value is implicitly given to $\text{sgn}(0)$ in this formulation?
For what relation between α and γ can we obtain the initial value problem in (a) after appropriate rescaling of v and t?

Note: For the sake of simplicity, we ignored the distinction between the friction coefficients at rest and at motion. Also note that the particle reaches a critical point in finite time. According to the comments after Theorem 13.1.1 this would have been impossible for a system that satisfies a Lipschitz condition everywhere.

5. Sometimes the perturbations grow much faster than the exact solution itself. Verify that the problem $y' = y - 2t/y$, $y(0) = 1$, has the solution $y = \sqrt{2t+1}$. Show by the linearized variational equation that a small perturbation at $t = 0$ will be amplified by the factor $e^{2t}/\sqrt{2t+1}$. Compare the growth of the perturbations and the growth of the solution over the interval $(0, 12)$. (See Computer Exercise 5.)

6. (a) Four of the orbits in Fig. 13.1.5 seem to be envelopes of families of straight lines. Explain this observation.

(b) Determine theoretically the speed of the rotation and shrinking of the squares in the figure, and make, for comparison, some measurements in the figure.

7. Let $y(t)$ be the solution of the initial value problem $y' = t^2 + y^2, \quad y(0) = 1$. Show that the solution is still finite at $t = 0.833$ by comparing $y(t)$ with a solution of the differential equation, $z' = a^2 + z^2$, for some suitable choice

of a. Also show theoretically that $y(t)$ becomes infinite before $t = 1$. (The numerical determination of the point where $y(t)$ was requested in computer exercise 12.1.8.)

8. A study of a population subject to competition (crowding) and toxines. (Extension of a model treated in Klamkin [9, 1987, p.317].)
Let $u(t)$, $v(t)$ be, respectively, the size of a population and the amount of toxines at the time t. Assume that the difference $r(u)$ between the birth rate and the death rate, due to other causes than the toxines, is of the form $r(u) = k_0 - k_1 u$, like in the logistic model (which is a particular case of this problem).
Furthermore, assume that $k_2 uv$ individuals are killed, per unit of time, by toxines in the environment. These toxines are produced by the population, and assume that the amount produced per unit of time is $k_3 u$. Finally, assume that that the toxines have a spontaneous exponential decay. All parameters are *non-negative*. This leads to equations of the form

$$u' = (k_0 - k_1 u)u - k_2 uv, \quad v' = k_3 u - k_4 v.$$

(a) Reduce the number of parameters from 5 to 2 by scaling t, u, v, i.e. set $t = lx, u = my_1, v = ny_2$, so that we obtain a differential system of the form

$$y_1' = y_1 - y_1^2 - a y_1 y_2, \quad y_2' = y_1 - b y_2. \tag{13.1.30}$$

Express a, b, l, m, n in terms of the original parameters. See also Computer Exercise 12.2.2.

(b) Investigate the critical points and their stability, if $a > 0$, $b \geq 0$, in particular the limits as $t \to \infty$.

(c) Is it true that, with this model, neither y_1 nor y_2 can become negative, if the initial values are positive? Assume that $y_1(0)$ is a small positive number and that $y_2(0) = 0$. Find an upper bound for y_1.

9. Compute $\exp(\|A\|_\infty)$ and $\exp(\mu_\infty(A))$ for

$$A = \begin{pmatrix} -10 & 1 \\ 1 & -10 \end{pmatrix}.$$

Let $y(t)$ be the solution of an equation of the form $y' = Ay + r(t), y(0) = 0$, where $\|r(t)\|_\infty \leq 1$. Find a constant c such that $\|y(t)\|_\infty \leq c$ for all positive t.

10. Show the following relations:

$$\|(I - hJ)^{-1}\| \leq (1 - \mu(hJ))^{-1}, \qquad \text{if } \mu(hJ) < 1,$$

$$\|(A - zI)^{-1}\| \leq (\Re z - \mu(A))^{-1}, \qquad \text{if } \Re z > \mu(A).$$

11. There are several alternative definitions of the logarithmic norm, some of which are better suited for a generalization to semibounded operators in infinite-dimensional spaces. Show that $\mu(A)$ is equal to the limits of the following expressions,

$$\frac{\|(I-\epsilon A)^{-1}\|-1}{\epsilon}, \quad (\epsilon \downarrow 0), \qquad \|A+kI\|-k, \quad (k\to\infty).$$

Use the second expression in an alternative derivation of Theorem 13.1.8: set $u = e^{-kt}v$, derive a differential inequality for $\|v\|$, then return to $\|u\|$. Show that the inequality becomes sharpest in the limit, $k\to\infty$.

12. (a) Set $\phi(t;A) = t^{-1}\ln\|e^{At}\|$. Show that $\mu(A) = \lim_{t\downarrow 0}\phi(t;A) = \sup_{t>0}\phi(t;A)$. What happens to $\phi(t;A)$, as $t\to\infty$, and what is $\inf_{t>0}\phi(t;A)$? (Compare Problem 4 of Sec. 10.2.)

(b) Discuss the analogous questions for $t<0$, $t\uparrow 0$, and $t\to-\infty$.

13. (a) Assume that all solutions of the linear system $u' = A(t)u$ satisfy the inequality $\|u(t)\| \le c_0\|u(x)\|\ \ \forall t\ge a,\ \ \forall x\in(a,t)$. Let $U(t)$ be the fundamental matrix solution of this linear system, see (13.1.11) and Theorem 13.1.5. Show that $\|U(t)U(x)^{-1}\| \le c_0$.

(b) Assume that $B(t,u)$ is a matrix-valued function such that $\|B(t,u)\| \le c_1,\ \forall t\ge a$ if $\|u\|\le c_2(t)$. Then show that all solutions of the pseudo-linear system $u' = \big(A(t)+B(t,u)\big)u$ satisfy the inequality,

$$\|u(t)\| \le c_0\|u(a)\| + c_0\int_a^t c_1\|u(x)\|dx,$$

as long as this inequality implies that $\|u(t)\|\le c_2(t)$. Generalize to the case when c_1 is allowed to depend on t.

(b) **The Gronwall-Bellman Lemma.**: Let $g(t)$ be a differentiable function, and let $k(t)$ be a continuous function, such that $k(t)\ge 0,\ \forall t\ge a$, and set $K(t) = \int_a^t k(x)dx$. Assume that a continuous function $y(t)$ for $t\ge a$ satisfies the inequality

$$y(t) \le g(t) + \int_a^t k(x)y(x)dx. \tag{13.1.31}$$

Show that

$$y(t) \le g(t) + \int_a^t k(x)g(x)e^{K(t)-K(x)}dx,$$

Show that if $k(t)=k>0$, $g(t)=g$ are constant and $a=0$, then $y(t)\le ge^{kt}$. Apply the result to find a bound for $\|u(t)\|$ in Problem (a).

Hint: Let $w(t)$ be the solution of the integral equation obtained when the inequality in (13.1.31) is replaced by an equality. Note that $w(t)$ is the solution of the differential equation $w'(t) = g'(t) + k(t)w(t),\ w(a) = g(a)$. Solve this differential equation by the formula of Theorem 13.1.5 and integrate the result by parts.

Note: This lemma gives an alternative approach, to some of the questions

treated by the logarithmic norm technique in this book, sometimes with stronger results. On the other hand, the restriction to non-negative functions $k(t)$ is not needed in the logarithmic norm technique. An analogous result for difference equations is given as a problem of Sec. 13.3.

Computer Execrcises

1. Write a program for Runge's 2nd order method, according to Example 1.3.1, or with some other level of ambition. It is good, if it can be used to treat most of the computer exercises below. They are *non-stiff* problems. With a possible exception for the last exercise (Arenstorf's equation) you can obtain results matching the resolution of the screen with rather short computing time on a personal computer.

The program should be convenient to apply to e.g. different initial conditions, different values of TOL, and to different values of a small number (e.g. three) of parameters of the differential system. The program should be able to provide output in numerical and, if possible, also in graphical form. The numerical output should be stored in such a form that it can be processed further outside the program, e.g. for interpolation (in order to obtain a neat table) or for finding intersections of the orbit with some curve, or for the study of the step size sequence. It was pointed out in Example 1.3.1 that linear interpolation is sufficient for this purpose.

Read the texts of the exercises below in order to find hints about what should be included in a program to be used not only for the demonstration of the orbits, but also for the experimental study of a numerical method for ODEs. Distinguish between what can be done in *the general part* and what should be done in *problem dependent parts* of your program, and think of the communication between them.

Think also of convenient program tools for the input of the parameters of the problem and the method, for the interaction during a run, and for the post-processing of results that the program has produced. Concerning the input: it is convenient to have a prepared set of default values of the input data, which can be overwritten by an input from the keyboard or by a driver program. The details will depend on your choice of language and the possibilities for interaction.

Note that a rough guess of the size of the first step is needed. The step size control will usually soon find a suitable step size, even if the guess is off by several powers of ten. The program sketched in Example 1.4.1, usually finds a good step size faster, if the guess is too large than if it is too small.

We suggest that the program should be able to work in the following four modes:

(i) Automatic step size control, according to Sec. 1.4.

(ii) Constant step size

(iii) According to a prescribed sequence of times $t_0,\ t_1,\ t_2, \ldots$, usually obtained from a previous run. (This is useful when you are interested in differences between solutions with small changes of parameters or initial values. See also comments after Theorem 13.8.2.)

(iv) Every time step defined by the sequence $t_0,\ t_1,\ t_2, \ldots$ is divided into two steps of equal length. (This is useful for the estimation of the global error, when the exact solution is not known, and for the improvement of the results afterwards by Richardson extrapolation. This yields 3rd order accuracy.

It is in most cases suitable to begin with $TOL = 10^{-3}$. You can then judge, for what accuracy you will still get a reasonable computing time and a reasonable size of the output to be used at the post-processing.

You may also obtain good hints by looking at other codes for ODEs, e.g., those contained in Hairer et al. [7, 1993] Run also some of the computer exercises of this chapter with some professional package, such as LSODE, EPISODE, DIFEX1 or DEABM. They are described in Hairer et al. [7, 1993, p.374 ff.].

2. Write a code for the equation of Example 13.2.2, i.e. $y' = -my^{1+1/m}$, $y(0) = 1$.

(a) Run the program for $m = \frac{1}{2}$ from $t = 1$ to $t = 10000$ with two different tolerances. Plot $\log y$, $\log h$, $N(t)/100$ and the logarithm of the relative error, divided by TOL, versus $\log t$ Compare the plotted quantities with the values predicted in Example 13.2.2.

(b) Run also the case $m = 1$ with similar output over a longer time. Automatic interrupt when $y < 10^{-3}\,TOL$. Will the step size ever be restricted by numerical instability?

3. Consider systems of the form $y' = Ay$, $y(0) = c$, where A is a constant 2×2 matrix. Run a few cases, including the following:

(a) Run "the circle test" with Runge's 2nd order method. It is written in Problem 13.2.11c as a scalar complex equation, $z' = iz$, $z(0) = 1$. If your program can handle complex variables conveniently, you may run it in this form, but you can also transform it to an equivalent real system of the form $y' = Ay$. Determine by some combination of experimental and theoretical work, how small the tolerance and the step size have to be, in order that the circle should not be "thick" on your screen, even after 10 revolutions. Run it also with a larger tolerance, so that it becomes thick after two or three revolutions. Does it spiral inwards or outwards, when it has become "thick" ? Compare your experiment with the theoretical analysis of Problem 17c. Is it true that the thickness depends much more on the linear interpolation used in the graphics than on the truncation error of the method? Do your *numerical* results in your file "spiral" in the same way (outwards or inwards) as the curve you see on your screen?

Perform the circle test also for Euler's method, and compare with the theoretical results of Problem 13.2.11c.

(b) Reproduce Fig. 13.1.5. This equation can also be written as a scalar complex equation. Notice that you can run all the 40 curves by a for-loop in the

"function". This gives you a system of 40 complex equations or 80 real equations. Can you handle the output from such a big system, or do you prefer to cut it into smaller pieces?

(c) Consider the system $y_1' = y_2;\ y_2' = y_1$. (This cannot be written as a scalar complex equation.) Show that the orbits are hyperbolas of the form $y_1^2 - y_2^2 = c$. Run a few cases with initial conditions of the form $y_1(0) = -1,\ y_2(0) = 1 - \delta$ and $y_1(0) = -1 + \delta,\ y_2(0) = 1$, where $0 < \delta \ll 1$. Take e.g. $\delta = 0.005$ and $\delta = 0.02$.

Look at the orbits on the screen. One of the asymptotes seems to be "attractive" and the other is "repulsive". Explain why.

Look at the numerical output. Make a conjecture about how the shortest distance from the origin depends on $\delta,\ 0 < \delta \ll 1$.

Prove the conjecture.

4. Run the shot problem, described in Sec. 1.4. Reproduce Fig. 1.4.1. Take $TOL = 10^{-4}$ and another tolerance. Interrupt after that y has become negative for the first time. Use linear inverse interpolation to find the landing point. Compare the accuracy and the number of function evaluations with the results given in Sec. 1.4 with and without Richardson extrapolation.

This particular problem can be run efficiently with constant step size. Run it also with two different step sizes and make Richardson h^2-extrapolation of the landing point. (Alternatively, do this with variable step size according to mode (iv) described in Exercise 1.)

Note: The interpolation error depends of the location of the point within the step. A consequence is that the error of a value obtained by linear (inverse) interpolation may not show the same proportionality to TOL or h^2 as the errors at the points primarily computed by the integration routine. This can be overcome by the use of higher order (inverse) interpolation.

5. (a) Run the Predator-Prey problem as in Example 13.1.4, with TOL (or a constant step size) chosen so that the curves (see Fig. 13.1.6) become closed curves to the accuracy of your screen. Determine the periods.

(b) Make a second run, and apply Richardson extrapolation to improve the estimates of the periods. (See the Note of Exercise 4.)

(c) A modified Lotka-Volterra model reads

$$y_1' = (a - \epsilon y_1 - b y_2) y_1, \quad y_2' = (-c + d y_1) y_2.$$

Choose (say) $\epsilon - 0.2$, and run it with the same parameters and initial values as before. Note that the qualitative character of the orbits changes a lot.

6. The solution of the differential equation $y' = t^2 + y^2,\ y(0) = 1$ tends to infinity at $t = a$, where a is to be determined to (say) 3 or 4 decimal places. (See also Problem 12.2.5)

(a) Set $y = 1/u$, and solve the differential equation for u numerically. Interrupt when u has become negative, and determine a by inverse interpolation.

(b) In a more complicated context the kind of transformation suggested in (a) may be impractical, and we shall therefore see what can be done with a

more direct approach to the original equation and an appropriate termination criterion.

One criterion of a general nature is to stop when $t_{n+1} = t_n$ in the computer. This criterion is worthwhile to have in the program, together with some suitable message, also for many other reasons. Determine how many steps it needs, for various tolerances. How well is a determined? How big is y when it stops?

Another idea is to use Aitken extrapolation of the sequence t_n. Stop when two successive extrapolated values differ by some fraction of TOL. (The fraction is to be tuned.) Determine how many steps it needs for various tolerances, and how well a is determined. Does the cancellation in the denominator of the Aitken formula cause trouble? If the idea works well, try to give some theoretical support for it, for example on the basis of the theoretical step control strategy.

(c) Make some experiments, in order to see, if the two strategies described in (b) work for other equations of the form $y' = t^2 + y^c,\ y(0) = 1,\ (c > 1$, not necessarily an integer).

7. The numerical solution of the differential equation $dy/dt = f(t)$ may be a practical way to perform numerical quadrature, due to the well developed techniques for the automatic control of stepsize in the packages for solving ODEs. Test your program, and some more professional package, on the computation of the slowly convergent integral

$$\int_0^\infty (t^3 + t^2 + 1)^{-1/2}\, dt,$$

to (say) 4 decimal places, and make a graph or a table that shows the variation with t of the step size and the number of steps. Is it true that the latter grows like $a + b \log t$, for $t \gg 1$?

Decide yourself to what extent one or two terms of an expansion like the one in Example 3.1.9 are useful for large t, e.g., in order to determine when to stop. How do you choose the tolerance and estimate the error? (See computer exercise 1 of this section, suggestion (iv).)

13.2 Control of Step Size and Numerical Stability

In applied mathematics, a theoretical study of a continuous model of a system with discrete structure, is often easier than a direct approach to the original discrete problem; the study of error propagation is one example of this. In Sec. 13.1.3 the error propagation in the numerical treatment of a single differential equation by a one-step method was treated by a direct discrete approach, although eventually a sum was approximated by an integral. The result is the approximate bound (13.1.8), illustrated in Fig. 13.1.4. Later the same type of bound was obtained in Theorem 13.1.5, see (13.1.20), for the change of the solution of a differential system caused by a continuous perturbation function $r(t, u)$, indicating that such a function can

be used as a model for the sequence of error pulses. This is, however, not the whole truth. Unless ceratin restrictions are satisfied. the error propagation mechanism of the numerical compuatations can namely be rather different from this mechanism of the differential system itself, for which the (exact) variational differential equation tells the story.

For one-step methods we assume to begin with that

$$\|hf'(y)\|^2 \ll 1. \tag{13.2.1}$$

In Sec. 13.4 we shall derive the same *continuous model for error propagation* in a different way for the study of multistep methods. For them we have to add the requirement that they be strongly zero-stable (see Def. 13.2.3), a condition that is automatically satisfied by consistent one-step methods.

For some numerical methods condition (13.2.1) can be relaxed considerably and this opens the gate for the efficient application to an important class of differential systems called **stiff**. This encourages a further development in Sec. 13.2.1 of the continuous model to include other important features of practical computation. The results in Theorem 13.2.1 and Fig. 13.2.1 have the same simple structure as (13.1.8) and Fig. 13.1.4, but a more realistic interpretation.

An introduction to numerical stability is given in Sec. 13.2.2. Implicit and linearly implicit methods with applications to stiff and differetial-algebraic systems, will be discussed in Sec. 13.2.3.

13.2.1 Scale Functions and Step Size Control

The automatic control of step size is an important issue in the numerical treatment of ODEs. Unfortunately, in many cases, the assumptions used in the derivation of (13.1.9). lead to a rather inefficient step size sequence. We shall now modify the assumptions in a more useful way, and derive results of the same simple structure as (13.1.9), Fig. 13.1.4 and Theorem 13.1.8, but the interpretation is more general. It just means rather simple transformations of the variables t and u in (13.1.20) and the results of the theorem.

Practical problems often contain the following two complications:

A. Time steps of very different orders of magnitude may be needed in different parts of the orbit. We shall therefore introduce a function $\tau(t)$ that describes a **local time scale** of the motion. This is in the same spirit as many notions in other branches of Applied Mathematics, such as halving time, half width etc. Roughly speaking, the local time scale of a function should be the length of an interval around t, where the value of the function changes notably. To begin with, let $\tau(t)$ be any positive, piecewise continuous function. We then define a function $\xi(t)$, called the **age** of the motion, that measures small increments of time in the local time scale, i.e.

$$\xi(t) = \int_a^t \frac{dx}{\tau(x)}. \tag{13.2.2}$$

B. The size of different components of $y(t)$ can differ by several orders of magnitude, and an individual component can vary by several orders of magnitude

along an orbit. So we need another scale function too, **a diagonal scaling matrix** $S(t)$ for measuring errors etc.. The request for accuracy may be in terms of relative accuracy for each component. Sometimes it can be rather expensive to obtain high relative accuracy for example near a point, where a component is zero. In some problems high relative accuracy may be necessary also in such cases, but in other problems it is not, and the computer should be able to take advantage of this. In some packages, the user can, for each component y_i, give a non-negative scale factor s_i (or accept a default value given by the program) that defines a breakpoint: relative accuracy is requested when $|y_i| > s_i$ and absolute accuracy is requested when $|y_i| \leq s_i$. The non-zero elements of the diagonal matrix $S(t)$ are defined thus,

$$S_i(t) = \max(|y_i(t)|, s_i), \qquad i = 1, 2, 3, ..., s. \tag{13.2.3}$$

Note that requests for either pure relative or pure absolute accuracy comes as particular cases, with appropriate choice of the factors s_i. The exact formulation of all this may vary between implementations, which is also the case for several other details of this discussion. The general strategy described here has been applied in several programs by one of the authors.

If $u(t)$ is the absolute global error, then the **scaled global error** $v(t)$ (or the "mixed absolute-relative" error) is defined by the equation,

$$S(t)v(t) = u(t), \tag{13.2.4}$$

and similarly for the local error vector. Let $l(t)$ be an estimate of the absolute local error vector in the time step that leads to t. Our **theoretical step control strategy** is to determine the step size $h(t)$ so that

$$\|S(t)^{-1}l(t)\|\tau(t)/h(t) \approx TOL, \tag{13.2.5}$$

where TOL is a tolerance to be given by the user. For most methods $l(t)$ is approximately proportional to h^{p+1} (at a given point t), where p is called the **order of accuracy**, or **order of consistency**, of the method. (The order of consistency becomes the order of accuracy, if the method is numerically stable.) We can therefore write the relation (13.2.5) in the form,

$$ah^p \approx 1.$$

Let h_1 be the step size just used. If $ah_1^p > 1$, the step is to be recomputed with step size

$$h = h_1 \cdot (ah_1^p)^{-1/p},$$

otherwise the step is accepted, and this value, divided by a safety factor, is suggested for the size of the next step. The safety factor is to be tuned by the implementer. It accounts for some approximations used in the error estimate and, above all, for the fact that the value is based on information from the past, while it is to be used for the step size of the near future.

Now we return to the continuous model for error propagation. Introduce the scaled error vector v and the age ξ into (13.1.20), i.e.

$$u = Sv, \qquad dt = \tau d\xi, \qquad r(t) = l(t)/h(t),$$

$$\frac{d(Sv)}{d\xi} = \tau(JSv + l/h),$$

Since $d(Sv)/d\xi = Sdv/d\xi + (\tau dS/dt)v$, we obtain,

$$\frac{dv}{d\xi} = \tau\left(S^{-1}JS - S^{-1}\frac{dS}{dt}\right)v + \frac{\tau}{h}S^{-1}l. \tag{13.2.6}$$

This equation has the same structure as equation (13.1.20). By (13.2.5), the norm of the last term is approximately equal to TOL or a little smaller. So, we have the following consequence of Theorem 13.1.8:

Theorem 13.2.1.

Let $S(t)$ be the scale function defined by (13.2.3), and let the step size control be defined by (13.2.5), i.e. the error per local unit of time should be approximately equal to TOL. The age ξ of the motion is defined by (13.2.2). Assume that in a neighborhood D of the orbit,

$$\tau\mu\Big(S^{-1}f'(y)S - S^{-1}dS/dt\Big) \le \mu^*. \tag{13.2.7}$$

Then, at age ξ, the norm of the scaled global error, $v(\xi)$, does not exceed $g(\xi)$, where

$$\frac{dg}{d\xi} = \mu^* g + TOL, \qquad g(0) = \|v(0)\| = \|S^{-1}u(0)\|. \tag{13.2.8}$$

Hence

$$\|v(\xi)\| - \|v(0)\|e^{\mu^*\xi} \le \begin{cases} TOL\,\xi, & \text{if } \mu^* = 0; \\ \frac{TOL}{\mu^*}(e^{\mu^*\xi} - 1), & \text{if } \mu^* \ne 0;. \end{cases} \tag{13.2.9}$$

In particular, if $\mu^* < 0$, then $\|v(\xi)\| \le g(\xi) \to -TOL/\mu^*$, as $\xi \to \infty$.

If needed, this result can be applied with piece-wise constant μ^* with the age measured from the most recent breakpoint. So we have the same situation as in Fig. 13.1.4 and (13.1.9), but it allows more general interpretations:

1. The errors are scaled by a diagonal matrix $S(t)$, which allows also "mixed absolute-relative errors".

2. The independent variable is the age ξ instead of the ordinary time t.

3. μ^* is defined by (13.2.7). It is usually more reasonable to assume a uniformly valid bound for τJ than for J itself, provided that the local time scale has been adequately defined.

Up to this point the results are valid for any positive function $\tau(t)$. Now we shall specify the notion of local timescale more, and see that it can be quite easy to apply in practice. Assume that we work with a numerical method of order p. There are two alternative natural definitions of $\tau(t)$. Set

$$a_m(t) \approx \|S^{-1}y^{(m)}(t)\|. \tag{13.2.10}$$

Figure 13.2.1. *Scaled global error divided by TOL versus age for* $\mu^* = \pm 1, \pm 0.5, 0$.

We use approximate equality here, since some sort of averages must be used in order to avoid nuisance with zeros of the mth derivative. In practice this nuisance is diminished also by a moderation our general strategy by a bound for the ratios of two consecutive step sizes, which is suitable also for other reasons. The two definitions are as follows, where q stands for quotient, and r stands for root.

$$\tau_q(t; y, p) = a_p(t)/a_{p+1}(t), \qquad \tau_r(t; y, p) = a_p(t)^{-1/p}, \tag{13.2.11}$$

or shorter: $\tau_q(t) = a_p(t)/a_{p+1}(t)$, $\tau_r(t) = a_p(t)^{-1/p}$. The "dimension" of both measures is "time".

For several methods, e.g. linear multistep methods, the local error in a step is, approximately,

$$l(t) = c_p h^{p+1} y^{(p+1)}(t), \tag{13.2.12}$$

where p is the order of accuracy, and c_p is called the **error constant** of the method. We now use τ_q. By (13.2.16),

$$TOL \approx \|S^{-1} l(t)\| \tau_q(t)/h(t) = \|c_p h^p S^{-1} y^{(p)}(t)\| = c_p h^p a_p(t) = c_p (h/\tau_r)^p. \tag{13.2.13}$$

In practice, the derivatives are approximated by finite differences. By the way, this also reduces the risk for the nuisance mentioned above. The expression on the right hand side of (13.2.13) is even easier to to compute than $l(t)$, and it demands less storage. *The program does not need to estimate the local time scale explicitly!*

Theorem 13.2.2.

Assume that equation (13.2.12) is approximately valid, and set

$$k = |TOL/c_p|^{1/p}.$$

Let $\xi_r(t)$ *be the age function defined by (13.2.2) with* $\tau(t) = \tau_r(t)$, *and let* $N(t)$ *be the number of steps needed for the interval* $[a, t]$, *with the step control strategy*

described above. Then

$$h(t) \approx \tau_r(t)k, \qquad N(t) \approx \xi_r(t)/k.$$

Proof. By (13.2.13),

$$TOL \approx |c_p|\,|h/\tau_r|^p.$$

This proves the first relation. We use this to establish the second relation:

$$N(t) = \sum_{x_i \le t} \frac{h(x_i)}{h(x_i)} \approx \int_a^t \frac{dx}{\tau_r(x)k} = \frac{\xi_r(t)}{k}.$$

□

Comment: Typical values of k are between 0.01 and 0.1 times the number of function evaluations in a step, if p is adequately chosen for the accuracy requested; a higher value of p when high accuracy is requested. For Runge's 2nd order method $0.02 < k \approx \sqrt{6TOL} < 0.2$ roughly means that $7\,10^{-5} < TOL < 7\,10^{-3}$ is an adequate range for TOL. For problems of moderate size the method can well be used for smaller tolerances, if no higher order method is conveniently available.

In the 1960's there was a lively discussion whether to compare the tolerance with the local error per step or the local error per unit of time. Our strategy gives a kind of Solomonic answer: we use the local error per unit of time in the local time scale,$\tau_q(t)$). Formally this sounds like accepting the second opinion, but one can show (Problem 12) that

$$\|S^{-1}l(t)\| = TOL|TOL/c_p|^{1/p}\tau_r(t)/\tau_q(t).$$

So, if $\tau_r(t)/\tau_q(t)$ is constant during a motion, *this strategy gives the scaled local error per step a constant norm along an orbit.* That sounds more like accepting the first opinion. If relative error is requested, it can be shown (Problem 10) that $\tau_r(t)/\tau_q(t)$ is indeed constant, if $y(t)$ is an exponential or a power, but it is usually not exactly like that.

Typically, $\tau_r(t)/\tau_q(t)$ fluctuates between rather moderate bounds, and these measures of local timescale usually do not depend heavily on p either, if (say) $2 \le k \le 8$. These notion have no high precision, but they are useful for decision. The notions of age and local timescale describe features of a motion that are related to, respectively, the amount of work and the step size needed in the numerical simulation. The role of the tolerance is concentrated in the constant k,and so is, to a large extent, also the dependence of the numerical method.

Example 13.2.1.

Fig. 13.2.2 shows, for $p = 2$ and $p = 5$, the functions ξ_q, ξ_r, τ_r, τ_q for a linear system $y' = Ay$. Here, $A = T\Lambda T^{-1}$, where $\Lambda = \mathrm{diag}(-100, -10, -1, -0.1)$, and T is a random 4×4 matrix, i.e. the elements are independent random numbers in the interval [0,1], and $y(0)$ is a random vector. Note the sharp plateaus, when $\tau_q(t)$ has

Figure 13.2.2. *The functions ξ_q, ξ_r, τ_r, τ_q for $p = 2$, and $p = 5$.*

reached the reciprocals of the eigenvalues of $-A$, also called the **time constants of the system**. This happens, approximately at $t = 0, 0.1, 1, 10$, a little earlier for $p = 2$ than for $p = 5$.

The curves for $\tau_r(t)$ show a smoother transition between the levels. This is quite natural, since $\tau_r(t)$ is the geometric average of the first p functions $\tau_q(t)$.

Another interesting feature of this example is that *the local time scale of the motion is much larger than the smallest time constant of the system, when t is large*. They become, respectively, 10 and 0.01. When this happens one says that *the motion has become* **stiff**. *A problem is said to be stiff*, if there is a time interval, where the motion is stiff.

The increase of the local time scale is explained by the following decomposition of the solution $y(t)$. Here t_j is a column of the matrix T, i.e. the eigenvector of A belonging to the eigenvalue λ_j, and γ_j is a component of the vector $T^{-1}y(0)$, $j = 1, 2, \ldots, s$, $(s = 4)$. By (13.1.13) and (13.1.16), $y(t) = \exp(At)y(0) = T\exp(\Lambda t)T^{-1}y(0)$, and

$$y(t) = \sum_{j=1}^{s} \gamma_j t_j \exp(\lambda_j t) \tag{13.2.14}$$

The term with $\lambda_j = -100$ dies out at about $t = 0.05$ (say). Similarly, the term with $\lambda_j = -10$ dies out at about $t = 0.5$, etc.. Roughly speaking, the local time scale is determined by the fastest of those terms which have not yet died out.

The decomposition (13.2.14) is *valid and interesting for general diagonalizable matrices*, and our discussion can be extended also to ODEs with complex eigenvalues. In that case, either the real part or the modulus of an eigenvalue is relevant, depending on the context.

A linear system with constant coefficients was considered in the example.

The concepts introduced are most relevant in discussions of what happens to small disturbances (errors). Therefore the natural generalization of the concept of time constants to a non-linear problem is related to the variational equation (13.1.7). Natural candidates to the name of time constants at a given time are then *the reciprocals of the eigenvalues of the Jacobian.* The smallest time constant can often be estimated by the reciprocal of some norm of the Jacobian, which is often time-dependent. So, we must talk about a **local time constant** in the definition of *stiffness in a non-linear problem.* This is a useful generalization, although the usage of this terminology must be taken with a grain of salt, unless the time scale of the variation of the local time constants is much larger than the time constants themselves.

Stiff problems is an important class of problems, that requires special numerical methods; else the time step will become much shorter than the one predicted by Theorem 13.2.2. A stiff motion is namely surrounded by other solutions to the same system, for which the local time scale is more like the *smallest* time constant. The numerical method must have good stability properties, in order that the motion should not be kicked out to one of those tracks, where a much smaller step size is required. We shall return to this question, first in Example 13.2.4.

The curves of the previous example were obtained by computer, with the use of the exact solutions of the ODEs. Simple examples that can be treated by analytic methods also provide insight about how the theoretical step control works, and about the relation of the global error to the tolerance in different situations. In the following two examples, we set the safety factor $\beta = 1$, and deal with the exact derivatives of the solution along the exact motion.

Example 13.2.2.

The differential equation,

$$y' = -my^{1+1/m}, \qquad y(1) = 1, \qquad (m > 0),$$

has the solution, $y(t) = t^{-m}$. Note that

$$(-1)^p y^{(p)}(t) = m(m+1)(m+2)...(m+p-1)t^{-m-p}, \tag{13.2.15}$$

so that

$$\tau_q(t) = t/b_q, \qquad b_q = p + m,$$
$$\tau_r(t) = t/b_r, \qquad b_r = \Big(m(m+1)(m+2)...(m+p-1)\Big)^{1/p}.$$

Hence $\xi(t) = b\ln(t)$, with $b = b_q$ or $b = b_r$. In this example, $\tau_r(t)/\tau_q(t)$ is constant, so the scaled local error will be constant during the motion. (Actually, $2 < b_q/b_r < 3$, for $0.5 < m < 1$.) By Theorem 13.2.2,

$$h(t) \approx kt/b_r, \qquad N(t) \approx b_r \ln(t)/k, \qquad k = |TOL/c_p|^{1/p}.$$

Notice that the step size grows proportionally to time. The values of p, c_p and k depend on the particular numerical method. We have

$$\mu(J(t)) = f'(y(t)) = -(m+1)y^{1/m} = -(m+1)/t.$$

We request relative accuracy, i.e. $S(t) = y(t) = t^{-m}$. Since $S^{-1}JS = J$, we then obtain,

$$J - S^{-1}\frac{dS}{dt} = -\frac{m+1}{t} + \frac{m}{t} = -\frac{1}{t}.$$

Then, by (13.2.7), we can choose $\mu^* = \max(-\tau_q(t)/t) = -1/b_q$. By Theorem 13.2.1, or Fig. 13.2.1, the estimated norm of the *relative* global error, $v(\xi)$, will grow like ξTOL, to begin with, (if there is no initial error), and then it converges towards $b_q TOL$. Hence the *absolute* error bound tends to zero like $b_q t^{-m} TOL$.

With constant step size $h = h(1)$ the number of steps needed are to reach t is hence $(t-1)/\ln(t)$ times as large as with our theoretical step control strategy. For example, for $t = 10^4$, wehave$(t-1)/\ln(t) \approx 10^3$. Note that this conclusion is independent of m, p, c_p and TOL.

The case with $m = \frac{1}{2}$, i.e. $y' = -\frac{1}{2}y^3$ with exact solution $y(t) = t^{-1/2}$, was run for three different values of TOL, with a program, using Runge's 2nd order method ($p = 2$) is used with $c_p = 1/6$. We set $TOL' = TOL/1.2$, where 1.2 is the safety factor used in the method. The row named "theor.", is calculated according to formulas above.

TOL	rel.err$/TOL'$	$N(10^4)$	$N(10^4)\sqrt{TOL'}$
$4 \cdot 10^{-4}$	1.59	196	3.53
10^{-4}	1.67	377	3.39
$\frac{1}{4} \cdot 10^{-4}$	1.70	740	3.33
theor.	2.50		3.26

The deviation of the values of rel.err$/TOL'$ from 2.5 is mainly due to the simplified error estimate used in the program. In this example the error is smaller than expected. In other problems it can instead be a few times larger than expected.

Example 13.2.3.

Consider the differential equation,

$$y' = \lambda y, \qquad y(0) = 1, \qquad (\lambda \in \mathbf{R}).$$

The solution is $y(t) = \exp(\lambda t)$, and hence $y^{(p)}(t) = \lambda^p y$. Set $S(t) = \max(y(t), 0.001)$. As long as $\lambda t > \ln(0.001) = -6.9$, we have $y(t) > 0.001$. By (13.2.11) and (13.2.7),

$$S(t) = y(t), \quad \tau_q(t) = \tau_r(t) = 1/|\lambda|, \quad \xi = |\lambda t|, \quad \mu^* = |\lambda|^{-1}(\lambda - \lambda) = 0.$$

Then, by Theorem 13.2.1, the relative error does not exceed $g(\xi)$, where $dg/d\xi = TOL, \quad g(0) = 0,$, i.e., $g(\xi) = TOL\,\xi = TOL|\lambda t|$, for any λ. Therefore,

$$|\text{Absolute error bound}| \approx TOL|\lambda t|\exp(\lambda t). \tag{13.2.16}$$

By Theorem 13.2.2, $|\lambda h| = |TOL/c_p|^{1/p}$, i.e., the step size is constant.

Nothing more is to be said about the case $\lambda \geq 0$. So, now *we assume that* $\lambda < 0$. By Eqn, (13.2.16), the bound for the *absolute* global error has a maximum

equal to TOL/e, when $\xi = |\lambda t| = 1$, and it decreases then towards $0.0069TOL$ at $\xi = 6.9$.

For $\xi > 6.9$, we have instead,

$$S(t) = 0.001, \quad \tau_q = 1/|\lambda|, \quad \mu^* = \lambda\tau_q = -1, \quad \tau_r = (0.001e^{\xi})^{1/p}/|\lambda|.$$

Now $g(\xi) =$ the *absolute* error bound divided by 0.001. (13.2.8) becomes,

$$dg/d\xi = -g + TOL.$$

So $g \to TOL$, hence the absolute error bound decreases towards $0.001TOL$. By Theorem 13.2.2, the theoretical step size increases rapidly:

$$|\lambda h| \approx |0.001e^{\xi}TOL/c_p|^{1/p}.$$

If we had continued to use the *relative* error in the step size control, i.e. $S(t) = y(t)$, the step size would have remained constant, and the absolute error bound would have converged towards zero, as $t \to \infty$ according to (13.2.16).

The step size sequence becomes completely different for the almost equivalent problem

$$y' = \lambda(y - 1), \quad y(0) = 0, \quad \lambda < 0,$$

with the solution $y(t) = 1 - e^{\lambda t}$, since the error relative to $1 - e^{\lambda t}$ is not the same as the error relative to $e^{\lambda t}$.

The equation $y' = -y$, $y(0) = 1$, $t \in [0, 20]$, was run with Runge's 2nd order method similarly to the previous example. Relative error control was used over the whole interval. As expected, this resulted in an almost constant stepsize. The agreement with the theoretical values is almost perfect in this example, since the local error estimate is almost perfect.

TOL	rel.err$/TOL'$	$N(20)$	$N(20)\sqrt{TOL'}$
$4\,10^{-4}$	19.8	465	8.37
10^{-4}	19.9	919	8.27
$\frac{1}{4}10^{-4}$	19.9	1827	8.22
theor.	20		8.16

13.2.2 Introduction to Numerical Stability

So far, we have discussed a theoretical strategy for the step size control. With a few modifications, this has been used in actual programs, and the actual step size sequences are often close to those predicted by theoretical calculations like the above (when they can be done). Some of the differences between this theory and practice are as follows:

A. Most programs put *restrictions on the time step changes.* (A simple case is seen in the program for Runge's 2nd order method outlined in Sec. 1.3.) There are several reasons. It was mentioned above that a rapid change of the

suggested step size may be due to a weakness of the error estimate. Another reason is that the change of the step size causes some extra computation ("overhead costs"), so it may sometimes be better to keep it unchanged for a while than to increase it by only a few per cent.

B. The local error of some numerical methods, e.g. the Runge-Kutta methods, is not given by (13.2.12) that is the basis of Theorem 13.2.2. Nevertheless, we shall in Sec. 13.6 consider the step size control of the 2nd order method introduced in Sec. 1.3 from this point of view. This is a simple example of a strategy recently applied to several so-called embedded Runge-Kutta methods: the step size control is here based on an estimate that is $O(h^p)$, i.e. the order of magnitude of the *global* error (one of the essential features of our strategy). Most old programs used an estimate of the *local* error, that is, $O(h^{p+1})$, in their step size control.

C. When $\|hf'(y)\|^2$ is not small, the error propagation in the numerical algorithm can be very different from the error propagation in the differential system. Therefore, the choice of step size is, and should be, influenced by the numerical stability properties of the numerical method. The rapid increase of the step size in Example 13.2.3 for large values of ξ, when $\lambda < 0$, must therefore be taken with a grain of salt. We shall see that *for Euler's method and other methods that are not designed to handle stiff problems*, see Example 13.2.1 above, *the step size must be bounded by some multiple of the smallest time constant of the system.* This multiple depends on the particular method. Sometimes this bound, which is independent of the tolerance, can be much lower than the step size predicted by Theorem 13.2.2. If one attempts to solve a stiff problem by a method that is not suited for them, the largest allowed step size will usually be found (approximately) as a result of the step size control, without any estimation of the time constants.

Example 13.2.4.

We shall study the behavior of Euler's method for a linear system, $y' = Ay$, $y(0) = c$, where A is a constant, diagonalizable $s \times s$ matrix. We obtain the difference equation,

$$y_{n+1} = (I + h_n A)y_n, \qquad y_0 = c. \tag{13.2.17}$$

Assume that A can be diagonalized by a well conditioned transformation T, i.e., $T^{-1}AT = \Lambda = \mathrm{diag}(\lambda_j)$. If we set $y = Tz$, and $d = T^{-1}c$, then the differential and difference equationsbecome, respectively

$$z' = \Lambda z, \qquad z(0) = d,$$

$$z_{n+1} = (I + h_n \Lambda)z_n, \qquad z_0 = d.$$

Each of these vector equations falls apart into s scalar equations:

$$w' = \lambda w, \qquad w_{n+1} = (1 + h_n \lambda)w_n, \tag{13.2.18}$$

where $\lambda \in \text{Spectrum}(A)$, and $w \in \mathbf{C}$, (since even a real matrix can have complex eigenvalues). The initial value is unimportant to our discussion. For the sake of simplicity, set$w(0) = w_0 = 1$. Note that $w(t)$ is proportional to the length of the component of $y(t)$ in the direction of the eigenvector belonging to the eigenvalue λ. Similarly for w_n.

Let us now restrict the discussion to the case where A has a real negative spectrum, e.g., the matrix of Example 13.2.1, where the spectrum is $\{-100, -10. -1. - 0.1\}$. Note that if $h_n\lambda < -1$, the sequence $\{w_n\}$ will have sign changes, in spite that the exact solution $w(t)$ has constant sign. It is even more remarkable that although $w(t)$ is exponentially decreasing, the inequality $|w_{n+1}| \leq |w_n|$, holds only if $h_n\lambda \geq -2$. Thus, *the criterion for numerical stability for Euler's method is, in the case of real negative eigenvalues, that*

$$h \leq h_{\text{stab}} = 2|\lambda_{\max}|^{-1}, \tag{13.2.19}$$

where $|\lambda_{\max}|^{-1}$ *is the smallest time constant of the system* If $h_n > h_{\text{stab}}$ for all $n > n_1$ (say), then $|w_n|$ grows exponentially, and after a few steps the sequence $\{y_n\}$ of Example 13.2.1 has very little in common with the exact solution $y(t)$.

Actually, *the same result happens to be valid also for Runge's 2nd order method.* (For other useful methods the coefficient 2 is to be replaced by some other value.) We shall later see that *there exist methods, where this coefficient is infinite*, i.e., the step size is not restricted by the risk for numerical instability. But there is a price for the use of them: they are **implicit**, i.e., the value of $f(y_{n+1})$ is requested in the step, where y_{n+1} is computed. So *a system of linear or non-linear equations is to be solved in every step.* This increases the volume of computation in a step considerably, unless the Jacobian matrix has special properties. In many cases, however, this is more than compensated for by an enormous reduction of the number of steps. More about this in Sec. 13.4.

With the automatic control of step size the solution is, however, not likely to grow indefinitely, due to numerical instability, even if the method is not designed to handle stiff problems efficiently. The step size control will instead restrict the step size. Exactly what happens, depends on fine details in the implementation of the control, for example how sensitive the approximate error estimate is to perturbations. We shall try to describe it in principle.

In the beginning, the step size is small. A smooth and accurate solution is produced. If the motion becomes smoother as time goes by, the step size increases, and it can happen that the stability criterion is violated during a few steps. The error then grows, although it remains (almost) on the tolerated level. The motion is therefore kicked out a little to one of those tracks, mentioned at the end of Example 13.2.1, where a smaller step size is requested. So, the step size is reduced, so that a smooth solution is produced again. Soon, the step size increases again etc.. The whole cycle is repeated again and again, so that *the step size fluctuates around the bound* h_{stab} *set by the stability criterion.* The results are still reliable. *Note that this is achieved by a well designed step size control, without any estimation of the time constants.* The only disaster that happens, if the automatic step size control is well designed, is that the time steps may become very short.

At the time of writing the phenomena just described are still the subject of a lively research, that has been timely reviewed by Hairer and Wanner [8, 1991], Section IV.2.

For Euler's method, $p = 1$, $c_p = 2$. Theorem 13.2.2 predicts the theoretical step size: $h_{\text{theo}} \approx 2TOL\tau_r$. The smallest time constant of the system of Example 13.2.1 equals 0.01, and the local time scale is $\tau_r \approx t$. Then, by (13.2.19), $h_{\text{stab}} = 0.02$. The stepsize is therefore bounded by the stability property of Euler's method, i.e., $h = h_{\text{stab}}$, when $h_{\text{stab}} < h_{\text{theo}}$, which in this case becomes $t > 0.01/TOL$. For Runge's 2nd order method, $h_{\text{theo}} \approx \sqrt{6TOL}\tau_r$ is larger, while h_{stab} is the same, and hence the restriction of h due to the stability bound comes earlier.

Example 13.2.5.

Our program for Runge's 2nd order method (essentially according to Sec. 1.4) was applied to the problem $y' = 1-y$, $y(0) = 0$, $t \in [0, 100]$ with $S = \max(0.001, |y|)$. For $TOL = 0.0016$ the first step size became $h = 0.01$. It then grew rapidly and at about $t \approx 12$ it became 2.7 that exceeds $h_{\text{stab}} = 2$. This shows that the "motion" has become stiff. The step size control then makes the the stepsize fluctuates between 1.8 and 2.2. The results are accurate enough. The relative error fluctuates about 20% around $1.2\,TOL$. The whole computation was done in 86 steps. Only 4 of them needed recomputation.

For smaller tolerances the behavior was similar, but the violation of the stability condition was smaller. The relative error was close to TOL all the time. For $TOL < 10^{-6}$ less than 0.5% of the steps were rejected.

A message given by these examples, is that we have to distinguish between intervals, where a motion is not stiff and the step size can be predicted by Theorem 13.2.2, and intervals where it is stiff, and thestep size may be restricted by the numerical stability properties of the method.

The reduction of the study of the linear system, $y' = Ay$, to the scalar equations, $w' = \lambda w$, $w \in \text{Spectrum}(A)$, which was done above for Euler's method can be done in the same way for most numerical methods for ODEs. The scalar equation,

$$y' = \lambda y, \qquad \lambda \in \mathbf{C}, \tag{13.2.20}$$

is therefore a widely used **test equation** for the study of the stability of numerical methods. It turns out that the solution of the difference equation produced by the numerical method depends only on the dimensionless parameter combination λh.

Definition 13.2.3.

The **stability region** *of a numerical method for the initial value problem for ODEs is the set of complex numbers $q = \lambda h$, such that the application of the method with a constant step size h to the test equation $y' = \lambda y$, produces a sequence $\{y_n(q)\}$ that is bounded as $n \to \infty$, for any set of initial data.*

Example 13.2.6.

Figure 13.2.3. *Upper half of the stability regions for Euler's method, Runge's 2nd order method and Kutta's Simpson's rule (4th order, see Sec. 13.3)*

By (13.2.18), the stability region of Euler's method is determined by the inequality, $|1+q| \leq 1$. It is therefore a disc of unit radius in the complex plane, with center at -1.

Example 13.2.7.

When Runge's 2nd order method is applied with constant step size h to the test equation $y' = \lambda y$, we obtain the difference equation

$$y_{n+1} = \Big(1 + q + \frac{1}{2}q^2\Big)y_n, \quad (q = \lambda h),$$

hence the stability region is defined by the inequality $|1+q+\frac{1}{2}q^2| \leq 1$. Its intersections with the real axis are at 0 and -2, just like Euler's method. The verification of these statements is left as an exercise. See Fig. 13.2.3.

Example 13.2.8.

The method defined by the formula

$$y_{n+1} = y_n + hf(y_{n+1}), \tag{13.2.21}$$

is called the **implicit Euler method** or the *backward Euler method*. It is only 1st order accurate, but it has interesting stability properties. For the test equation, we obtain the difference equation $y_{n+1} = y_n + qy_{n+1}$, i.e., $y_{n+1} = (1-q)^{-1}y_n$. The stability region is therefore determined by the inequality $|1-q| \geq 1$. It is therefore *the exterior of a disk of radius 1 and center at 1.* Note in particular that the whole negative half-plane belongs to the stability region. The step size is therefore not restricted by stability reasons, if the eigenvalues of the Jacobian have negative real parts.

On the other hand, S contains also a large part of the right half-plane. Therefore, if the step size is too large in a problem where the (exact) motion is unstable, the computations may not indicate any instability. This can in some applications be rather dangerous.

Definition 13.2.4.

A method is **consistent** *with $y' = f(y)$, if it integrates all differential equations of the form $y' = c = const.$ correctly, if the initial data are correct.*

For example, Euler's method and Runge's 2nd order method are consistent, since for $y' = c$, they produce $y_{n+1} = y_n + ch$. So if $y_n = y(0) + nh$ they produce $y_{n+1} = y(0) + (n+1)h$.

Below we give formal definitions of stability. Denote the stability region of a method by S. We discuss only methods that with a sufficient number of in-data to a step, produce a unique result (y_{n+1}), provided that the step size is small enough. (The latter precaution is needed for implicit methods.)

Definition 13.2.5.

A method is **zero-stable**, *if $0 \in S$. A method is* **strongly zero-stable** *if, for some $\delta > 0$, S contains the closed disk with radius δ and center at $-\delta$.*
Set $y_n(\infty) = \lim_{q\to\infty} y_n(q)$, assumed that the limit exists.
A method is **∞-stable** *if the sequence $\{y_n(\infty)\}$ is bounded as $n \to +\infty$.*
A method is **strongly ∞-stable** *if $\lim_{n\to+\infty} y_n(\infty) = 0$.*
A method is **A-stable** *if S includes the left half-plane $\{q : \Re q < 0\}$.*
A method is **$A(\alpha)$-stable**, *$0 < \alpha < \pi/2$, if S includes the sector $\{z : |\arg(-z)| < \alpha\}$.*

A method that is not zero-stable will, in floating point arithmetic, for almost all well-conditioned initial value problems, produce solutions which quickly become useless, no matter how the step size has been chosen. For several classes of methods, it can be shown that, as $h \to 0$, *the sequence $\{y_n\}$ converges* on a finite interval to the solution of the differential equation, for any differential system that satisfies very general conditions, *if and only if the method is consistent and zero-stable.* The "general conditions" are roughly the same as the conditions assumed in the existence and uniqueness Theorem given in the next section. Some results of this type will be derived in Secs. 13.4 and 13.5. For more complete results, we refer to the excellent monographs of Butcher [1, 1986] and Hairer et al. [7, 1993].

The four stability concepts mentioned last are useful in order to find methods which are useful for stiff problems.

It is important to realize that the background to the test problem and the stability region is a *system* of ODEs. Nobody is interested in integrating the test equation itself with $|\lambda h| > 1$ (say). These concepts give a helpful guidance in more general situations, e.g. *nonlinear systems* and *variable stepsize*, although they do not exactly provide "the truth, the whole truth and nothing but the truth" in these cases. Since small perturbations in nonlinear systems are propagated according to

the linearized variational equation, it is natural to substitute rough estimates of **the eigenvalues of the Jacobian** along the expected motions for λ in the test equation. For some classes of ODEs, inequalities are known for these eigenvalues, e.g., the Jacobian may be known or expected to be negative definite.

Finally, it is sometimes believed that numerical methods are useless, e.g., when the eigenvalues are positive, because λh is then outside the stability region. That is a misunderstanding. Most methods provide results with good *relative* accuracy, when applied to the equation $y' = y$, see Example 13.2.3. More generally, if a motion is sensitive to perturbations in the ODEs, the results obtained by the most popular numerical methods with a well designed step size control are usually no more sensitive to perturbations than the solutions to the ODEs are themselves.

There are, in fact, examples where the numerical results are less sensitive to perturbations than the exact solutions, and *that can be good or bad,* see Sec. 13.4. Note for example that the stability region of the implicit Euler method contains a substantial part of the right half-plane, where $\Re\lambda > 0$, and the absolute value of the solution of$y' = \lambda y$ tends to ∞ with t. So, this method may sometimes produce a smooth solution with no indication that the exact solution of the ODE system may be very sensitive to perturbations.

13.2.3 Investigation of Stability

In this section we give some algorithms and graphical methods for investigation of stability of numerical methods. At the application to the linear test equation,

$$y' = \lambda y, \quad y(0) = 1, \quad q = \lambda h = \text{const.} \in C, \tag{13.2.22}$$

see also Sec. 13.1.4, most numerical methods yield a difference equation, the characteristic polynomial of which is of the form,

$$\Psi(\zeta, q) = \psi_k(q)\zeta^k + \psi_{k-1}(q)\zeta^{k-1} + \ldots + \psi_1(q)\zeta + \psi_0(q). \tag{13.2.23}$$

where the $\psi_j(q)$ are real polynomials, the highest degree of which is m. The k characteristic roots are named $\zeta_i(q)$, $i = 1, 2, \ldots, k$. It happens that several different methods yield the same characteristic equation for this test problem.

By the implicit function theorem for analytic functions ζ and q are analytic functions (a conformal mapping) of each other in the neighborhood of every point (ζ_0, q_0) that satisfies (13.2.23), except at branch points, i.e. points where $\partial\Psi/\partial\zeta = 0$ or $\partial\Psi/\partial q = 0$. The first of these relations expresses that (13.2.23) has a multiple root, when it is considered as an equation for ζ for a given q. The roots $\zeta_i(q)$, $i = 1, 2, \ldots, k$ are branches of an analytic function. Two or more branches can meet, where $\partial\Psi/\partial\zeta = 0$. $\zeta_i(q)$ is continuous also at branch points, though it may not be differentiable there. Take for example $\Psi(\zeta, q) = \zeta^2 - q$. (Consider also the different behavior in the example $\Psi(\zeta, q) = \zeta^2 - q^2$.)

There are exceptional cases, where some of the above statements are not strictly true, for example, the points where $\psi_k(q) = 0$. We can avoid dealing with them as exceptional by considering the complex variables ζ and q as points on a **Riemann sphere** and hence consider ∞ as an ordinary point (the pole of a

stereographic projection from the sphere to the plane). Neighborhoods, distances, continuity etc. are to be considered on the sphere.

If $\psi_k(q) \to 0$ as $q \to q_1$ then $\zeta_i(q) \to \infty$ for at least one i. We then say that $\zeta_i(q_1) = \infty$. The multiplicity of this root is $k - k'$, if the degree of of $\Psi(\zeta, q)$ drops from k to $k - k'$, as $q \to q_1$. The use of the Riemann sphere is convenient in many other respects. It allows us, for example, to say that the function $\zeta = 1/q$, defined by the equation $q\zeta - 1 = 0$ is continuous everywhere, also at $q = 0$. Similarly, the continuity of the branches $\zeta_j(q)$ holds without exceptions on the Riemann sphere. (If you feel insecure about the handling of ∞, it may help to introduce $\hat{\zeta} = 1/\zeta$ and/or $\hat{q} = 1/q$ into the characteristic equation, and see what happens as $\hat{\zeta} \to 0$ and/or $\hat{q} \to 0$.)

What we have said about $\zeta(q)$ holds, mutatis mutandis, also for the inverse function. It has m branches denoted $q_j(\zeta)$, $j = 1, 2, \ldots, m$. The characteristic polynomial when $q = \infty$ consists of the terms of $\Psi(\zeta, q)$ which contain q^m.

We shall only consider *consistent* methods. Since $y(h) = e^q$, one of the roots, denoted $\zeta_1(q)$ and called the *principal root*, should therefore approximate e^q, when $|q|$ is small. We say that the **order of linear consistency** is $\bar{p}$, if

$$\zeta_1(q) - e^q \sim \bar{c} q^{\bar{p}+1}, \quad \bar{c} \neq 0, \quad q \to 0. \tag{13.2.24}$$

$\bar{c}$ is called the **linear error constant**. For most methods $\bar{p}$ is equal to the usual order of consistency p, defined in Sec. 13.2.1, e.g., for all linear multistep methods, but there are methods for which $p < \bar{p}$. For a consistent method $p \geq 1$, and hence $\bar{p} \geq 1$. It follows that

$$\zeta_1(0) = 1, \quad \zeta_1'(0) = 1. \tag{13.2.25}$$

The stability region S is the set of complex numbers q such that the root condition (Theorem 13.8.6) is satisfied by $\Psi(\zeta, q)$, considered as a polynomial in ζ with q as a parameter. S is symmetric about the real axis, since we obtain conjugate values for q for conjugate values of ζ.

We saw in Example 13.2.8 that the stability region of the implicit Euler method is the exterior of a bounded region (actually a disk). This can also be expressed by saying that ∞ belongs to the interior of S. An example, where ∞ is a boundary point of S, is the θ-method for $\theta = \frac{1}{2}$, for which S is the half-plane $\Re q \leq 0$, see Problem 13.2.9. The term ∞-stable introduced in Sec. 13.1.4 is an expression for this point of view.

For an explicit method the degree of $\psi_k(q)$ is less than m. Then, for $q = \infty$ at least one of the roots of $\Psi(\zeta, q)$ is infinite. It follows that an explicit method of the class considered cannot be ∞-stable. In other words: *The stability region for an explicit method is a bounded set in* $\mathbf{C}$.

It follows from continuity considerations that *a point q on the boundary ∂S belongs to S, unless the characteristic equation has, for this value of q, a multiple zero of unit modulus.* One can show that the multiplicity cannot exceed 2 when $q \in \partial S$ (but it can be higher at other parts of the boundary locus), and that the presence of a double root is visible as a **cusp** on the boundary ∂S, i.e., *a point from which at most one ray points into S*, see Fig. 13.2.4a *Corners* of ∂S belong, however, to S, see Fig. 13.8.3b. So, *S is a closed set on the Riemann sphere, if*

its boundary has no cusps. This has some nice consequences. For example, in Sec. 13.1.4 a method is defined to be A-stable if S includes the *open* left half-plane $\{q : \Re q < 0\}$. If there were a boundary cusp on the imaginary axis, then there must also be points in the open left half plane that do not belong to S. The conclusion is that *for an A-stable method, S includes the imaginary axis too.*

A zero $\zeta_j(q)$ of $\Psi(\zeta, q)$ (for a fixed q) is called an *unstable root* if $|\zeta_j(q)| > 1$.

Figure 13.2.4. *(a) The boundary locus of a 10-step method.*

The map to the q-plane of the unit circle $\{\zeta = e^{i\phi} : \ 0 \leq \phi < 2\pi\}$ of the ζ-plane is named the **boundary locus** or root locus. It consists of one or more (at most m) *curves that divide the $q-$plane into several parts* (e.g., six parts in Fig. 13.2.4a that shows the boundary locus of a certain 10-step method). Note the important rule that *the number of unstable roots, counted with their multiplicity and including infinite roots, is constant within each part.* This follows from the continuity of $\zeta_j(q)$. (These integers are shown in Fig. 13.2.4a.) S is the small area marked with the digit 0. Notice that the boundary ∂S is composed by only a few pieces of the boundary locus.

Another important rule is due to the fact that, on the "microscopic scale", the orientation is conserved at the conformal mapping from the ζ-plane to the q-plane. We therefore look upon the boundary locus as a motion in the q-plane, along every branch of $q(\zeta)$, (to be marked by arrows in a plot) generated by a counter-clockwise motion along the unit circle in the ζ-plane. The neighborhoods of a short arc of the unit circle outside and inside the circle are then mapped into, respectively, the right and the left neighborhood of the corresponding arc of the boundary locus. *If q is moved from the right to the left of an arc of the boundary locus, the corresponding move in the ζ-plane tells that the number of unstable roots is decreased by one.* It is not necessary to watch the plotting process in order to set the arrows correctly. For a consistent method it follows from (13.2.24) that at least one branch of the

boundary locus will, at the beginning, move from $q = 0$ upwards, closely to the positive imaginary axis. In most cases this is enough for a correct setting of the arrows along the whole boundary locus (check the arrows in Fig. 13.2.4a).

There is a hidden assumption in the last paragraph: *We assume that a small arc of the boundary locus corresponds to it only one small arc of the unit circle.* We continue on this assumption, because it has an important practical consequence:

If we know the number of unstable roots for **one** *value of q only, then we can, by the application of the rule of the previous paragraph, and successively obtain the number of unstable roots for all the parts that the complex plane has been divided into by the boundary locus.* (For the 10-step method of Fig. 13.2.4a it has been found that there are 3 unstable roots. Starting from this fact, check the markings in the figure!) In particular: *the stability region S is easily found*; it is the union of the parts, where the number of unstable roots is equal to zero, (In Fig. 13.2.4a there one such part only.) It can happen that there is no such part; *S can be empty.* S can also, for example, *degenerate* into a line or a curve segment (see the next example), or into a point, Hairer and Wanner [1991, p.263].

There are, however, exceptions from this, e.g. if $F(\zeta, q)$ contains only even powers of ζ. The boundary locus is the circumscribed twice, hence he number of unstable roots decreases by 2 instead of 1, when the locus is passed from the right to the left. A less obvious exceptional case of the same type is found in Problem 20a. Next example illustrates that it can also happen that the boundary locus is traversed back and forth; the number of unstable roots is then the same on both sides. We *conjecture* that the most general exceptional case are combinations of these types, where the whole of the boundary locus (or at least each unicursal closed part of it) is traversed the same number of times (where backwards is negative). If this is true, it is enough to *check the number of unstable roots for one more value of q*, in a different part, in order to determine the number of unstable roots in the whole q-plane. (In the case of Fig. 13.2.4a, it was checked that there are 3 unstable roots for $q = -1$, so if we believe the conjecture, we conclude that the marking is correct.)

Example 13.2.9.

The two-step method $y_{n+2} - y_n = 2hf(y_{n+1})$ is named the **leap-frog method** *or the explicit midpoint method. For the test equation we obtain the difference equation $y_{n+2} - y_n = 2qy_{n+1}$ with the characteristic equation $\zeta^2 - 2q\zeta - 1 = 0$, hence $q = \frac{1}{2}(\zeta - 1/\zeta)$. For $\zeta = e^{i\phi}$, we obtain $q = i \sin\phi$, hence the boundary locus degenerates to the closed line segment from $-i$ to $+i$, traversed up and down. For $q = \infty$ the characteristic equation reads $\zeta = 0$. One root is 0, the other is ∞, hence the region outside this line segment is not in S. For $q = i \sin\phi$, the roots of the characteristic equation are $e^{i\phi}$ and $e^{-i\phi}$, which are simple if $\phi \neq \pi/2$ and $\phi \neq 3\pi/2$, i.e if $q \neq \pm i$. S* is therefore the open line segment from $-i$ to i. *The endpoints are cusps (of an unusual kind), but the points between are not cusps, because* two *directions from them (up and down) lead into S.*

A more typical kind of cusp is shown in Fig. 13.2.4b for

$$\Psi(\zeta, q) = (\zeta - 1)(\zeta + 1)^2 - 4\zeta^3 q.$$

The cusp at $q = 0$ is generated by a double root at $\zeta = -1$. In this example S is the

outer region. So, from the cusp only one direction points into S.

We shall now limit the discussion to the simplest cases, where $\Psi(\zeta, q)$ is either an affine function of q or an affine function of ζ. The linear multistep methods belong to the former category; the Runge-Kutta methods belong to the latter.

The **Runge-Kutta methods**, which will be studied more deeply in Sec. 13.3, yield at the application to the test equation $y' = \lambda y$ a difference equation of the form $y_{n+1} = R(q)y_n$, where $R(q)$ is a *rational function*, $R(q) = F(q)/G(q)$, where the polynomials F, G should have no common divisor. Hence we may write $\Psi(\zeta, q) = F(q) - G(q)\zeta$. Here F is a mth degree polynomial, while the degree of G is at most m. (m is the number of stages of the method.) The classical Runge-Kutta methods are *explicit*. In this case, $G(q) \equiv 1$, i.e., $R(q)$ is a polynomial. Fig. 13.2.5

Figure 13.2.5. *Boundary locus of two Runge-Kutta methods.*

show the boundary locus of two widely used Runge-Kutta methods. The former is Kutta's Simpson's rule, also called the classical fourth order Runge-Kutta method, $m = p = 4$,

$$\zeta = \sum_{j=0}^{4} q^j/j! = e^q - q^5/120 - \dots.$$

The latter is called Dopri5 (see Sec. 13.3), $m = 6$, $p = 5$, and

$$\zeta = \sum_{j=0}^{5} q^j/j! \ + q^6/600 = e^q + q^6/3600 + \dots.$$

In order to plot the boundary locus in the general case, *all* roots of the mth degree algebraic equation $F(q) - G(q)\zeta = 0$ are to be computed for $\zeta = e^{2\pi ij/N}$, $j = 0, 1, \dots N$ for some suitable value of N.

If we use a program that returns all roots of an algebraic equation, without the use a first approximation suggested by the user, the ordering of the roots must

be inspected for every ζ and perhaps changed, otherwise the plot of the boundary locus can become rather strange.

The following alternative approach has the advantage that it is not necessary to know the coefficients of F and G. We shall see, in Sec. 13.6, that the numerical value of $R(q)$ is easily computed directly from the coefficients which define the method. The algorithm starts a "trip" at $\zeta = 1$, $q = 0$, and follows a "quasi-continuous" variation of a root of the equation $R(q) = \zeta$, i.e., one value of q is determined for $\zeta = e^{2\pi ij/N}$, $j = 0, 1, \ldots mN - 1$, using e.g. the secant method with the previous values of q (or something better) as initial guesses. (A special rule is needed at the first point.) Note that the unit circle will be traversed m times during the "trip". We will obtain m branches, and if we are lucky, they are all different, and we have obtained the boundary locus, Fig. 13.2.5a.

It happens, however, that we do not find m different branches this way. For example, in Fig. 13.2.5b the boundary locus consists of three separate curves. There is no chance to find the small curves with $q = 0$ as a starting point. The algorithm therefore must record the computed roots of the equation $R(q) = 1$. If less than m different roots have been obtained (within a tolerance), all roots of the equation $R(q) = 1$ must be computed, and the algorithm has to make a new "trip" (or more), starting from one of the remaining roots. We refrain from a discussion of the multiple root case.

In Fig. 13.2.5a and Fig. 13.2.5b we can see the map of the unit circle, with, respectively, the 4 and the 6 different starting points, because different line type has been used. If you want a better grasp of of the two algorithms, try to find out what would happen if $\Psi(\zeta, q) = \zeta - q^2$, (even though this example does not correspond to any consistent numerical method). A more satisfactory discussion of these matters would require the concept of a Riemann surface, but that is beyond the scope of this book.

For the Runge-Kutta methods it is easy to find the stability region, in a plot of the boundary locus. Since the characteristic equation is linear in ζ, the number of unstable roots (ζ for a given q) can be 0 or 1 only. Moreover, the boundary locus cannot intersect itself. (It can have a sort of double cusp, at a point where $\partial\Psi/\partial q = 0$. It is easier to find examples of this, if one plots more general level curves, $|\zeta| = r$, for a method. This is sometimes of practical interest.)

In Fig. 13.2.5a, S is marked by a zero; it is the interior of the closed curve. In Fig. 13.2.5b, S is the union of the interior of the three closed curves. A Runge-Kutta method is always strongly zero-stable; the origin and the area ust to the left of it belong to S. S can be unbounded, if $R(q)$ is not a polynomial.

A **general linear multistep** method for the differential system $y' = f(y)$, $y(0) = y_0$, is defined by the difference equation

$$\sum_{i=0}^{k} \left(\alpha_i y_{n+i} - h\beta_i f(y_{n+i})\right) = 0, \tag{13.2.26}$$

where α_i and β_i are real parameters, h the step length. The formula (13.2.26) is also called a *linear k-step method.* We shall study this class of methods more thoroughly in Sec. 13.4, together with another class of methods, named **one-leg**

k-step methods, defined by a similar difference equation:

$$\sum_{i=0}^{k} \alpha_i y_{n+i} \; - \; hf\Big(\sum_{i=0}^{k} \beta_i y_{n+i}\Big) = 0. \tag{13.2.27}$$

The **generating polynomials**

$$\rho(\zeta) = \sum_{i=0}^{k} \alpha_i \zeta^i, \qquad \sigma(\zeta) = \sum_{i=0}^{k} \beta_i \zeta^i, \tag{13.2.28}$$

play a fundamental role in the theory of multistep methods. We shall always *assume that the polynomials ρ and σ have no common factors.* For the standard test problem $y' = \lambda y,\ y(0) = 1,\ \lambda h = q$, the linear multistep method and the one-leg method with the same coefficients yield identical difference equations. The characteristic equation reads,

$$\rho(\zeta) - q\sigma(\zeta) = 0, \text{ hence } q = \rho(\zeta)/\sigma(\zeta). \tag{13.2.29}$$

For $q = \infty$ the characteristic equation reads $\sigma(\zeta) = 0$. The boundary locus becomes a single curve that is easily plotted after the computation of $q(\zeta) = \rho(\zeta)/\sigma(\zeta)$ for $\zeta = e^{2\pi ij/N}$, $j = 0, 1, \ldots N$ for some suitable value of N. So, q is a rational function of ζ, (We saw above that it is the other way around for the Runge-Kutta methods; for them ζ is a rational function of q.)

The map of the unit disk by this rational function is usually much larger than S, because it contains all q such that $\rho(\zeta) - q\sigma(\zeta)$ has *at least one* zero in the unit disk, (find this set in Fig. 13.2.5a) while the condition $q \in S$ requires that *all* zeros have to be located there. The correct expression for S in terms of $\rho(\zeta)/\sigma(\zeta)$ is instead as follows:

Theorem 13.2.6.

The complement of the closed unit disk is mapped onto the interior of the complement of S by the rational function $q = \rho(\zeta)/\sigma(\zeta)$.

Review Questions

1. Describe the theoretical step size strategy of the text, its concepts, assumptions and results, in particular Fig 12.1.9.
2. What is, in this text, meant by the time constants of a system, the local time scale of a motion, a stiff motion and a stiff problem? Consider also a non-linear problem. Give an example of a stiff problem.
3. Define the stability region of a numerical method for the initial value problem for ODEs. How is the study of the numerical solution of $y' = Ay$, (under a certain condition) reduced to the study of the scalar test equation. (It is sufficient to explain it for Runge's 2nd order method).
4. Define zero-stability, strong zero-stability, $\infty-$stability, strong $\infty-$stability, A-stability, $A(\alpha)$-stability.

5. Tell what is likely to happen to the step size variation, when Euler's method, or some other method with a bounded stability region, is applied to a stiff problem with our theoretical step control strategy . You may assume that the Jacobian is negative definite.
6. Describe Runge's 2nd order method and the implicit Euler method. Give the formulas for their stability regions, and sketch the regions. Are the methods (strongly) zero-stable, (strongly) $\infty-$stable, A-stable ?

Problems

1. (a) Is it true that the basic formulas for Runge's 2nd order method, given for the non-autonomous system $y' = f(t,y)$, $y(0)$ given, yield the same results as the formula you obtain, when you apply the method to the autonomous system $y_1' = 1$, $y_2' = f(y_1, y_2)$, $y_1(0) = 0$, $y_2(0) = y(0)$? Motivate your answer.
2. (a) Show that the application of Runge's 2nd order method to the system $y' = Ay$ yields the formula $y_{n+1} = (1 + Ah + \frac{1}{2}A^2h^2)y_n$.
   (b) Consider the inhomogeneous scalar problem

$$y' = \lambda y + (\alpha - \lambda)e^{\alpha t}, \quad y(0) \text{ given}, \tag{13.2.30}$$

   and the system $z' = Az, \quad z(0) = (1,\ y(0))^T$, with

$$A = \begin{pmatrix} \alpha & 0 \\ \alpha - \lambda & \lambda \end{pmatrix}.$$

   Show that $z_2(t) = y(t)$, but that, if $\alpha \neq \lambda$, Runge's 2nd order method yield different results in the two problems, already in the first step, (even if $y(0) = 1$).
   (c) (13.2.30), $y(0) = 1$, with the solution $y(t) = e^{\alpha t}$ is a useful test problem for the study of numerical methods. Now we shall use it for the study of Runge's 2nd order method with constant step size. Set $p = \alpha h$, $q = \lambda h$, $t_0 = 0$, $t = t_n = nh$, and notice that $pn = \alpha t$. Show that

$$y_{n+1} = (1 + q + q^2/2)y_n + (p - q)(q/2 + e^{p/2})e^{pn},$$

   and verify that the only solution of this recurrence relation with the initial condition $y_0 = 1$ reads

$$y_n = be^{pn} + (1 - b)(1 + q + q^2/2)^n,$$

   where

$$b = \frac{(e^{p/2} + q/2) \cdot (p - q)}{e^p - (1 + q + q^2/2)}.$$

   Finally, assume that $|p| \ll 1$, $|q| \ll 1$, $\lambda \neq \alpha$, and show that *the global relative error* equals

$$(b - 1)(1 - (1 + q + q^2/2)^n e^{-pn}) \approx \frac{\alpha^2 h^2}{24} \frac{(3\lambda + \alpha)(1 - e^{(\lambda - \alpha)t})}{\lambda - \alpha}.$$

Hints: You cannot neglect $O(h^3)$-terms in the numerator of $b-1$. Part of this becomes easier if you use (13.2.32) below.

(d) We shall now investigate the rule for the choice of step size. Assume that λ and α are real and that $\lambda \leq \alpha$. Show that $k_2 - k_1 \approx \frac{1}{2}(\alpha h)^2 e^{\alpha t}$, and hence $\frac{1}{6}(\alpha h)^2 \approx TOL$. Put this into the final result of (c). At last, show that, in the notation of Theorem 13.2.1, the ratio of this result to the bound obtained in (13.2.9) is $|1 + \frac{3}{4}\mu^* \mathrm{sgn}(\alpha)|$, $(\alpha \neq 0)$. (See Computer exercise 6.)

3. Consider the linear system $y' = Ay - r$, where A and r are constant, and A is non-singular. Note that $A^{-1}r$ is a critical point, and show that *every motion converges to this point, as* $t \to \infty$, *if* $\mu(A) < 0$. Use this to show that

$$\|A^{-1}\| \leq |\mu(A)|^{-1}, \qquad \text{if} \quad \mu(A) < 0. \tag{13.2.31}$$

4. (a) Show that the relations

$$N'(t) \approx \frac{1}{h(t)}, \qquad \frac{h_{n+1} - h_n}{h_n} \approx h'(t_n),$$

are valid for any reasonably smooth step size sequence.

(b) Show that if $h(t) \approx pt + q$ for $t \in [a, b]$, then

$$N(b) - N(a) \approx \frac{\ln(b + q/p) - \ln(a + q/p)}{p}.$$

How is this simplified when $q/p \ll a < b$?

(c) Suppose that $h(t) \approx 0.1t$ for $10^{-5} \leq t \leq 10^5$. How many per cent does the step size increase from a step to the next? Estimate the total number of steps for the whole computation.

(d) Assume that $h(t) \approx k\tau(t)$, where $\tau(t)$ is smooth, and k is independent of t. Show that $(h_{n+1} - h_n)/h_n^2 \approx \tau'(t)/\tau(t)$. Note that the right hand side is independent of TOL, but it may depend strongly on t. A value of k is given in Theorem 13.2.2 for the theoretical control strategy, but the result just derived holds under more general conditions.

6. Derive the formula given in Sec. 13.2.3, according to which the norm of the scaled local error is proportional to $\tau_r(t)/\tau_q(t)$ during a motion (if the order p is constant). Show also that $\tau_r(t)/\tau_q(t)$ is constant if $y(t)$ is an exponential or a power function.

7. Derive the expression, given in Example 13.2.7, for the stability region of Runge's 2nd order method.

(a) Show how the discussion of the linear problem $y' = Ay$, where A is a constant diagonalizable matrix, can be reduced to the study of the scalar test equation (13.2.20).

(b) In Fig. 13.2.3 the stability region seems to be symmetric around the line $\Re q = -1$. Show that it is so, and determine the exact coordinates of the top of the stability region.

8. Let A be the tridiagonal matrix of Example 10.3.4.

(a) Show in two ways that Euler's method produces bounded solutions to the system,

$$\frac{dy}{dt} = -cAy \qquad (c > 0),$$

if $0 < hc \leq \frac{1}{2}$. The first way is to apply Gershgorin's Theorem to find an interval that contains the spectrum of A, and use this to show that the spectrum of $-hcA$ lies in the stability region of Euler's method for all $hc \in [0, \frac{1}{2}]$. In the second way you first show that $\|I - hcA\|_\infty \leq 1$, for all $hc \in [0, \frac{1}{2}]$.

(b) Is the same true for Runge's 2nd order method?

(c) Show that all solutions of the ODE system are bounded, as $t \to \infty$, by the use of $\mu_\infty(A)$.

(d) Show that all solutions of ODE system tend to zero, as $t \to \infty$, by the use of the knowledge about the spectrum of A.

9. The *θ-method* is a one parameter family of methods defined by the equation,

$$y_{n+1} - y_n = h\big((1-\theta)f(y_{n+1}) + \theta f(y_n)\big), \quad 0 \leq \theta \leq 1.$$

Show that the stability region is a disk if $\theta > \frac{1}{2}$, a half-plane if $\theta = \frac{1}{2}$ and the exterior of a disk if $\theta < \frac{1}{2}$. Show also that the method is 2nd order accurate for the test equation $y' = \lambda y$, if $\theta = \frac{1}{2}$, and only 1st order accurate for other values of θ. (We shall later see that the latter results hold for any differential system.)

10. (a) For the problem of Example 13.2.2, i.e. $y' = -my^{1+1/m}$, $y(1) = 1$, $(m > 0)$, show that if the relative error is considered all the time, then $|h(t)f'(y(t))| \leq (1 + 1/m)k \ \forall t$, where $k = |TOL/c_p|^{1/p}$. So, although the step size grows proportionally to time, *this problem does not require a method designed for stiff problems*. (In the past, there was a common misconception that the presence of different time scales during a motion makes the problem stiff, but it is rather when the *local* time scale becomes very much larger than the smallest *local* time constant, the motion becomes stiff.)

(b) For the problem of Example 13.2.3, i.e. $y' = \lambda y$, $y(0) = 1$, $(\lambda \in \mathbf{R})$, with the mixed absolute-relative error strategy, show that h reaches the stability limit for Runge's 2nd order method when $y(t) \approx \frac{3}{2} 0.001 TOL$.

11. In the study of numerical methods, one often encounters estimations of the following type. Let

$$\phi(q) = aq^{p+1} + bq^{p+2} + O(q^{p+3}), \quad |q| \ll 1, \quad nq = z = O(1).$$

(a) Show that $(e^q - \phi(q))^n = e^{q'n}$ where

$$q' = q - (1-q)\phi(q) + O(q^{p+3}), \quad (|q| \ll 1).$$

(b) Show that

$$(e^q - \phi(q))^n - e^z \approx -ze^z\Big(aq^p + (b - a + cza^2/2)q^{p+1} + ...\Big), \qquad (13.2.32)$$

where $c = 0$ for $p > 1$, and $c = 1$ for $p = 1$. For example, when the test equation $y' = \lambda y$, is used in the study of a p'th order method, one sets $q = \lambda h$, $z = \lambda t$. The result is also valid if λ is a matrix, a, b are scalars, and $\|q\| \ll 1$.
(c) Apply this result to *"the circle test"* i.e. the test problem $y' = iy$, $y(0) = 1$, step size h. The orbit is the unit circle of the complex plane. (See also Prob. 1.3.3.) Show that the global error for Runge's 2nd order method is approximately $te^{it}(ih^2/6 + h^3/8 + \ldots)$, and that the step size with the theoretical strategy becomes constant, $h = \sqrt{6TOL}$. Note that the radial error is positive and an order of magnitude smaller than the tangential error. How is TOL to be chosen in order that the error should be less than 10^{-3} after 10 revolutions? Suppose that the orbit is plotted with a straight line segment for each step. At what value of t does the global error become larger than the (local) interpolation error of the plotting?
Also show that he global error for Euler's method is

$$te^{it}(h/2 + (t/8 - i/3)h^2 + \ldots),$$

so that, for this method, the radial error is positive and much larger than the tangential error. Also show that $h = 2TOL$ with the theoretical strategy.

Computer Execrcises

1. We saw in Problem 5 above that the solution of the differential equation $y' = y - 2t/y$, $y(0) = 1$ is very sensitive to perturbations in the initial value. Study what happens in the numerical solution (with the correct initial value) up to $t = 12$ (or longer) with various tolerances that give decent computing times. Design a suitable termination criterion. Plot y versus t with several tolerances on one sheet, in linear scales. Plot on another sheet $\log|\text{error}|$ versus t.
2. Run the inhomogeneous test equation,

$$y' = \lambda y + (\alpha - \lambda)e^{\alpha t}, \quad t \in [0, 3], \quad TOL = 10^{-3},$$

and compare the actual global error divided by TOL with the bounds obtained in Theorem 13.2.2 and Problem 3. Take $\lambda = -2$, $\alpha = -2, -0.5, +0.5$, $y(0) = 1$, and make also one run with $y(0) = 0$.
3. A generalization of the classical dog curve problem. A dog chases a rabbit. The dog is smarter than usual, for it looks ahead of the dog by an angle equal to α radians. The speed of the rabbit is 1; the speed of the dog is $b > 1$. The motions can be described in the complex plane as follows. Let $z_D(t), z_R(t)$ be the positions at the time t of, respectively, the dog and the rabbit; $z_R(t) = (1 + a/16) + it$, i.e. the rabbit runs along a straight line. For the dog we obtain the differential equation:

$$z_D' = be^{i\alpha}\frac{z_R - z_D}{|z_R - z_D|}, \quad z_D(0) = a/16.$$

If necessary this can be replaced by two real equations. A chasing is terminated, e.g. when $|z_D - z_R| < 2TOL$ or $t = t_{end}$. Run five cases, $\alpha = 0.8 - 0.2a$, $a = 0,\ 1,\ 2,\ 3,\ 4$, and plot them on the same sheet. Try different values of b.

Note: A variant is to write $z_R = z_D + re^{i\phi}$, and solve the real differential equations for r, ϕ, or a single equation with either r or ϕ as the independent variable.

4. (Arenstorf orbits; a restricted 3-body problem of Astronomy.) The following 2nd order *complex* equation is a simple model for the motion of a satellite of negligible mass in the gravitational field of the earth (mass=$1-m$) and the moon (mass=m). The earth E and the moon M rotate in a circle around their mass center. The motion takes place in the plane of this circle. It is studied in a coordinate system that rotates with the earth and the moon. The location at time t of the satellite is described by a complex number, $z(t) = x(t) + iy(t)$. $E = -m$, $M = 1 - m$, (hence the mass center is at the origin.)

The satellite is therefore influenced by a centrifugal force and a Coriolis force in addition to the gravitational forces from the earth and the moon. The equation reads, in dimensionless form, (after a scaling of space and time):

$$z'' = z - 2iz' - (1-m)(z-E)|z-E|^{-3} - m(z-M)|z-M|^{-3}.$$

We choose (according to Hairer et al. [1987,p.128 and 197], $m = 0.012277471$,

$$z(0) = 0.994, \quad z'(0) = -i(2.0015851063790825 + 0.0301475231782543a).$$

For $a = 0$ and $a = 1$, the orbits should become (very different) closed curves. For $0 < a < 1$ the motion looks rather chaotic. The motion is, however, very sensitive to perturbations. Run it with $TOL = 10^{-3}$, 10^{-4}, until $t = 18$, for $a = 0$, $a = 1$, and some value between. Then take $a = 0$, with tolerances 10^{-3}, 10^{-4}, 10^{-5}, ... What tolerance and how many steps are needed for $a = 0$ in order that the orbit should look closed on the screen,

(a) at the first return to the neighborhood of the starting point?

(b) also at the second return to the neighborhood of the starting point?

You are likely to give up case (b), due to lack of time, if you try to solve it with Runge's 2nd order method. After this you will appreciate that there are problems for which a method of high order of accuracy is needed, even if the request for final accuracy is modest.

An experiment on a VGA screen with the 5th order Runge-Kutta method DOPRI5 (see Sec. 13.3), with an adequate step size control and an adequate (cubic Hermitean) interpolation for the graphics, showed that the final accuracy in case b) required a small tolerance, $TOL = 10^{-7}$. 681 steps (4086 function evaluations) were needed. How far do you get with 4086 function evaluations with Runge's 2nd order method when $TOL = 10^{-7}$?

13.3 One-step Methods

The idea behind most **one-step methods**, is that (t_n, y_n) and a suggested value for the step size $h_n = t_{n+1} - t_n$ are the only input data to the step, where y_{n+1} is computed, and the error committed is estimated. If the error is acceptable, a new step size is suggested for the next step, otherwise y_{n+1} is recomputed with a smaller step size. The one-step method is characterized by an **increment function** Ψ such that

$$y_{n+1} = y_n + h\Psi(t_n, y_n; h). \tag{13.3.1}$$

One of the oldest numerical methods for solving differential equations is based on approximating the solution by a succession of Taylor series expansions. This method is developed in Sec. 13.3.1. The best known one-step methods are the **explicit Runge-Kutta** methods, or in short RK-methods. They will be studied in Sec. 13.3.2, in particular the order condition. The practical step size control is treated in Sec. 13.3.2, where examples of so-called embedded RK-methods are given. Standard references for RK-methods are Hairer et al. [7, Ch. 2] and Butcher [1], where most of the omitted proofs can be found.

13.3.1 The Taylor-Series Method

In Example 3.1.2 we solved an initial-value problem by substituting a power series $y(x) = \sum_{n=0}^{\infty} c_n x^n$ with undetermined coefficients. From the differential equation a recursion formula for the computation of the coefficients was derived. We now show how this method can be extended to a stepwise process. If the solution is sufficiently smooth we have by Taylor's formula

$$\begin{aligned} y(x+h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \cdots \\ + \frac{h^p}{p!}y^{(p)}(x) + \frac{h^{p+1}}{(p+1)!}y^{(p+1)}(\xi). \end{aligned} \tag{13.3.2}$$

Euler's method can be viewed as an approximation of the first two terms in this expansion. If we can evaluate higher derivatives of y, we can obtain a method of order p by neglecting the remainder term in (13.3.2) and using the formula

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2}y''_n + \frac{h^3}{3!}y'''_n + \cdots + \frac{h^p}{p!}y_n^{(p)}. \tag{13.3.3}$$

The first neglected term can be used as an estimate of the local discretization error.

Following Euler we express the derivatives of $y(x)$ in terms of the partial derivatives of $f(x, y)$. Starting with the differential equation $y'(x) = f(x, y(x))$ we differentiate both sides with respect to x to obtain

$$y'' = f_x + f_y y' = f_x + f_y f, \tag{13.3.4}$$

where we have used the notation $f_x = \partial f/\partial x$ and $f_y = \partial f/\partial y$. Differentiating again we obtain

$$y''' = f_{xx} + f_{xy}y' + f_y(f_x + f_y f) + (f_{xy} + f_{yy}y')f \tag{13.3.5}$$

$$= (f_{xx} + 2f_{xy}f + f^2 f_{yy}) + (f_x + f_y f)f_y. \tag{13.3.6}$$

For higher derivatives the formulas soon become very complicated. For an autonomous system $f_x = f_{xy} = f_{xx} = \ldots = 0$, and the formula simplifies considerably. On the other hand, for systems y and f are vectors, and so f_x is a vector, f_y a matrix, etc.

If $f(x, y)$ is composed of elementary functions it is often possible to obtain simple recursion formulas for the successive derivatives using an extension of Newton's series approach. If we introduce the **Taylor coefficients** of $y(x)$ and $f(x, y(x))$ at x_n

$$Y_i = \frac{1}{i!} y_n^{(i)}, \qquad F_i = \frac{1}{i!} f(x, y(x))_n^{(i)},$$

we can write (13.3.3)

$$y_{n+1} = \sum_{i=0}^{p} h^i Y_i.$$

Differentiating $y' = f(x, y(x))$ we get the relation

$$(i+1) Y_{i+1} = F_i, \qquad i = 0, 1, 2, \ldots.$$

If $f(x, y)$ is an algebraic composition of elementary functions we can find formulas for recursively generating the Taylor coefficients F_i and Y_i. We have, for example,

$$f = p \pm q \Longrightarrow F_i = P_i + Q_i,$$

where P_i and Q_i are the Taylor coefficients of p and q. Similarly, by the Cauchy formula

$$f = pq \Longrightarrow F_i = \sum_{j=0}^{i} P_j Q_{i-j}.$$

For the Taylor coefficients F_i of $f = p/q$ we write $p = fq$, and use the Cauchy formula $P_i = \sum_{j=0}^{i} F_j Q_{i-j}$. Solving for F_i w get

$$F_i = \frac{1}{Q_0} \Big(P_i - \sum_{j=0}^{i-1} F_j Q_{i-j} \Big).$$

which is a recursion formula for F_i. Recursion formulas can also be derived for the Taylor coefficients of many elementary functions, see Hairer et al. [1987].

Example 13.3.1.

Determine the first six Taylor coefficients for the function which is the solution to the initial value problem

$$y' = 1 + xy + y^2.$$

Using the formulas above for the Taylor coefficients of a sum and product leads to the recursion $Y_0 = y(x_n)$, $Y_1 = F_0 = f(x_n, y_n)$,

$$(i+1) Y_{i+1} = F_i = Y_{i-1} + X_0 Y_i + \sum_{j=0}^{i} Y_j Y_{i-j}, \qquad i = 1, 2, 3, \ldots,$$

where $X_0 = x_n$. In particular if we take $x_n = 0$, $y(0) = 0$, we obtain $Y_0 = 0$, $Y_1 = 1$,

$$\begin{aligned}
2Y_2 &= Y_0 + X_0 Y_1 + 2Y_0 Y_1 \quad \Longrightarrow Y_2 = 0,\\
3Y_3 &= Y_1 + X_0 Y_2 + 2Y_0 Y_2 + (Y_1)^2 \quad \Longrightarrow Y_3 = 2/3,\\
4Y_4 &= Y_2 + X_0 Y_3 + 2(Y_0 Y_3 + Y_1 Y_2) \quad \Longrightarrow Y_4 = 1/6 = 0,\\
5Y_5 &= Y_3 + X_0 Y_4 + 2(Y_0 Y_4 + Y_1 Y_3) + (Y_2)^2 \quad \Longrightarrow Y_5 = 3/2 = 0.
\end{aligned}$$

Thus

$$y(h) = h + \frac{2}{3}h^3 + \frac{1}{6}h^4 + \frac{3}{2}h^5 + \ldots.$$

Notice that the computation of the recursion formulas for the Taylor coefficients need only be done once. The same recursion formulas can be used at each step (the numerical values of the Taylor coefficients are of course different at each step).

The Taylor series method was used a great deal in hand computation, but was less popular during the first years of the computer age. Since then, programming techniques and languages have been improved, and the popularity of the method has risen again. If $f(x, y)$ is the composition of a sequence of algebraic operations and elementary functions it is easy to write subroutines, which recursively compute the Taylor coefficients. There exist programs which automatically generate such subroutines from a Fortran statement for $f(x, y)$.

It is difficult to make a general comparison between the efficiency of Taylor series methods and the methods previously mentioned. An advantage of the Taylor series approach is that the order of the method and the step size can be varied simply in order to get the required precision. It can also be executed in interval analysis and developed to provide reliable error bounds, see Moore [1966].

Other types of series expansions have also been used with success in differential equation problems–for example, Chebyshev series, see Fox and Parker [1968].

13.3.2 Explicit Runge-Kutta Methods

Let us try to improve Euler's method in the following way. The midpoint rule for numerical integration gives the formula

$$y_{n+1} = y_n + hf\Big(x_n + \frac{h}{2}, y\big(x_n + \frac{h}{2}\big)\Big),$$

which is of second order accuracy. What value should we take for $y(x_n + h/2))$? An obvious possibility is to compute an approximation using Euler's method with step size $h/2$. We then obtain **Runge's second order method** which can be written

$$\begin{aligned}
k_1 &= hf(x_n, y_n),\\
k_2 &= hf\Big(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1\Big),\\
y_{n+1} &= y_n + k_2.
\end{aligned}$$

This method is a special case of a **two-stage** second order Runge-Kutta method. It is called a two-stage method since two evaluation of the function $f(x, y)$ are made

at each step. Since

$$y(x_n + h/2) = y_n + \frac{1}{2}k_1 + O(h^2)$$

this method has the same order as the midpoint rule.

The idea behind Runge-Kutta methods is to compute the value of $f(x, y)$ at several strategically chosen points near the solution curve in the interval $[x_n, x_{n+1}]$, and to combine these values in such a way as to match as many terms as possible in the Taylor series expansion. Since only one previous solution value y_n is needed to calculate y_{n+1} no special starting procedure is needed, and the step size $h = h_n$ may be changed at any step in the calculation.

A general explicit Runge-Kutta method of s stages has the form

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + c_2 h, y_n + a_{21}k_1), \\ k_3 &= hf(x_n + c_3 h, y_n + a_{31}k_1 + a_{32}k_2), \\ &\cdots \\ k_s &= hf(x_n + c_s h, y_n + a_{s1}k_1 + \cdots a_{s,s-1}k_{s-1}), \\ y_{n+1} &= y_n + b_1 k_1 + b_2 k_2 + \cdots + b_s k_s. \end{aligned}$$

It is customary to symbolize the method (13.3.7) by the tableau

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \vdots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} \tag{13.3.7}$$

Usually, the c_i satisfy the conditions

$$c_2 = a_{21}, \quad c_3 = a_{31} + a_{32}, \ \ldots, \ c_s = a_{s1} + a_{s2} + \cdots + a_{s,s-1}, \tag{13.3.8}$$

i.e. they are the row sums of the matrix $\{a_{ij}\}$ in the tableau (13.3.7). These conditions are motivated by the requirement that the predicted y-values where $f(x, y)$ is evaluated should be exact for the special differential equation $y' = 1$.

We now consider the following general two-stage Runge-Kutta method

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + \gamma h, y_n + \alpha k_1), \\ y_{n+1} &= y_n + \beta_1 k_1 + \beta_2 k_2. \end{aligned}$$

We derive conditions for this method to be of second order by computing the Taylor expansion of y_{n+1} as a function of h. We have

$$\begin{aligned} y_{n+1} = y_n + h\beta_1 f_n + h\beta_2 f_n + h^2\beta_2(\gamma f_x + \alpha f_y f)_n \\ + \frac{h^3}{2}\beta_2(\gamma^2 f_{xx} + 2\alpha\gamma f_{xy} f + \alpha^2 f_{yy} f^2)_n + O(h^4). \end{aligned}$$

We wish to make this series expansion agree as closely as possible with the Taylor series for the solution. The best we can do is to match terms up to h^2, which gives the **order conditions**

$$\beta_1 + \beta_2 = 1, \qquad \alpha = \gamma = \frac{1}{2\beta_2}.$$

If these are satisfied the local error is $O(h^3)$, and thus the method of second order. We obtain a one-parameter family of second order Runge-Kutta methods. Taking $\beta_2 = 1$ and thus $\beta_1 = 0$, $\alpha = \gamma = 1/2$ gives the midpoint method. Taking $\beta_2 = 1/2$ we obtain **Heun's method**:

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + h, y_n + k_1), \\ y_{n+1} &= y_n + \frac{1}{2}(k_1 + k_2). \end{aligned} \tag{13.3.9}$$

With $s = 3$ it is possible to construct Runge-Kutta methods of third order accuracy. We determine the six parameters $b_1, b_2, b_3, c_2, c_3, a_{32}$ by matching terms in the series expansion. If the method is to be of third order these parameters must satisfy the system of four nonlinear equations:

$$\begin{aligned} b_1 + b_2 + b_3 &= 1, \\ b_2 c_2 + b_3 c_3 &= 1/2, \\ b_2 c_2^2 + b_3 c_3^2 &= 1/3, \\ b_3 a_{32} c_2 &= 1/6. \end{aligned} \tag{13.3.10}$$

This system has a family of solutions with two parameters, which can be chosen as c_2 and c_3. If $c_2 \neq c_3$ and $c_2 \neq 2/3$ then b_2 and b_3 can be obtained from the second and third equation. The two remaining unknowns b_1 and a_{32} are then found from the first and last equations. The two remaining cases when either $c_2 = c_3$ or $c_2 = 2/3$ give two one-parameter families of solutions. One of the simplest Runge-Kutta methods of third order with easily remembered coefficients is Heun's third order method,

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + h/3, y_n + k_1/3), \\ k_3 &= hf(x_n + 2h/3, y_n + 2k_2/3), \\ y_{n+1} &= y_n + (k_1 + 3k_3)/4. \end{aligned} \tag{13.3.11}$$

It can be shown that for a fourth order Runge-Kutta method four stages are needed. There are now ten parameters which must satisfy eight order conditions. Again there is a two parameter family of solutions, which is now more complicated to derive. Historically the most well-known method is the **classical Runge-Kutta** four stage method of fourth order given by

$$k_1 = hf(x_n, y_n), \tag{13.3.12}$$

$$
\begin{aligned}
k_2 &= hf(x_n + h/2, y_n + k_1/2),\\
k_3 &= hf(x_n + h/2, y_n + k_2/2),\\
k_4 &= hf(x_n + h, y_n + k_3),\\
y_{n+1} &= y_n + (k_1 + 2k_2 + 2k_3 + k_4)/6.
\end{aligned}
$$

This method can be considered as an extension of Simpson's method

$$
\int_{x_n}^{x_n+h} f(x)dx \approx \frac{h}{6}\Big(f(x_n) + 4f\Big(x_n + \frac{h}{2}\Big) + f(x_n + h)\Big).
$$

In the special case where $f(x, y)$ is independent of y and a function of x only they are identical.

Kutta also gave another fourth order method which is slightly more accurate and related to the Newton's 3/8-quadrature rule. We give the tableaus of this and the classical method below:

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{3} & \frac{1}{3} & & & \\
\frac{2}{3} & -\frac{1}{3} & 1 & & \\
1 & 1 & -1 & 1 & \\
\hline
 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8}
\end{array}
\qquad\qquad
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

Much research has been undertaken, in order to determine the "best" fourth order Runge-Kutta method. The freedom in the choice of the parameters can be used to minimize the coefficients in the error terms of order h^5. For a discussion and numerical comparison of these methods see Hairer, Nørsett and Wanner [1987, Sec. II.1]. The differences in accuracy between them appears to be fairly small.

Higher order Runge-Kutta methods have been constructed by Butcher [1986] and others. The derivation of methods of order five and six is rather straightforward, but order 7 and 8 are more complicated. The number of order conditions grows very quickly with the order of the method which makes these constructions a difficult task. For a method of order five there are 17 order condition and this rises to 1205 for a tenth order method!

An interesting and difficult question to answer is how many stages s are necessary to construct a method of order p. The question if there might be a fifth order method with five stages was not answered (negatively) until the mid-sixties. That six stages are needed for $p = 5$ partly explains the popularity of the classical Runge-Kutta four-stage method. It takes two stages to gain one order. A seven stage six-order method has been given by Butcher, who also proved that for $p \geq 7$ no explicit Runge-Kutta method exists of order p with $s = p+1$ stages. He has also succeeded to prove that for $p \geq 8$ no explicit Runge-Kutta method exists of order p with $s = p + 2$ stages. The following table summarizes these results.

The highest order obtained for an explicitly constructed Runge-Kutta method is 10. This was first achieved by Curtis [1975] using 18 stages, and then Hairer [1978] constructed a method using 17 stages.

Table 13.3.1. *Maximal order of explicit Runge-Kutta methods.*

p	2	3	4	5	6	7	≥ 8
s	2	3	4	6	7	9	$> p+2$

13.3.3 Error Estimation and Step Size Control

In Runge-Kutta methods no estimate of the local discretization error is directly available. However, such estimates are necessary for choosing the step sizes h_n to get the required precision of computed results.

The oldest (used already by Runge [1895]) and most straightforward device for the control of the local error is **step doubling** and Richardson extrapolation. Suppose we compute starting from (x_n, y_n) *two* steps with step size $h/2$ using a Runge-Kutta method of order p giving a value y_{n+1}. We then compute, again starting from (x_n, y_n), *one* step with step size h to obtain $\tilde{y}_{n+1}$. Then

$$y(x_n) = y_{n+1} + 2c\Big(\frac{h}{2}\Big)^{p+1} + O(h^{p+2})$$
$$y(x_n) = \tilde{y}_{n+1} + ch^{p+1} + O(h^{p+2}),$$

and subtracting we obtain

$$y_{n+1} - \tilde{y}_{n+1} = 2c\Big(\frac{h}{2}\Big)^{p+1}(2^p - 1) + O(h^{p+2}).$$

Hence an approximation $\hat{y}_{n+1}$ of order $p+1$ is

$$\hat{y}_{n+1} = y_{n+1} + e_{n+1}, \qquad e_{n+1} = (y_{n+1} - \tilde{y}_{n+1})/(2^p - 1). \tag{13.3.13}$$

Here e_{n+1} gives a simple estimate of the error in the *unextrapolated* value y_{n+1}. The solution is then advanced either from y_{n+1} or $\hat{y}_{n+1}$. In the latter case this is called local extrapolation.

The error estimate in (13.3.13) can be used to automatically adjust the step size as follows. Let $l_n = \|e_{n+1}\|$ be a measure of the local error. A common policy is to keep the local error per unit step below a given tolerance ϵ,

$$l_n \leq \epsilon(x_{n+1} - x_n) = \epsilon h. \tag{13.3.14}$$

The new step size h' is chosen to satisfy this condition for the next step. This leads to the choice

$$h' = h\left(\frac{\theta\epsilon h}{l_n}\right)^{1/p}, \tag{13.3.15}$$

where $\theta \leq 1$ (typically $\theta = 0.9$) is a preset safety factor. If the criterion (13.3.14) is not satisfied for the current step, this step is *rejected* and the computations of y_{n+1} repeated with the new step size h'. In several programs the user is asked to

set upper and lower bounds for the permissible step size. Further, h is usually not allowed to increase or to decrease too fast.

Using (13.3.13) with the classical fourth order Runge-Kutta method we need $8+3=11$ function evaluations to proceed two (half) steps from y_n to $\hat{y}_{n+1}$. (Note that the function evaluation of $f(x_n, y_n)$ can be shared by the two different steps sizes.) This is an overhead of of $3/8 = 37.5\%$.

It is more efficient to use Runge-Kutta formulas which simultaneously give approximations to the local error. The idea is to use *a pair* of Runge-Kutta methods characterized by the tableau

$$\begin{array}{llllll}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \vdots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s \\
 & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_{s-1} & \hat{b}_s
\end{array}$$

The pair of formulas are usually constructed so that the methods

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i, \qquad \hat{y}_{n+1} = y_n + \sum_{i=1}^{s} \hat{b}_i k_i,$$

have orders p and $q = p+1$. The difference of the two results then provides an error estimate. Such formulas are called **embedded Runge-Kutta methods**.

The idea of using such formulas was first proposed by Merson. The most well-known of his methods is given by the five stage method

$$\begin{array}{lllllll}
0 & & & & & & \\
\frac{1}{3} & \frac{1}{3} & & & & & \\
\frac{1}{3} & \frac{1}{6} & \frac{1}{6} & & & & \\
\frac{1}{2} & \frac{1}{8} & 0 & \frac{3}{8} & & & \\
1 & \frac{1}{2} & 0 & -\frac{3}{2} & 2 & & \\
\hline
b_i & \frac{1}{10} & 0 & \frac{3}{10} & \frac{2}{5} & \frac{1}{5} \\
\hat{b}_i & \frac{1}{6} & 0 & 0 & \frac{2}{3} & \frac{1}{6}
\end{array}$$

It can be shown that $\hat{y}_n$ is a fourth order method. Although y_n is in general only a third order method, for a linear differential equations with constant coefficients $f(x,y) = Ay + bx$ it becomes effectively fifth order. An estimate of the local truncation error is given by the difference

$$l_{n+1} = \frac{1}{30}(-2k_1 + 9k_3 - 8k_4 + k_5), \tag{13.3.16}$$

see Lambert [1973, pp. 131–132]. Two steps here take ten function evaluations against the eleven required by the step doubling process described previously. However, we have to continue with the fourth order estimate whereas with Richardson

extrapolation we can use local extrapolation to get a fifth order estimate. Also, when applied to a nonlinear differential equation it frequently grossly overestimates the error which leads to a poor step-size control. In spite of this the method has been used successfully, e.g., in the NAG subroutine library.

Another popular embedded Runge-Kutta method is the method of order 4(5) developed by Fehlberg, which requires six function evaluations per step. The first five of these combine to produce a fourth-order method and all six give a fifth order method. Since we assume that the higher order solution is the more accurate this should be used to advance the solution. This method with local extrapolation has been implemented in a much used program called RKF45 by Shampine and Watts [1977]. However, it suffers from the disadvantage that the two formulas are based on the same quadrature formula, which leads to poor step-size control for some problems.

Fehlberg devised his methods so that the error terms for the lower order result y_1 were minimized. Practical results indicate that local extrapolation is preferable. If this is used, then the error terms of the higher order result $\hat{y}_1$ should instead be minimized. This is done in a more recent seven stage method of order 5(4) by Dormand and Prince given below.

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{2}{9} & \frac{2}{9} & & & & & & \\
\frac{1}{3} & \frac{1}{12} & \frac{1}{4} & & & & & \\
\frac{5}{9} & \frac{55}{324} & -\frac{25}{108} & \frac{50}{81} & & & & \\
\frac{2}{3} & \frac{83}{330} & -\frac{13}{22} & \frac{61}{66} & \frac{9}{110} & & & \\
1 & -\frac{19}{28} & \frac{9}{4} & \frac{1}{7} & -\frac{27}{7} & \frac{22}{7} & & \\
1 & -\frac{19}{200} & 0 & \frac{3}{5} & -\frac{243}{400} & \frac{33}{40} & \frac{7}{80} & \\
\hline
b_i & \frac{431}{5000} & 0 & \frac{333}{500} & -\frac{7857}{10000} & \frac{957}{1000} & \frac{193}{2000} & -\frac{1}{50} \\
\hat{b}_i & \frac{19}{200} & 0 & \frac{3}{5} & -\frac{243}{400} & \frac{33}{40} & \frac{7}{80} & 0
\end{array}
$$

This method is also constructed so that $a_{si} = \hat{b}_i$ for all i, and therefore the last evaluation of f in a current step can be re-used for the first evaluation in the the following step. This method seems to be the most efficient of the methods of order 5(4), and is suitable for tolerances down to about 10^{-7}.

Another well known Runge-Kutta code with step size control is DVERK, based on a pair of formulas of order 5 and 6. This method requires eight function evaluations per step and is due to Verner. It is implemented and available in the IMSL Library. Of higher order embedded methods the 13th stage 7th order formula with 8th order error estimate by Fehlberg has been much used for high precision computations, e.g., in astronomy. Excellent results have also been obtained by the 8(7) method of Prince and Dormand [1981]. The coefficients of these methods are given in Hairer, Nørsett, and Wanner [7, 1993], which also includes a subroutine DOPRI8 implementing the Prince and Dormand method with step size control. This code is preferable for tolerances between approximately 10^{-7} and 10^{-13}.

For a code to be efficient over a wide range of accuracy requirements, it is important to be able to vary the order of the formulas being used. Such variable-order codes must estimate the error that would have resulted in using formulas of

different orders. Most codes that implement Runge-Kutta methods, however, are of fixed order. Variable-order codes are common for multistep methods, see Sec. 13.3.

13.3.4 Stiff Problems

There is one class of problems where most of the previously mentioned methods require step sizes which are much shorter than what one would expect if one looks at the solution curve.

Example 13.3.2.
The initial value problem $y' = 100(\sin x - y)$, $y(0) = 0$, has the exact solution

$$y(x) = \frac{1}{1.0001}\Big(\sin x - 0.01\cos x + 0.01e^{-100x} \Big).$$

Notice that the exponential term is less than 10^{-6} already for $x = 0.1$. One might expect that the step size $h = 0.1$ would be possible for $x > 0.1$. However, calculations with the classical fourth order Runge-Kutta method with various step sizes gave the following values for $x = 3$:

Step size	No. of steps	$y(3, h)$
0.015	200	0.151004
0.020	150	0.150996
0.025	120	0.150943
0.030	100	$6.7 \cdot 10^{11}$

We see that $h = 0.025$ gives a good result, whereas for $h = 0.030$ one has a frightful numerical instability.

Example 13.3.3.
The problem

$$\begin{aligned} y_1' &= y_2, \qquad y_1(0) = 1, \\ y_2' &= -1,000y_1 - 1,001y_2, \qquad y_2(0) = -1, \end{aligned}$$

has the exact solution $y_1(x) = -y_2(x) = e^{-x}$. The general solution is $y_1(x) = -y_2(x) = Ae^{-x} + Be^{-1000x}$. Here the same Runge-Kutta method explodes approximately for $h > 0.0027$; this is a very unsatisfactory step size for describing the solution e^{-x}. An explanation for the explosion can be found in Problem 12.5.3.

The two examples above have in common that there are processes in the system described by the differential equation with significantly different time scales. The methods described require that one takes account of the fastest process, even after the corresponding component has died out in the exact solution. More difficult variants—large nonlinear systems—occur in many very important applications—,e.g., in simulation of chemical reactions, electronic networks, control systems, etc. These problems are called **stiff problems**.

Consider the test problem

$$y' = a(y - g(x)) + g'(x), \qquad y(0) = v, \tag{13.3.17}$$

with a real, which has the exact solution

$$y(x) = (v - g(0))e^{ax} + g(x).$$

If $v = g(0)$, then the solution equals $g(x)$, which is assumed to be a slowly varying function. However, if $v \neq g(0)$ and $a \ll 0$, then the solution contains a transient, which rapidly approaches zero.

In Fig. 13.3.4 we compare the explicit and implicit Euler's methods

$$y_{n+1} = y_n + hf(x_n, y_n), \qquad y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}),$$

applied to a stiff problem. The unstable behavior of the explicit method is clearly seen.

Figure 13.3.1. *Comparison of the explicit and implicit Euler's methods.*

13.3.5 A-Stability

For stiff problems adequate definition of stability, see Sec. 13.2.2, is $A(\alpha)$-stability, which means that the stability region S should include the sector $|\arg(qh) - \pi| < \alpha$. The special case of the left half-plane, $\alpha = \pi/2$, is called **A-stability**.

Example 13.3.4.

The trapezoidal method gives if applied to the test problem $y' = qy$ the difference equation

$$y_{n+1} = y_n + \frac{1}{2}hq(y_{n+1} + y_n), \qquad y_0 = 1.$$

Hence $(1 - \frac{1}{2}hq)y_{n+1} = (1 + \frac{1}{2}hq)y_n$, and

$$y_n = \left(\frac{1 + \frac{1}{2}hq}{1 - \frac{1}{2}hq}\right)^n.$$

It follows that y_n is bounded if and only if

$$\left|1 + \frac{1}{2}hq\right| \leq \left|1 - \frac{1}{2}hq\right|,$$

and, by Fig. 12.7.2, this is equivalent to $\mathrm{Re}(\frac{1}{2}hq) \leq 0$. *The left half-plane is the stability region of the trapezoidal method, and hence it is precisely A-stable.* Also note that if $\mathrm{Re}(q) < 0$, then for any choice of step size the solution by the trapezoidal method has the important property in common with the exact solution of the test problem that it tends to zero as nh becomes large.

Figure 13.3.2. *Proof of A-stability of the trapezoidal method.*

If the trapezoidal method is applied to a linear system $y' = Ay$, where some of the eigenvalues $\lambda_i(A)$ have large modulus and negative real part, it is relevant to consider $(1 + \frac{1}{2}hq)(1 - \frac{1}{2}hq)$ when $|hq| \gg 1$. When $|hq| \to \infty$, this quantity tends to -1. This shows that there will appear slowly decreasing *oscillations*. The size of the oscillations can be reduced by the application of the smoothing formula

$$\hat{y}_n = (y_{n+1} + 2y_n + y_{n-1})/4.$$

A-stability implies for multistep methods that $p \leq 2$, and it can be shown that the trapezoidal method has the smallest truncation error among all A-stable linear multistep methods. These definitions of stability apply also to other methods than the linear multistep methods. There are several variations on the theme of stability. An A-stable method for which it holds that for the test equation

$$y_{n+1}/y_n \to 0, \qquad \mathrm{Re}(hq) \to \infty$$

is said to strongly A-stable or **L-stable**. As remarked above the trapezoidal method is not L-stable. However, the backward Euler method can be shown to be L-stable, see Problem 13.4.9.

The methods which have been most successful for stiff problems are all implicit. For example the *trapezoidal method* combined with smoothing and Richardson extrapolation, has been used with success; and so has the **implicit midpoint method**

$$y_{n+1} = y_n + hf\left(\frac{1}{2}(x_n + x_{n+1}), \frac{1}{2}(y_n + y_{n+1})\right). \tag{13.3.18}$$

PROBLEMS

1. (Newton (1671). Derive the Taylor series approximation up to terms of order h^6 for the initial value problem

$$y' = 1 - 3x + y + x^2 + xy, \qquad y(0) = 0.$$

2. Determine a Taylor series expansion for the solution of the equation $y' = y^2$, $y(0) = 1$, about $x = 0$. Use this approximation to compute y for $x = 0.2$ and $x = 1.2$ to four decimals. Compare with the exact solution,and explain why the second case ($x = 1.2$) was unsuccessful.

3. Use the classical Runge-Kutta fourth order method to compute an approximation to $y(0.2)$, where $y(x)$ is the solution to the differential equation $y' = x + y$ with $y(0) = 1$. Compute with six decimals, for two different step sizes, $h = 0.2$ and $h = 0.1$. Extrapolate. Compare with the exact result.

4. Determine the order of the two Runge-Kutta methods

$$y_{n+1} = y_n + k_2, \qquad \hat{y}_{n+1} = y_n + \frac{1}{6}(k_1 + 4k_2 + k_3),$$

where

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + h/2, y_n + k_1/2), \\ k_3 &= hf(x_n + h, y_n - k_1 + 2k_2). \end{aligned}$$

5. We want to compute values of the function

$$y(x) = \int_0^\infty \frac{e^{-t^2}}{t + x} dt$$

for a sequence of values of $x > 1$. We can proceed in the following way: $y(x)$ is computed for $x = 1$ using some method for numerical integration; one finds $y(1) = 0.6051$. Show that y satisfies the differential equation

$$y' + 2xy = -1/x + \sqrt{\pi}.$$

By solving the differential equation numerically with initial value $y(1) = 0.6051$, more values can be computed. Determine $y(x)$ for $x = 1.2$ and $x = 1.4$ by the classical Runge-Kutta method and estimate the error in $y(1.4)$.

6. The stability regions of explicit Runge-Kutta methods of order $s \leq 4$, are shown in Fig. 12.5.1. (Since the regions are symmetric with respect to the imaginary axis only half of them are shown.) Calculate all the intersections of the region corresponding to $s = 4$ with the real and imaginary axis.

Figure 13.3.3. *Stability regions of explicit Runge-Kutta methods.*

13.4 Linear Multistep Methods

One-step methods only use information from the previous point (x_n, y_n) to compute the approximation of y_{n+1}. In contrast in multistep formulas we assume that we know approximations $y_n, y_{n-1}, \ldots, y_{n-k+1}$ to the exact solution at the k points x_{n-j}, $j = 0, 1, \ldots, k-1$.

A **general linear multistep** method for the differential equation $y' = f(x, y)$, $y(0) = y_0$, is defined by the difference equation

$$\sum_{i=0}^{k} (\alpha_i y_{n+i} - h\beta_i f_{n+i}) = 0, \tag{13.4.1}$$

where α_i and β_i are real parameters, h the step length and $f_i = f(x_i, y_i)$. The formula (13.4.1) is also called a linear k-step method. This class includes all the previously considered classical linear multistep methods, in particular the midpoint method, Euler's method and the trapezoidal method. The y_n can be computed recursively from (13.4.1) if in addition to the initial value y_0, $k-1$ more values $y_1, \ldots, y_{k-1}$ are given. If $\beta_k \neq 0$ the method is **implicit**, and then this may be true only for sufficiently small h.

13.4.1 The Adams Methods

An important class of linear multistep methods dates back to work by Adams about 1855. Following Adams we consider the integrated form of the first order differential equation (13.1.1)

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(t, y(t))\, dt.$$

We now replace the function $f(t, y(t))$ in the integral by the polynomial $p_{k-1}(t)$ of degree $k-1$ interpolating the values

$$(x_i, f_i), \qquad i = n-k+1, \ldots, n,$$

where $f_i = f(x_i, y_i)$. Assume that the points are equidistant, $x_i = x_0 + ih$, and use Newton's interpolation formula for equidistant interpolation (see (3.4.9) we can

write this polynomial

$$p_{k-1}(t) = p_{k-1}(x_n + sh) = \sum_{j=0}^{k-1} (-1)^j \binom{-s}{j} \nabla^j f_n,$$

where ∇ denotes the backward difference operator, see Sec. 3.2.1. Inserting $p_{k-1}(t)$ and integrating we get the numerical formula

$$y_{n+1} = y_n + h \sum_{j=0}^{k-1} \gamma_j \nabla^j f_n, \qquad \gamma_j = (-1)^j \int_0^1 \binom{-s}{j} ds.$$

Note that the coefficients γ_j do not depend on the order k. Inserting numerical values for we get the family of **explicit Adams methods** of increasing order

$$y_{n+1} - y_n = h\Big(1 + \frac{1}{2}\nabla + \frac{5}{12}\nabla^2 + \frac{3}{8}\nabla^3 + \frac{251}{720}\nabla^4 + \ldots\Big) f_n. \tag{13.4.2}$$

The backward differences can be expressed in function values using $\nabla^j f_n = (1 - E^{-1})^j f_n$, and we obtain in particular for $k = 1, 2, 3, 4$ the methods of order up to $p = 4$.

$$\begin{aligned}
y_{n+1} &= y_n + h f_n; \\
y_{n+1} &= y_n + \frac{h}{2}(3f_n - f_{n-1}); \\
y_{n+1} &= y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2}); \\
y_{n+1} &= y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3});
\end{aligned}$$

The first formula here is the explicit Euler method. An attractive feature of these methods is that independent of order k only one evaluation of the function f is needed at each step.

By interpolating also the point (x_{n+1}, f_{n+1}) we obtain a family of implicit formulas. In the equidistant case the corresponding interpolation polynomial equals

$$p_k^*(t) = p_k^*(x_n + sh) = \sum_{j=0}^{k} (-1)^j \binom{-s+1}{j} \nabla^j f_{n+1}.$$

Inserting this into the integral we obtain

$$y_{n+1} = y_n + h \sum_{j=0}^{k} \gamma_j^* \nabla^j f_{n+1}, \qquad \gamma_j^* = (-1)^j \int_0^1 \binom{-s+1}{j} ds,$$

which gives the family of **implicit Adams methods** of increasing order $p = k + 1$

$$y_{n+1} - y_n = h\Big(1 - \frac{1}{2}\nabla - \frac{1}{12}\nabla^2 - \frac{1}{24}\nabla^3 - \frac{19}{720}\nabla^4 - \frac{3}{160}\nabla^5 - \ldots\Big) f_{n+1}. \tag{13.4.3}$$

Expressed in function values the methods up to order $p = 5$ are

$$
\begin{aligned}
y_{n+1} &= y_n + hf_{n+1};\\
y_{n+1} &= y_n + \frac{h}{2}(f_{n+1} + f_n);\\
y_{n+1} &= y_n + \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1});\\
y_{n+1} &= y_n + \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2});\\
y_{n+1} &= y_n + \frac{h}{720}(251f_{n+1} + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3});
\end{aligned}
$$

The first two formulas here are the implicit Euler method, and the trapezoidal method, respectively. For an operator derivation of the Adams methods see Problem 9 of Section 3.2.

The local errors in the Adams methods are approximately equal to the first neglected term in the series. Note that the coefficients γ_j^* decrease much faster than γ_j, and hence the implicit Adams methods have smaller local error.

Multistep methods need a special starting procedure to compute the $k-1$ extra starting values $y_{k-1}, \ldots, y_1$. These can be obtained, e.g., by using a Runge-Kutta method of appropriate order. However, the usual practice is to simply start with a multistep method of order one and a very small step size, and then successively increase the order and step size. We discuss this in greater depth in Sec. 13.4.4.

If f is a nonliner function, then using the implicit Adams methods one has to solve a nonlinear system at each step. This can be done by fixed point iteration. We write

$$
y_{n+1} = h\beta_k f_{n+1} + u_n, \qquad u_n = y_n + h\Big(\beta_{k-1} f_n + \ldots + \beta_0 f_{n-k+1}\Big),
$$

where u_n is known and iterate

$$
y_{n+1}^{(m+1)} = h\beta_k f(x_{n+1}, y_{n+1}^{(m)}) + u_n, \qquad m = 0, 1, 2, \ldots. \tag{13.4.4}
$$

If the step size h is small enough $y_{n+1}^{(m+1)}$ converges to the solution y_{n+1} of the implicit formula. A sufficient condition for convergence is

$$
h\beta_k \left\| \frac{\partial f}{\partial y} \right\| < 1. \tag{13.4.5}
$$

A good initial approximation for the implicit Adams method of order $p+1$ can be obtained by using the explicit Adams method of order p. The explicit formula is called a **predictor**, while the implicit formula is called a **corrector**. The whole procedure is called a **predictor-corrector** method.

The stopping of the iterations may be controlled by comparing the difference $y_{n+1}^{(m+1)} - y_{n+1}^{(m)}$ to some preset tolerance. In this case usually a maximum of three iterations are allowed. Another possibility is to use a predetermined number of iterations. The latter is more common, and it is advisable to choose a step length

such that one iteration will suffice. These codes usually recompute the function value $f(x_{n+1}, y_{n+1}^{(1)})$ (note that this value is needed for the next step) and hence use two function evaluation per step.

More generally we could consider the integral equation

$$y(x_{n+1}) = y(x_{n-i}) + \int_{x_{n-i}}^{x_{n+1}} f(t, y(t))dt.$$

Taking $i = 0$ we get the Adams methods. With $i = 1$ we obtain the **Nyström methods**. Inserting the polynomial $p_{k-1}(t)$ and integrating we obtain the explicit methods

$$y_{n+1} = y_{n-1} + h\sum_{j=0}^{k-1} \kappa_j \nabla^j f_n,$$

or

$$y_{n+1} = y_{n-1} + h\Big(2f_{n+1} + \frac{1}{3}\nabla^2 f_n - \frac{1}{3}\nabla^3 f_n - \frac{29}{90}\nabla^4 f_n + \ldots\Big).$$

The special case

$$y_{n+1} = y_{n-1} + 2hf_n,$$

is the **explicit midpoint** method which we recognize from Sec. 13.3.2. Implicit Nyström methods can be similarly derived. The Nyström methods are in general not useful since they suffer from a weak instability as exemplified by the explicit midpoint method.

13.4.2 Local Error and Order Conditions

With the multistep method (13.4.1) we associate the linear difference operator

$$\mathcal{L}_h y(x) = \sum_{i=0}^{k} (\alpha_i y(x + ih) - h\beta_i y'(x + ih)), \tag{13.4.6}$$

where $y(x)$ is an arbitrary function, continuously differentiable on an interval that contains the values $x+ih$ for $i = 0, 1, \ldots, k$. We say that the **order of consistency** is p, if p is the largest integer such that $\mathcal{L}_h P(x)$ vanishes identically for any pth degree polynomial. An equivalent definition is:

Definition 13.4.1.

The method (13.4.1) is said to be of **order** *p, if for all y with continuous derivatives of order $p + 1$ it holds that*

$$\mathcal{L}_h y(x) \sim c_{p+1} h^{p+1} y^{(p+1)}(x), \quad h \to 0. \tag{13.4.7}$$

If $p \geq 1$, the multistep method is said to be **consistent**.

Expanding $y(x + ih)$ and its derivative $y'(x + ih)$ in Taylor series about x, inserting these into (13.4.6), and collecting terms gives

$$\mathcal{L}_h y(x) = c_0 y(x) + c_1 h y'(x) + \ldots + c_p h^p y^{(p)}(x) + \ldots,$$

where $c_0, c_1, \ldots, c_p, \ldots$ are constants. The order of the method is given by the first non-vanishing term in this expansion. Hence, the value of p and the constant c_{p+1} can be determined by using as test functions the polynomials $x^q/q!$, $q = 1, \ldots, p+1$, which leads to the following order conditions:

Theorem 13.4.2.

The multistep method (13.4.1) is of order p, if and only if, the following conditions are satisfied:

$$\sum_{i=0}^{k} \alpha_i = 0, \qquad \sum_{i=0}^{k} \alpha_i i^q - q \sum_{i=0}^{k} \beta_i i^{q-1} = 0, \quad q = 1, \ldots, p. \tag{13.4.8}$$

The constant c_{p+1} is given by

$$c_{p+1} = \frac{1}{(p+1)!} \Big(\sum_{i=0}^{k} \alpha_i i^{p+1} - (p+1) \sum_{i=0}^{k} \beta_i i^p \Big) \neq 0. \tag{13.4.9}$$

Definition 13.4.3.

By the local truncation error of a multistep method (13.4.1) at x_{n+k} we mean the error $y(x_{n+k}) - y_{n+k}$, where $y(x)$ is the exact solution of $y' = f(x, y)$, $y(x_n) = y_n$, and y_{n+k} is the numerical solution obtained from (13.4.1) by using the exact starting values $y_i = y(x_i)$ for $i = n, n+1, \ldots, n+k-1$.

For $k = 1$ this definition coincides with the definition of the local error for one-step methods. The local error is essentially equal to $\alpha_k^{-1} \mathcal{L}_h y(x)$, see Hairer et al. [7, 1993, Ch. III.2].

The **generating polynomials**

$$\rho(\zeta) = \sum_{i=0}^{k} \alpha_i \zeta^i, \qquad \sigma(\zeta) = \sum_{i=0}^{k} \beta_i \zeta^i, \tag{13.4.10}$$

play a fundamental role in the theory of multistep methods. We have

$$\mathcal{L}_h e^x = (\rho(e^h) - h\sigma(e^h)) e^x.$$

Hence, the method is of order p if and only if

$$\rho(e^h) - h\sigma(e^h) \sim c h^{p+1}, \qquad h \to 0. \tag{13.4.11}$$

In particular, consistency is easily shown to be equivalent to the equations

$$\rho(1) = 0, \qquad \rho'(1) = \sigma(1).$$

To compare the local errors of multistep methods of the same order we could use the constant c_{p+1} in (13.4.9). However, that is not a suitable measure of accuracy, since multiplication of (13.4.1) by a constant will change c_{p+1}. It can be

shown that a more relevant measure is

$$C = c_{p+1}/\sigma(1), \qquad \sigma(1) = \sum_{i=0}^{k} \beta_i, \tag{13.4.12}$$

which is called the **error constant** of the method (13.4.1).

Example 13.4.1.

For the Adams methods we have

$$\rho(\zeta) = \zeta^k - \zeta^{k-1}, \qquad \rho'(\zeta) = k\zeta^{k-1} - (k-1)\zeta^{k-2},$$

and hence consistency gives $\rho'(1) = 1 = \sigma(1)$. It follows that for these methods $C = c_{p+1}$.

13.4.3 Linear Stability Theory

In Sec. 13.2.2 we used the simple differential equation problem

$$y' = \lambda y, \qquad y(0) = 1, \tag{13.4.13}$$

where λ is a complex constant, as a test problem for studying the stability of numerical methods for initial value problems. The stability region S of a numerical method was defined in Def. 13.2.3 as the set of complex values of $q = \lambda h$ for which all solutions y_n of the test problem (13.4.8) remain bounded as $n \to \infty$. If S contains the origin, the method is zero stable.

A linear multistep method is zero-stable if and only if all solutions of the difference equation

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = 0 \tag{13.4.14}$$

are bounded for all positive n. The solution y_n can be interpreted as the numerical solution for the differential equation $y' = 0$. We find that $y_n = \zeta_j^n$ satisfies (13.4.14) if and only if ζ is a root of $\rho(\zeta)$ defined in (13.4.10). Further, if ζ_j has multiplicity $m_j > 1$, then a solution is $y_n = p_j(n)\zeta_j^n$, where $p_j(n)$ is a polynomial of degree $m_j - 1$. Thus we have the following result:

Theorem 13.4.4.

Necessary and sufficient for stability of the linear multistep method (13.4.1) are the following **root conditions***:*

i. All roots of $\rho(\zeta)$ should be located inside or on the unit circle $|z| \leq 1$;

ii. The roots on the unit circle should be simple.

Example 13.4.2.

For the explicit and implicit Adams methods $\rho(\zeta) = \zeta^k - \zeta^{k-1}$, and besides the simple root $\zeta = 1$ there is a root $\zeta = 0$ of multiplicity $k-1$. Note that by the consistency condition $\rho(1) = 0$, there is always one root equal to 1. For the Adams methods all the other roots are at the origin.

The relevance of the stability concept defined above is shown by the following theorem, which summarizes several theorems proved in Henrici [1962, Chap. 5]. A hint to a proof is given by Problems 5 and 6 (c) of this section.

Theorem 13.4.5.

Suppose that $y(x)$ is a $p+1$ times differentiable solution of the initial-value problem, $y' = f(x,y)$, $y(0) = y_0$, $p \geq 1$, $\|y^{(p+1)}(x)\| \leq K_0$, and that $f(x,y)$ is differentiable for all x, y. Suppose further that $\{y_n\}$ is defined by the equations

$$y_n = y(x_n) + \epsilon_n, \qquad n = 0, 1, \ldots, k-1,$$

$$\sum_{i=0}^{k} (\alpha_i y_{n+i} - h\beta_i f(x_{n+i}, y_{n+i})) = \epsilon_n, \qquad k \leq n+k \leq (b-a)/h.$$

If the multistep is stable and satisfies (13.4.7), then there exist constants K_1, K_2, h_0 such that for all $x_n \in [a,b]$, $h \leq h_0$,

$$\|y_n - y(x_n)\| \leq \Big(c_{p+1} h^p (x_n - a) K_0 + \sum_{i=0}^{n} \|\epsilon_i\|\Big) K_1 e^{K_2 (x_n - a)}. \tag{13.4.15}$$

K_1 depends only on the coefficients of the method, while K_2 also contains an upper bound for $\|\partial f / \partial y\|$.

In view of this result, the integer p is called the **order of accuracy** of the method.

It is sufficient to consider the trivial case $f(x,y)$ constant in order to show that stability and consistency are *necessary* for such a result, with $p > 0$. A corollary of the theorem and this remark in a more precise formulation is that

$$\text{Consistency} + \text{Stability} \Longleftrightarrow \text{Convergence}$$

Convergence here includes uniform convergence in $[a,b]$, when $h \to 0$, for all f which satisfy the assumptions made in Sec. 13.1.1, as well as a requirement that the effect of perturbations of the initial values should tend to zero when the perturbations do so themselves. The formulation given above occurs in numerous other applications of finite-difference methods to ordinary and partial differential equations where these three concepts are defined appropriately for each problem area. "Consistency" usually means that the difference equation formally converges to the differential equation as $h \to 0$, while "convergence" is related to the behavior of *the solutions* of the difference equations.

If the termination of the iteration in an implicit multistep method is controlled by a tolerance on the residual, then an error bound can be obtained by Theorem 13.4.5. If a predictor-corrector technique with a fixed number of iterations is used, then this theorem does not guarantee that the $\|\epsilon_n\|$ do not grow.

The stability of such a scheme can be different from the stability of the corrector method.

Predictor-corrector methods are considered in an extension of the multistep methods, named **multivalue methods**; see Gear [1971, Chap. 9]. This extension is important in several other respects, but it is beyond the scope of this presentation.

It is important to distinguish between the stability question of the differential equation (see Figs. 13.1.2a and 13.1.2b) and the stability questions for the numerical methods. The former is well-conditioned—e.g., if $e^{L(b-a)}$ is of moderate size, where L is an upper bound for the logarithmic norm, defined in Sec. 13.1.4. Compare the general distinction between an ill-conditioned problem and an unstable algorithm in Sec. 2.4.5.

For a multistep method to be of order p it has to satisfy the $p+1$ order conditions (13.4.8). For a k-step method we have $2k+1$ free parameters, if we scale the coefficients so that $\alpha_k = 1$. It therefore seems that order $p = 2k$ should be possible to attain. However, because there is a *conflict between consistency and stability* these methods are not zero-stable are therefore not of practical interest. Stability requirements impose the following restrictions on the attainable order of multistep methods, the so called "first Dahlquist-barrier":

Theorem 13.4.6.

The order p of a zero-stable linear k-step method satisfies:

$$p \leq \begin{cases} k+2, & \text{if } k \text{ is even;} \\ k+1, & \text{if } k \text{ is odd;} \\ k, & \text{if } \beta_k/\alpha_k \leq 0 \text{ (if the method is explicit);} \end{cases}$$

A zero-stable method with $p = k+2$ is called an **optimal** method. An example of an optimal method is the **Milne-Simpson method**

$$y_{n+2} = y_n + h\frac{1}{3}(f_{n+2} + 4f_{n+1} + f_n),$$

for which $p = 4$, $k = 2$. However, the Milne-Simpson method like all optimal methods is only weakly stable and may show an exponential error growth of the type illustrated for explicit midpoint method. This severely limits its use as a general purpose method.

Example 13.4.3.

The method

$$y_{n+2} = -4y_{n+1} + 5y_n + h(4f_{n+1} + 2f_n)$$

is the only explicit 2-step method with order $p = 3$. The characteristic equation

$$\rho(\zeta) = \zeta^2 + 4\zeta - 5 = 0$$

has the two roots $\zeta_1 = 1$ and $\zeta_2 = -5$, and hence is not zero-stable.

The midpoint method exemplifies that K_2 may be positive even though L is negative in (13.4.15). Hence if $b-1$ is large, the error bound of Theorem 13.4.5 (as well as the actual error) can be large unless h and the perturbation level are very small. Therefore, *the stability concept just defined is not always sufficient.* This is true in particular for stiff problems, see Sec. 13.3.4.

13.4.4 Variable Step and Order

For efficiency it is necessary to vary the step size and order used with multistep methods during the integration. To change the order in the family of Adams methods is simple. We can increase or decrease the order one step by just adding or deleting a term in the formulas in Eqs. (13.4.3) and (13.4.5). Since the number of function evaluations per step is independent of the order, the order can be chosen such that the new step size is maximal, consistent with the local error criterion.

Changing the step size, on the other hand, is not as simple as with one step methods. In the derivation of the classical multistep methods in Sec. 13.4.1 it was assumed that numerical approximations $y_n, \ldots, y_{n-k+1}$ are available at equidistant points $x_{n-j} = x_n - jh$, $j = 0, 1, \ldots, k-1$. One possibility, used already by Adams, is to use interpolation to reconstruct initial values on an equidistant net, whenever the step size is changed. This technique is no longer used, since the resulting formulas are not very stable even in case the change is restricted by

$$\frac{1}{2} = \omega \le h_n/h_{n-1} \le \Omega = 2, \qquad h_n = x_{n+1} - x_n.$$

Instead we now outline how to directly derive Adams methods for variable step sizes. We now use Newton's general interpolation method, which can be written

$$\begin{aligned} p(t) &= \sum_{j=0}^{k-1} \left(\prod_{i=0}^{j-1} (t - x_{n-i}) \right) f[x_n, x_{n-1}, \ldots, x_{n-j}] \\ &= \sum_{j=0}^{k-1} \left(\prod_{i=0}^{j-1} \frac{t - x_{n-i}}{x_{n+1} - x_{n-i}} \right) \Phi_j^*(n), \end{aligned}$$

where we have introduced the scaled divided differences

$$\Phi_j^*(n) = \left(\prod_{i=0}^{j-1} (x_{n+1} - x_{n-i}) \right) f[x_n, x_{n-1}, \ldots, x_{n-j}].$$

The explicit Adams method can then be written

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} p(t)dt = h_n \sum_{j=0}^{k-1} g_j(n) \Phi_j^*(n),$$

where

$$g_j(n) = \frac{1}{h_n} \int_{x_n}^{x_{n+1}} \prod_{i=0}^{j-1} \frac{t - x_{n-i}}{x_{n+1} - x_{n-i}} dt.$$

It is fairly easy to see that recursion formulas can be developed for computing the scaled divided differences $\Phi_j^*(n)$. Recursion formulas can also be derived for $g_j(n)$, although these are more complicated, see Hairer et al. [1987, III.5]. The cost of computing these integration coefficients is the biggest disadvantage to permitting arbitrary variations in step size for the Adams methods.

Formulas for the implicit Adams methods can be similarly developed. Here

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} p^*(t)dt,$$

where

$$p^*(t) = p(t) + \prod_{i=0}^{k-1}(t - x_{n-i}) f[x_{n+1}, x_n, \ldots, x_{n-k+1}].$$

Hence,

$$y_{n+1} = y_{n+1}^{(p)} + h_n g_k(n) \Phi_k(n+1),$$

where $y_{n+1}^{(p)}$ is the value predicted by the explicit Adams method, and

$$\Phi_k(n+1) = \prod_{i=0}^{k-1}(x_{n+1} - x_{n-i}) f[x_{n+1}, x_n, \ldots, x_{n-k+1}].$$

13.4.5 Backward Differentiation Methods

The multistep methods derived above were all based on numerical integration. We now derive a formula based instead on numerical differentiation. Let $q(x)$ be a polynomial which interpolates the values y_i, $i = n-k+1, \ldots, n+1$. Then we can write

$$q_k(t) = q_k(x_n + sh) = \sum_{j=0}^{k}(-1)^j \binom{-s+1}{k} \nabla^j y_{n+1}.$$

To determine y_{n+1} we require that

$$q'(x_{n+1}) = f(x_{n+1}, y_{n+1}).$$

This leads to the implicit **backward differentiation formulas**

$$\sum_{j=1}^{k} \delta_j^* \nabla^j y_{n+1} = h f_{n+1}, \qquad \delta_j^* = (-1)^j \frac{d}{ds}\binom{-s+1}{j}\Big|_{s=1} = \frac{1}{j}.$$

These methods can also be derived using the formula

$$hD = -\ln(1-\nabla) = \sum_{j=1}^{\infty} \frac{1}{j}\nabla^j,$$

see the table in Sec. 3.2.2.

The BDF family of multistep methods

$$\nabla y_{n+1} + \frac{1}{2}\nabla^2 y_{n+1} + \frac{1}{3}\nabla^3 y_{n+1} + \ldots + \frac{1}{k}\nabla^k y_{n+1} = hf(x_{n+1}, y_{n+1}) \qquad (13.4.16)$$

has been used with success on stiff differential equations. In particular, for $k = 1, 2$, we obtain

$$\begin{aligned} y_{n+1} - y_n &= hf(x_{n+1}, y_{n+1}), \\ \frac{3}{2}y_{n+1} - 2y_n + \frac{1}{2}y_{n-1} &= hf(x_{n+1}, y_{n+1}). \end{aligned}$$

For $k = 1$ we recognize the implicit Euler method. As k increases the local truncation error decreases, but the stability properties become worse. For $k > 6$ these methods are unstable and therefore useless.

The fixed point iteration in (13.4.4), however, is no good on stiff problems, since the convergence condition (13.4.5), means exactly that the fastest components limit the rate of convergence and thus also the step size which can be used. This is unacceptable, and instead one uses some *modification of Newton's method.* For example, for the BDF methods we have to solve a system of equations of the form

$$F(y_{n+1}) = y_{n+1} - h\beta_0 f(x_{n+1}, y_{n+1}) - \phi_n = 0, \qquad (13.4.17)$$

where ϕ_n is a constant that groups the terms from the previous points. Newton's method for this system is

$$(I - hJ)(y_{n+1}^{m+1} - y_{n+1}^m) = -y_{n+1}^m + h\beta_0 f(x_{n+1}, y_{n+1}^m) + \phi_n = 0,$$

where

$$J = \Big(\frac{\partial f}{\partial y}\Big)_{n+1}.$$

Here to compute the Jacobian matrix $J \in \mathbf{R}^{s\times s}$ we must evaluate s^2 partial derivatives $\partial f_i/\partial y_j$, $1 \le i, j \le s$, which often can be costly.

The BDF methods generalize more easily to variable step size than the Adams methods. The interpolation polynomial $q(t)$ of degree k that interpolates (x_i, y_i) for $i = n+1, n, \ldots, n-k+1$ can now be written using divided differences

$$q(t) = \sum_{j=0}^{k}\prod_{i=0}^{j-1}(t - x_{n-i+1})y[x_{n+1}, \ldots, x_{n-j+1}].$$

Differentiating with respect to t and putting $t = x_{n+1}$ we get

$$q'(x_{n+1}) = \sum_{j=1}^{k}\prod_{i=1}^{j-1}(x_{n+1} - x_{n-i+1})y[x_{n+1}, \ldots, x_{n-j+1}] = f(x_{n+1}, y_{n+1}).$$

PROBLEMS

1. (a) Show that for coefficients in the Adams methods it holds that

$$\gamma_0^* = \gamma_0, \qquad \sum_{i=0}^{k} \gamma_i^* = \gamma_k.$$

(b) Show by induction using the result in (a) that (13.4.4) can be written as

$$y_{n+1} = y_{n+1}^{(p)} + h\gamma_k \nabla^k f_{n+1},$$

where $y_{n+1}^{(p)}$ is the value obtained by the predictor.

2. Determine the stability region of:

(a) the Euler method $y_{n+1} = y_n + hf(x_n, y_n)$;

(b) the backward Euler method $y_{n+1} = y_n + hf(x_{n+1}, y_{n+1})$;

(c) What is the order of accuracy of the latter method?

(d) Are the methods stable?

3. (a) Design a third order method for the solution of a differential equation $y' = f(x, y)$ based on the explicit Adams formulas (13.4.3). Apply it to $y' = y^2$, $h = 0.1$, and compute y_3 and y_4 when

$$y_0 = 1.0000, \qquad y_1 = 1.1111, \qquad y_2 = 1.2500$$

are given. Use four decimals. Compare with the exact results.

(b) Improve the value of y_3 using an implicit Adams formula of (13.4.3) truncated after the second term. Go on from this value, compute y_4 using the explicit formula, compute f_4, and improve y_4 using the implicit method (predictor-corrector technique). Improve y_4 by another iteration.

(c) What is the approximate convergence rate for the iterations in (b)?

4. Show that β_1 and β_2 can be determined so that $p = 3$ for the multistep method

$$y_{n+1} = -4y_n + 5y_{n-1} + h\big(\beta_1 f(x_n, y_n) + \beta_2 f(x_{n-1}, y_{n-1})\big).$$

Calculate the error constant c_{p+1} in equation (13.4.9).

5. The explicit midpoint method applied to the test equation gives the difference equation

$$y_{n+1} = y_{n-1} + 2hqy_n.$$

(a) Show that the characteristic equation has the two roots

$$u_1 = z + (1 + z^2)^{1/2}, \qquad u_2 = -1/u_1, \qquad (z = hq).$$

(b) Show that for $x = nh$, $nz = nhq = qx$,

$$u_1^n = e^{qx(1 - z^2/6 + O(z^3))},$$

so that if $|z| \ll 1$ then u_1^n is close to the correct solution e^{qx}.

(c) Show that

$$u_2^n = (-1)^n u_1^{-n} = (-1)^n e^{-qx(1 - O(z^2))},$$

and hence if $q < 0$, then u_2^n produces exponentially growing oscillations, even though the solution to the differential equation decreases exponentially! This explains the weak instability seen previously.

6. (a) Show that all solutions of the difference equation

$$y_{n+1} - 2\lambda y_n + y_{n-1} = 0$$

are bounded, as $n \to \infty$, if $-1 < \lambda < 1$, while for any other λ in the complex plane there exists at least one solution which is unbounded.

(b) Let A be a diagonalizable matrix. Give, in terms of the eigenvalues of A, a necessary and sufficient condition for the boundedness as $n \to \infty$ of all solutions of the difference equation

$$y_{n+1} - 2Ay_n + y_{n-1} = 0$$

7. Consider the application of the linear multistep method, defined by the polynomials ρ and σ, to the usual test problem $y' = qy$, $y(0) = 1$.

(a) Show that if ζ_j is a simple root of ρ, then the difference equation has a particular solution which is close to $y_n = \zeta_j^n e^{\lambda_j q x_n}$, $x_n = nh$, where the so called **growth parameter** λ_j is given by

$$\lambda_j = \frac{\sigma(\zeta_j)}{\zeta_j \rho'(\zeta_j)}.$$

(b) Show that if $\zeta_j = 1$ and the method is consistent, then $\lambda_1 = 1$.

(c) Compute the growth parameters for the midpoint method and compare with the results of Problem 6.

(d) Compute the growth parameters for the Milne-Simpson's method

$$y_{n+1} = y_{n-1} + \frac{1}{3}h\big(f_{n+1} + 4f_n + f_{n-1}\big).$$

Is the method weakly stable?

8. The following results were obtained for the problem

$$y' = x^2 - y^2, \qquad y(0) = 1,$$

using the modified midpoint method with different step sizes:

$$h = 0.05 : 0.83602; \qquad h = 0.1 : 0.83672.$$

Compute a better value by extrapolation.

9. Determine the stability region of backward Euler method

$$y_{n+1} = y_n + f_{n+1}.$$

Is it A-stable? Is it L-stable? What is the order of accuracy of this method?

10. Show that the second-order backward differentiation method

$$\nabla y_n + \frac{1}{2}\nabla^2 y_n = hf_n$$

is $A(0)$-stable. (Actually it is A-stable.)

13.5 Extrapolation Methods

13.5.1 Extrapolated Euler's Method

In Euler's method for the initial-value problem

$$y' = f(x, y), \qquad y(0) = y_0, \tag{13.5.1}$$

one seeks approximate values $y_1, y_2, \ldots$ to the exact solution $y(x_1), y(x_2), \ldots$ by approximating the derivative at the point (x_n, y_n), $x_n = x_0 + nh$, with the difference quotient $(y_{n+1} - y_n)/h$. This gives the recursion formula

$$y_{n+1} = y_n + hf(x_n, y_n), \qquad y_0 = y(0). \tag{13.5.2}$$

The weakness of Euler's method is that the step size must be chosen quite small in order to attain acceptable accuracy.

Example 13.5.1.
For the initial value problem

$$y' = y, \qquad y(0) = 1,$$

the solution is computed with Euler's method first with $h = 0.2$ and then with $h = 0.1$, and is compared with the exact solution $y(x) = e^x$:

x_n	$y(x_n)$	y_n	hf_n	error	y_n	hf_n	error
0	1.000	1.000	0.200	0.000	1.000	0.100	0.000
0.1	1.105				1.100	0.110	-0.005
0.2	1.221	1.200	0.240	-0.021	1.210	0.121	-0.011
0.3	1.350				1.331	0.133	-0.019
0.4	1.492	1.440	0.288	-0.052	1.464	0.146	-0.028
0.5	1.649				1.610	0.161	-0.039
0.6	1.822	1.728		-0.094	1.771		-0.051

The error grows as x increases and is approximately proportional to h.

The following theorem gives a theoretical basis for the use of repeated Richardson extrapolation in connection with Euler's method.

Theorem 13.5.1.
Denote by $y(x, h)$ the result of the use of Euler's method with step length h on the differential equation problem (13.5.1). Then for the global error there is an expansion of the form

$$y(x, h) - y(x) = e_1(x)h + e_2(x)h^2 + \cdots + \cdots + e_q(x)h^q + O(h^{q+1}). \tag{13.5.3}$$

Proof. See Hairer et al. [1987, II.8]. The proof is related to the fact that there is a similar expansion for the local error of Euler's method. From Taylor's formula we

know that

$$\frac{y(x+h)-(y(x)+hy'(x))}{h}=\frac{h}{2}y''(x)+\frac{h^2}{3!}y'''(x)+\cdots+\cdots+\frac{h^q}{q!}y^{(q)}(x)+O(h^{q+1}).$$

□

In an extrapolation method based on Euler's method a basic step size H, and a sequence of integers $n_1 < n_2 < n_3 < \ldots$ are chosen. This defines a sequence of step sizes

$$h_i = H/n_i, \quad i = 1, 2, 3, \ldots.$$

Denote by $A_{i,1} = y_{h_i}(x_0 + H)$ the result of the numerical method using the step size h_i. We eliminate as many terms as possible from the error expansion (13.5.3) by computing the interpolating polynomial $p(h)$ such that

$$p(h_i) = A_{i,1}, \quad i = j, j-1, \ldots, j-k+1,$$

and extrapolate to the limit $h \to 0$. The integration then proceeds from the point $(x_0 + H, A_{jk})$, where $A_{j,k} = p(0)$.

By the above theorem repeated Richardson extrapolation can be performed using the Aitken-Neville algorithm (see Sec. 4.2.3).

$$A_{i,k+1} = A_{i,k} + \frac{A_{i,k} - A_{i-1,k}}{(n_i/n_{i-k}) - 1}, \qquad i = 2, 3, \ldots \quad k = 1, \ldots, i-1. \tag{13.5.4}$$

The values $A_{i,k}$ then represents a numerical method of order k.

Several step sequences can be used. The classical choice used by Romberg are:

$$1, 2, 4, 8, 16, 32, 64, 128, \ldots,$$

For this sequence the denominators in (13.5.4) are

$$(n_i/n_{i-k}) - 1 = 1, 3, 7, 15, 31, \ldots.$$

If the round-off error in $y(x, h)$ has magnitude less than ϵ, then the resultant error in an extrapolated value is less than 8.26ϵ.

Example 13.5.2.

In the table below the value $A_{i,0}$ is the result for $x = 1$ of integrating the differential equation $y' = -y$ with initial condition $y(0) = 1$ and step size $h_i = 0.25 \cdot 2^{-i}$. This corresponds to taking $H = 1$ and $n_i = 4, 8, 16, 32$.

$$\begin{array}{lllllll}
A_{00} = 0.316406 & & & & & & \\
 & 27203 & & & & & \\
A_{10} = 0.343609 & & .370812 & & & & \\
 & 12465 & & -758 & & & \\
A_{20} = 0.356074 & & .368539 & & .367781 & & \\
 & 5981 & & -168 & & 12 & \\
A_{30} = 0.362055 & & .368036 & & .367868 & & .367880
\end{array}$$

We accept $A_{33} = 0.367880$ and estimate the truncation error as $|A_{32} - A_{33}| = 12 \cdot 10^{-6}$. The correct value is $y(1) = e^{-1} = 0.367879$.

The above sequence and the related

$$1, 2, 3, 4, 6, 8, 12, 16, \ldots.$$

have the advantage that for numerical quadrature, i.e., $y' = f(x)$ many function values can be saved and reused for larger n_i. However, for differential equations the most economical sequence is simply the harmonic sequence

$$1, 2, 3, 4, 5, 6, 7, 8, \ldots.$$

Above we used the result of the extrapolation after a basic step H as new starting value for the rest of the integration. This is called **active extrapolation**. Another way is **passive extrapolation**. This means that the results of extrapolation are accepted as output data, but that they are not used in the remaining of the calculation. Thus a passive extrapolation can be performed after the problem has been solved from start to finish with a sequence of step sizes. The result of Example 13.5.2 can be viewed as the result of passive extrapolation performed several times at $x = 1$.

Example 13.5.3.

See the table in Example 13.5.1. Denote by $\tilde{y}$ the result of *one* passive Richardson extrapolation:

x	$y(x, 0.1) - y(x, 0.2)$	$\tilde{y}$	$\tilde{y}(x) - y(x)$
0	0.000	1.000	0.000
0.2	0.010	1.220	-0.001
0.4	0.024	1.488	-0.004
0.6	0.043	1.814	-0.008

The accuracy in $\tilde{y}(x)$ is much better than in $y(x, 0.1)$. If one wants an improved result in an intermediate point–e.g., $x = 0.3$—then one gets a suitable correction by interpolating linearly in the second column, i.e.,

$$\tilde{y}(0.3) = y(0.3, 0.1) + \frac{1}{2}(0.010 + 0.024) = 1.348.$$

The error in $\tilde{y}(0.3)$ is -0.002.

One might think that active extrapolation should always be preferable, but with certain types of systems, passive extrapolation is better because it is numerically more stable. Note that passive extrapolation can only be used under conditions that the *changes in step size* are done in the same way for the different initial step sizes. A situation where passive extrapolation is permissible for two different initial step sizes h_0 and $h_0/2$ is illustrated in Fig. 12.6.1.

Repeated Richardson extrapolation can be used more generally to improve a numerical method. Given the differential equation 13.5.1 and a numerical method

Figure 13.5.1. *Passive extrapolation for two different initial step sizes.*

of order p, which we write $y_{n+1} = y_n + h\Phi(x_n, y_n, h)$. Denote by $y(x, h)$ the result of the numerical method at x using the step size h. Then extrapolation acn be applied if it can be shown that the global error for the method has an asymptotic expansion of the form

$$y(x,h) - y(x) = e_p(x)h^p + e_{p+1}(x)h^{p+1} + \cdots + \cdots + e_q(x)h^q + O(h^{q+1}).$$

13.5.2 The Explicit Midpoint Method

Euler's method together with repeated Richardson extrapolation is simple to use and, in some applications, also economically satisfying. Even more efficient extrapolation methods can be developed based on **symmetric** methods, whose error expansion only involve even powers of h,

$$y_h(x) - y(x) = e_p(x)h^p + e_{p+2}(x)h^{p+2} + \cdots + e_q(x)h^q + O(h^{q+2}), \qquad (13.5.5)$$

(p even). Then each extrapolation step takes the form

$$A_{i,k+1} = A_{i,k} + \frac{A_{i,k} - A_{i-1,k}}{(n_i/n_{i-k})^2 - 1}, \quad i = 2, 3 \ldots \quad k = 1, \ldots, i-1. \qquad (13.5.6)$$

and will increase the order by 2.

The **explicit midpoint method** or **leap-frog method**

$$y_{n+1} = y_{n-1} + 2hf(x_n, y_n), \qquad n = 1, 2, \ldots.$$

is a symmetric method of order two. By symmetry we mean that if we replace

$$h \leftrightarrow -h, \qquad y_{n+1} \leftrightarrow y_{n-1}$$

then $y_{n-1} = y_{n+1} - 2hf(x_n, y_n)$, i.e., we get back the original method. It is a two-step method, since it requires two values from the past, namely y_{n-1} and y_n, in the step where y_{n+1} is computed. It therefore requires a special formula for the calculation of y_1. This should be chosen so that it does not destroy symmetry, because otherwise error terms involving also odd power of h would be introduced. It was shown by W. B. Gragg in 1963 that *if Euler's method*

$$y_1 = y_0 + hf(x_0, y_0)$$

is used as starting formula then for $x = x_0 + nh$ and n even y_n has an expansion of the form

$$y(x,h) = y(x) + c_1(x)h^2 + e_2(x)h^4 + e_6(x)h^6 + \cdots.$$

If the harmonic sequence $2, 4, 6, 8, 10, 12, \ldots$ is chosen the denominators in (13.5.6) become

$$(n_j/n_{j-k})^2 - 1 = 3, 8, 15, 24, 35, \ldots.$$

One extrapolation will give a method of order four using only five evaluations of f. Note that a more accurate starting formula than Euler's method will give worse extrapolated results!

The following example shows that the stability properties of the modified midpoint method are not generally acceptable.

Figure 13.5.2. *Oscillations in the modified midpoint method solution.*

Example 13.5.4.

Apply the modified midpoint method to the equation $y' = -y$, $y(0) = 1$, with $h = 0.1$. The exact solution is $y(x) = e^{-x}$, and $y(0.1) = 0.904837$. In Fig. 12.6.2 the numerical solution corresponding to $y_1 = 0.90$ is shown by black circles, while the solution corresponding to $y_1 = 0.85$ is shown with white circles. Note that the perturbation of the initial value gives rise to growing oscillations with a growth of approximately 10% per step. This phenomenon is sometimes called weak instability.

In more realistic examples the oscillations become visible much later. For example, if $y_1 = e^{-0.1}$ correct to ten decimals, we have

The oscillations can be damped by applying the following symmetric smoothing formula for certain points where n is even

$$\hat{y}_n = \frac{1}{4}(y_{n-1} + 4y_n + y_{n+1}).$$

Because of symmetry this smoothing step will not introduce terms of odd powers and the asymptotic error expansion of $\hat{y}_n$ is again of the form (13.5.6). Another

Table 13.5.1. *Error in solution of $y' = -y$ by the modified midpoint method.*

x_n	0	0.1	...	5.0	5.1	5.2	5.3
$y(x_n)$	1	0.90484	...	0.00674	0.00610	0.00552	0.00499
y_n	1	0.90484	...	0.01803	−0.00775	0.01958	−0.01166
$y_n - y(x_n)$	0	0.00000	...	0.01129	−0.01385	0.01406	−0.01665

way of writing the smoothing step is

$$\hat{y}_n = (y_{n-1} + y_n + hf(x_n, y_n)).$$

This finally leads to the following **modified midpoint method** as the basis for Richardson extrapolation.

Algorithm 13.5.1

Let N be even, take $h = H/N$, and compute

$$\begin{aligned} y_1 &= y_0 + hf(x_0, y_0), \\ y_{n+1} &= y_{n-1} + 2hf(x_n, y_n), \quad n = 1, 2, \ldots, N; \\ \hat{y}_N &= \frac{1}{2}(y_{N-1} + y_N + hf(x_N, y_N)). \end{aligned} \tag{13.5.7}$$

Again it can be shown that $\hat{y}_N$ has an error expansion of the form (13.5.5). A simple proof of this is based on rewriting the method as a one-step algorithm in terms of odd and even indices,

$$u_k = y_{2k}, \qquad v_k = y_{2k+1}.$$

The method can then be written $u_0 = v_0 = y_0$,

$$\begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ v_k \end{pmatrix} + 2h \begin{pmatrix} f\big(x_{2k} + h, v_k + hf(x_{2k}, u_k)\big) \\ \frac{1}{2}\Big(f\big(x_{2(k+1)}, u_{k+1}\big) + f(x_{2k}, u_k)\Big) \end{pmatrix},$$

$k = 0, 1, \ldots, N/2$. This mapping from (u_k, v_k) to (u_{k+1}, v_{k+1}) can be shown to be *symmetric* since exchanging

$$u_{k+1} \leftrightarrow u_k, \qquad v_{k+1} \leftrightarrow v_k, \qquad h \leftrightarrow -h, \qquad x_k \leftrightarrow x_{k+2}$$

gives back the original formula.

Example 13.5.5.

For the initial-value problem $y' = y^2$, $y(0) = 0.25$ we get with $h = 0.5$, $N = 2$, and

$$\hat{y}_2 = \frac{1}{2}(0.28125 + 0.329101562 + 0.054153919) = 0.332252741.$$

n	x_n	y_n	$hf(x_n, y_n)$
0	0	0.25	0.03125
1	0.5	0.28125	0.039550781
2	1.0	0.329101562	0.054153919

Performing similar computations for $N = 4{,}8$ and forming the extrapolating scheme we get

$$
\begin{array}{lccc}
A_{00} = 0.3322\ 52741 & & & \\
 & 265,468 & & \\
A_{10} = 0.3330\ 49146 & & 0.333314614 & \\
 & 70,726 & & 1,162 \\
A_{20} = 0.3332\ 61325 & & 0.3333\ 32051 & 0.3333\ 33213
\end{array}
$$

Since the exact solution of the problem is $y(x) = 1/(4 - x)$, we have $y(1) = 0.333333333$. The error in the extrapolated error is thus $1/600$ of the error in $y(1, 1/8)$.

PROBLEMS

1. Given the initial value problem $y' = 1 + x^2y^2$, $y(0) = 0$. Compute $y(0.5)$ by Euler's method with repeated Richardson extrapolation. Use four decimals.

2. In a computation with Euler's method, the following results were obtained with various step sizes:

$h = 0.05$	$h = 0.1$	$h = 0.2$
1.22726	1.22595	1.22345

   Compute a better value by extrapolation.

3. (a) Determine an explicit expression for y_n when Euler's method is applied to the test problem

$$y' = -y, \qquad y(0) = 1.$$

   (b) For which values of h is the sequence $\{y_n\}_0^\infty$ bounded?
   (c) Compute $\lim_{h \to 0}(y(x, h) - e^{-x})/h$.

13.6 Second-Order Equations and Boundary Problems

13.6.1 Second-Order Differential Equations

Equations of the form

$$y'' = f(x, y, y') \tag{13.6.1}$$

with initial conditions

$$y(x_0) = y_0, \qquad y'(x_0) = z_0,$$

are often encountered in, e.g., astronomy and mathematical physics. The equation can be expressed as a set of two simultaneous first order differential equations

$$\begin{pmatrix} y' \\ z' \end{pmatrix} = \begin{pmatrix} z \\ f(x,y,z) \end{pmatrix}, \qquad \begin{pmatrix} y(x_0) \\ z(x_0) \end{pmatrix} = \begin{pmatrix} y_0 \\ z_0 \end{pmatrix}. \tag{13.6.2}$$

This system can be solved by the methods developed previously in this chapter.

Often a simplified version of (13.6.1) occurs, where the right hand side does not depend on y',

$$y'' = f(x,y). \tag{13.6.3}$$

For such systems many special methods have been developed. A simple finite-difference approximation to (13.6.3) is obtained by replacing the derivatives in the differential equation and initial condition by symmetric difference approximations. If we put $f_n = f(x_n, y_n)$, a method defined by the following relations:

$$y_{n+1} - 2y_n + y_{n-1} = h^2 f_n, \qquad y_1 - y_{-1} = 2hz_0 \tag{13.6.4}$$

This method is the simplest member of the **Störmer family** of methods, and we shall call it the **explicit central difference method**. The local error is obtained from the Taylor expansion

$$y(x_n+h)-2y(x_n)+y(x_n-h) = h^2y''(x_n)+\frac{h^4}{12}y^{(4)}(x_n)+\frac{h^6}{360}y^{(6)}(x_0)+\cdots. \tag{13.6.5}$$

The value y_{-1} can be eliminated by means of the first equation with $n = 0$. The starting procedure is, therefore

$$y_1 = y_0 + hz_0 + \frac{1}{2}h^2 f_0,$$

which is just the first terms in the Taylor-expansion of y_1. Then y_{n+1}, $n \geq 1$ can be computed successively by means of $y_{n+1} = 2y_n - y_{n-1} + h^2 f_n$. Note that at each step there is an addition of the form $O(1) + O(h^2)$; this gives unfavorable rounding errors when h is small. If we put $u_{i-1/2} = (y_i - y_{i-1})/h$ and rewrite the method as

$$\frac{1}{h}(y_{n+1} - y_n) = \frac{1}{h}(y_n - y_{n-1}) + hf_n$$

then the method can be defined by the formulas $u_{1/2} = z_0 + \frac{1}{2}hf_0$, and

$$y_n = y_{n-1} + hu_{n-1/2}, \qquad u_{n+1/2} = u_{n-1/2} + hf_n, \qquad n \geq 1. \tag{13.6.6}$$

This **summed form** of the method is mathematically equivalent, but numerically superior to the difference method (13.6.5). An alternative is to store y in double precision, while single precision is used in the computation of f, which is usually the most time-consuming part of the work. If such **partial double precision** is used, then the advantage of the summed form is reduced. See also Example 2.3.4 for a solution when double precision is not available.

Because of symmetry the following expansion holds for the explicit central difference method

$$y(x,h) = y(x) + c_1(x)h^2 + c_2(x)h^4 + c_3(x)h^6 + \cdots$$

(As usual, the rounding errors are ignored in this expansion, and certain conditions about the differentiability of f have to be satisfied). The expansion shows that the global error is $O(h^2)$ and that higher accuracy can be obtained with Richardson extrapolation according to the scheme (13.6.6).

The method (13.6.4) and its summed form (13.6.6) can be extended to equations of the form of (13.6.1) if one puts

$$y'_n \approx \frac{1}{2h}(y_{n+1} - y_{n-1}) = \frac{1}{2}(u_{n+1/2} + u_{n-1/2}),$$

but in this case the method becomes implicit.

Note that another extrapolation method is obtained by applying the modified midpoint method, Algorithm 13.5.1, directly to the first order system (13.6.2). This avoids the problem with rounding errors, which motivated the summed form of Störmer's method.

There are other ways to improve the order of accuracy of the method (13.6.4). This method is a special case of multistep methods of the form

$$\sum_{i=0}^{k}(\alpha_i y_{n+i} - h^2\beta_i f_{n+i}) = 0,$$

cf. (13.4.1). The family of methods for which $\sum_{i=0}^{k}\alpha_i y_{n+i} = y_{n+2} - 2y_{n+1} + y_n$ is frequently referred to as **Störmer-Cowell methods**. The best known such method is **Numerov's method** or the implicit difference correction method:

$$y_{n+1} - 2y_n + y_{n-1} = h^2\Big(f_n + \frac{1}{12}(f_{n+1} - 2f_n + f_{n-1})\Big) \tag{13.6.7}$$

A sufficiently accurate starting procedure is obtained from the formula

$$2hy'_0 = 2hz_0 = (y_1 - y_{-1}) - \frac{1}{6}h^2(f_1 - f_{-1}) + O(h^5),$$

see Problem 1.

By expressing the higher derivatives in (13.6.5) in terms of central differences we obtain the expansion

$$y_{n+1} - 2y_n + y_{n-1} = h^2\Big(f_n + \frac{1}{12}\Delta^2 f_{n-1} - \frac{1}{240}\Delta^4 f_{n-2} + \cdots)\Big).$$

Numerov's method is obtained by taking the first two terms of the right hand side. Note that taking further terms is not practical since these contains unknown expressions f_{n+2} etc.

In Numerov's method one can proceed in the following way. If we put

$$v_i = y_i - \frac{1}{12}h^2 f_i,$$

then the difference equation then takes the form, similar to (13.6.4)

$$v_{n+1} - 2v_n + v_{n-1} = h^2 f_n, \tag{13.6.8}$$

although in order to compute $f_n = f(x_n, y_n)$ one has to solve for y_n from the equation

$$y_n - h^2 \frac{1}{12} f(x_n, y_n) = v_n.$$

The summed form is similar to (13.6.6) with y replaced by v and a more accurate starting procedure. The error expansion for Numerov's method has the form

$$y(x, h) = y(x) + c_1(x)h^4 + c_2(x)h^6 + \cdots$$

and Richardson extrapolation can be applied.

If the differential equation is nonlinear, Numerov's method requires some iterative method. For the linear case, see Problem 2. Starting values can then be obtained from the explicit method obtained by using a **backward difference correction**

$$y_{n+1} - 2y_n + y_{n-1} = h^2 \Big(f_n + \frac{1}{12}(f_n - 2f_{n-1} + f_{n-2}) \Big) \tag{13.6.9}$$

The estimate here is less accurate, and therefore the global error of this method is only $O(h^3)$.

In the **deferred difference correction method** due to L. Fox one first computes a sequence y_n by solving the difference equation (13.6.4). Using this sequence one computes a correction term

$$C_n = h^2 \frac{1}{12}(f_{n+1} - 2f_n + f_{n-1}).$$

An improved solution $\hat{y}_n$ is then obtained by solving the difference equation

$$\hat{y}_{n+1} - 2\hat{y}_n + \hat{y}_{n-1} = h^2 f(x_n, \hat{y}_n) + C_n.$$

The procedure can be iterated and more sophisticated formulas for the correction C_n can be used. The global error of the solution produced by this methods also is $O(h^4)$. The deferred correction method is very useful in solving certain boundary value problems, see Sec. 13.6.2.

13.6.2 Boundary Value Problems

In this section we shall consider two point **boundary-value problems** for a second order scalar differential equation

$$y'' = f(x, y, y') \tag{13.6.10}$$

with **boundary** conditions

$$y(a) = \alpha \qquad y(b) = \beta, \tag{13.6.11}$$

We assume that f has continuous partial derivatives of arbitrary order in the *closed* interval $[a, b]$. More generally, we also consider boundary-value problems for a system of first order equations

$$y' = f(x, y), \qquad y \in \mathbf{R}^{s \times s}, \tag{13.6.12}$$

with boundary conditions given in two points a and b

$$r(y(a), y(b)) = 0, \tag{13.6.13}$$

where $r(y, z)$ is a vector of s, possibly nonlinear, functions. Often the boundary conditions are linear

$$r \equiv Ay(a) + By(b) - c = 0.$$

The boundary value problem (13.6.10)–(13.6.11) can be reduced to this form, by the standard substitution $y_1 = y$, $y_2 = y_1'$. The boundary conditions correspond to

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, c = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

In contrast to the initial-value problem, it can happen that the boundary-value problem has several or even no solution. Sufficient conditions given in the literature for the existence of a unique solution are often astonishingly poor. For example, it is often assumed in the case of linear boundary conditions that the matrix $A + B$ is nonsingular, which is not the case in most common applications. For the practically important special case of a single second order equation, better results exist.

In **free boundary problems** b is unknown, and we have $s + 1$ equations of the form (13.6.13). We introduce a new independent variable $t = (x - a)/(b - a)$, and put

$$z_{s+1} = b - a, \qquad x = a + tz_{s+1}, \qquad 0 \le t \le 1,$$

Then this reduces to a standard boundary problem for $z(t) = y(a + tz_{s+1})$. We have the $s + 1$ differential equations

$$\frac{dz}{dt} = f(a + tz_{s+1}, z(t))z_{s+1}, \qquad \frac{dz_{s+1}}{dt} = 0,$$

with boundary conditions which now can be written $\hat{r}(z(0), z(1)) = 0$. Eigenvalue problems are considered in Sec. 13.6.5.

In the following two different types of methods will be described, **shooting** methods and **finite difference** methods.

13.6.3 The Shooting Method

There is also an initial-value problem for (13.6.10):

$$y'' = f(x, y, y'), \qquad y(a) = \alpha \qquad y'(a) = \gamma, \tag{13.6.14}$$

If γ is fixed, then the value of y at $x = b$ can be considered as a function of γ—say, $g(\gamma)$. The boundary-value problem (13.6.10)–(13.6.11) can then be written

$$g(\gamma) = \beta. \tag{13.6.15}$$

In the shooting method, one solves this equation with, for example, the secant method; see Sec. 5.4. One guesses a value γ_0, and computes (approximately) $g(\gamma_0)$ by solving the initial-value problem of (13.6.14) using one of the methods described earlier in this chapter. One then chooses another value, γ_1, computes $g(\gamma_1)$ in the same way, and then, iterates according to

$$\gamma_{k+1} = \gamma_k - \big(g(\gamma_k) - \beta\big)\frac{\gamma_k - \gamma_{k-1}}{g(\gamma_k) - g(\gamma_{k-1})}, \qquad k = 1, 2, \ldots. \tag{13.6.16}$$

One can show that $g(\gamma)$ is linear in γ when (13.6.10) is a linear differential equation, even if the coefficients depend on x. In this case, γ_2 is the solution to to (13.6.15)—aside from the discretization errors and rounding errors in the computation of $g(\gamma_0)$ and $g(\gamma_1)$.

Note that there are several variations of the shooting method. We can, e.g., consider the initial value problem $y(b) = \beta$, $y'(b) = \gamma$, integrate in reverse time direction and match the boundary condition at $x = a$. This is called **reverse** shooting. Another possibility is to integrate from *both* boundaries, and matching the solutions at an interior point $x = m$ by continuity condition on $y(x)$. The latter approach is similar to the idea of multiple shooting described in detail below.

The shooting method can also be applied to the more general boundary-value problem (13.6.12)–(13.6.13). Let $y(x, p)$ be the solution to the differential equation $y' = f(x, y)$ with *initial conditions*

$$y(a) = p, \qquad p \in \mathbf{R}^{s\times s}.$$

Then the boundary-value problem is equivalent to the system of s equations

$$F(p) \equiv r(p, y(b, p)) = 0. \tag{13.6.17}$$

If the differential equation and boundary conditions are linear, then $y(x, p)$ and $F(p)$ become affine functions of p. Hence (13.6.17) becomes a linear system.

$$Ap + By(b, p) - c = 0. \tag{13.6.18}$$

In this case the problem has a unique solution, unless the corresponding homogeneous problem has a "non-trivial" solution.

The following example shows that the initial-value problem is ill-conditioned, even when the boundary-value problem is well-conditioned.

Example 13.6.1. (Stoer-Bulirsch)

Consider the system

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ 110 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

with boundary-values

$$y_1(0) = 1, \qquad y_1(10) = 1.$$

The general solution to the initial value problem $y_1(0) = 1$, $y_1'(0) = -10 + p$ is

$$y(x,p) = (1 - \frac{p}{21})e^{-10x}\begin{pmatrix} 1 \\ -10 \end{pmatrix} + \frac{p}{21}e^{11x}\begin{pmatrix} 1 \\ 11 \end{pmatrix}.$$

Hence $y_1(10) = 1$ corresponds to

$$\frac{p}{21} = \frac{1 - e^{-100}}{e^{110} - e^{-100}} \approx 3.5 \cdot 10^{-47}.$$

With floating point arithmetic one may well obtain, say $y_1'(0) = -10 + 10^{-9}$ instead; this would give $y_1(5) \approx e^{55}10^{-9}/21 \approx 3.7 \cdot 10^{13}$ instead of $2e^{-55} \approx 2.6 \cdot 10^{-24}$.(!!)

Example 13.6.2. (Troesch)

The exact solution of the boundary value problem

$$y'' = \lambda \sinh(\lambda y), \quad y(0) = 0, \quad y(1) = 1,$$

becomes infinite for $x \approx 1.03$ if $\lambda = 5$. The solution of the corresponding initial value problem $y(0) = 0$, $y'(0) = p$ becomes infinite for $x < 1$ if $p > 8e^{-\lambda} \approx 0.054$. The correct value is $p = 4.57504614 \cdot 10^{-2}$. Obviously it may be very difficult to find a sufficiently good initial value for p.

In such cases, the method described in Sec. 13.6.4 can be more advantageous. Another possibility is to use the **multiple shooting** method. In multiple shooting the interval $[a, b]$ is divided into m subintervals $[x_{i-1}, x_i]$, $i = 1, \ldots, m$,

$$a = x_0 < x_1 < \cdots x_m = b.$$

(Here m should be much smaller than the number of grid points needed in the numerical method for solving the initial value problem.) Let $y(x; x_k, p_k)$ be the solution of the initial value problem

$$y' = f(x,y), \qquad y(x_k) = p_k, \quad x \in [x_k, x_{k+1}],$$

$k = 0, 1, \ldots, m-1$. These m initial value problems are solved simultaneously. The continuity conditions,

$$F_i(p_i, p_{i+1}) = y(x_{i+1}; x_i, p_i) - p_{i+1} = 0, \quad i = 1, \ldots, m-1.$$

now appear as boundary conditions, in addition to the condition

$$F_m(p_1, p_m) = r(p_1, p_m).$$

Boundary and continuity conditions can now be written $F(p) = 0$. The Jacobian is

$$F'(p) = \begin{pmatrix} G_1 & -I & 0 & \cdots & 0 & 0 \\ 0 & G_2 & -I & \cdots & 0 & 0 \\ 0 & 0 & G_3 & \cdots & 0 & 0 \\ \cdots & & & & & \\ 0 & 0 & 0 & \cdots & G_{m-1} & -I \\ A & 0 & 0 & \cdots & 0 & B \end{pmatrix},$$

where

$$G_k = \frac{\partial F_k}{\partial p_k}, \qquad A = \frac{\partial r}{\partial p_1}, \qquad B = \frac{\partial r}{\partial p_m}.$$

Several decisions are to be made in the application of multiple shooting method:

1. Choice of starting trajectory $\hat{y}(x)$, e.g., some function that satisfies the boundary conditions.

2. Subdivision of the interval; it is often suitable to choose x_{i+1} from the initial approximation $\hat{y}(x)$ such that

$$\|y(x_{i+1}; x_i, \hat{y}(x_i)) - \hat{y}(x_{i+1})\| \approx \|\hat{y}(x_{i+1})\|.$$

3. Choice of iterative method, e.g., some modified Newton method. Pivoting for size can be essential in the solution of the linear systems encountered in each iteration!

Example 13.6.3.

For the problem in Example 13.6.2, with $\lambda = 5$ we can choose $\hat{y}(x)$ as linear function. A better initial solution can be determined form the linearized problem $y'' = 25y$.

(This is a particular case of the problem defined by (13.6.19) and (13.6.21) in the next section). When shooting is applied to this system, one obtains a system of $2p$ equations for the vector $y(0)$. The system is nonlinear if the differential equation is nonlinear. The techniques of Sec. 12.2.4 which do not require derivatives can then be applied. It is also possible to use techniques where derivatives are needed, e.g., Newton's method, but then one has to solve also the variational equation of the system $y' = f(y,t)$; see Sec. 13.1.2.

13.6.4 The Finite Difference Method

We first consider the boundary value problem (13.6.10)–(13.6.11) for a single second order equation. Divide the interval $[a,b]$ into N equal parts and put $h = (b-a)/N$. Let y_i denote the desired estimate of $y(x_i)$, $x_i = a + ih$. Replace derivatives in the differential equation as in Sec. 13.6.1 by symmetric difference approximations

$$vy'_n \approx \frac{y_{n+1} - y_{n-1}}{2h}, \qquad y''_n \approx \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2}.$$

In this way, the differential equation is transformed into a nonlinear system of equations

$$y_{n+1} - 2y_n + y_{n-1} = h^2 f_n, \qquad n = 1, 2 \ldots, N-1,$$

where

$$f_n = f\Big(x_n, y_n, \frac{y_{n+1} - y_{n-1}}{2h}\Big).$$

Together with the boundary conditions $y_0 = \alpha$, $y_N = \beta$ this system can be written in matrix form

$$Ay = h^2 f - r, \tag{13.6.19}$$

where

$$A = \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 & 0 \\ 0 & 1 & -2 & \cdots & 0 & 0 \\ \cdots & & & & & \\ 0 & 0 & 0 & \cdots & -2 & 1 \\ 0 & 0 & 0 & \cdots & 1 & -2 \end{pmatrix}, \qquad r = \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \vdots \\ 0 \\ \beta \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{pmatrix}.$$

Thus A is a **band matrix** (and in fact a tridiagonal matrix in this example.) If the differential equation is linear, then the system of equations is linear and tridiagonal—thus it can be solved by very little work. Note that the matrix of the system is not A, since f depends on y and y'.

For the error, even in the nonlinear cases, we have

$$y(x,h) = y(x) + c_1(x)h^2 + c_2(x)h^4 + c_3(x)h^6 + \cdots,$$

and Richardson extrapolation can be used with correction terms $\Delta/3, \Delta/15, \Delta/63, \ldots$.

Example 13.6.4.

The boundary-value problem

$$y'' + y = x, \qquad y(0) = 1, \qquad y(\frac{1}{2}\pi) = \frac{1}{2}\pi - 1$$

has the exact solution $y(x) = \cos x - \sin x + x$. The difference equation becomes

$$y_{n+1} - 2y_n + y_{n-1} - h^2 y_n = h^2 x_n, \qquad n = 1, 2 \ldots, N-1.$$

The solution of the system of equations for $N = 5$ and $N = 10$ and the result $\hat{y}$ of Richardson extrapolation is given by the following table:

$2x/\pi$	$y(x)$	$y(x, 0.1\pi)$	$y(x, 0.05\pi)$	$\Delta/3$	$\hat{y}(x)$	error
0	1.000000	1.000000	1.000000	0	1.000000	0
0.1	0.988334		0.988402			
0.2	0.956199	0.956572	0.956291	−94	0.956197	−2
0.3	0.908285		0.908337			
0.4	0.849550	0.849741	0.849597	−48	0.849549	−1
0.5	0.785398		0.785398			
0.6	0.721246	0.721056	0.721199	+48	0.721247	1
0.7	0.662541		0.662460			
0.8	0.614598	0.614224	0.6145505	94	0.614599	1
0.9	0.582463		0.582395			
1.0	0.570796	0.570796	0.570796	0	0.570796	0

The methods for improving accuracy described in Sec. 13.6.1, can also be used for boundary value problems. In particular, for equations of the form $y'' = f(x, y)$ Cowell's method gives $O(h^4)$ accuracy with about the same amount of computation as the $O(h^2)$-method just described.

If the equation is nonlinear, then one can use some modified Newton method. As a first approximation one can—in the absence of a better proposal— choose a linear function satisfying the boundary condition, i.e.

$$y_i^{(0)} = \alpha + (\beta - \alpha)/N, \qquad i = 0, 1, \ldots, N.$$

In boundary value problems one often encounters differential expressions of the form

$$\frac{d}{dx}\Big(p(x)\frac{dy}{dx}\Big). \tag{13.6.20}$$

These can be approximated at x_n by

$$\frac{1}{h}\Big(p_{n+1/2}\big(\frac{y_{n+1} - y_n}{h}\big) - \big(p_{n-1/2}\big(\frac{y_n - y_{n-1}}{h}\big)\Big) \tag{13.6.21}$$

with global error of of the form $c_1(x)h^2 + c_2(x)h^4 + \ldots$.

With boundary conditions of the form $b_0 y(b) + b_1 p(b) y'(b) = b_2$ one can introduce an *extra point*, $x_{N+1} = b + h$, and approximate the condition by

$$b_0 y_N + \frac{p_1(y_{N+1} - y_{N-1})}{2h} = b_2,$$

and similarly for the condition at $x = a$. One can also put b *between* two grid points. The form of the boundary conditions only affect the first and last rows in the matrix A in (13.6.19).

For systems of first-order equations $y' = f(x, y)$ the trapezoidal method can be used

$$y_{n+1} - y_n = \frac{1}{2}h\big(f(y_n, x_n) + f(y_{n+1}, x_{n+1})\big), \qquad n = 1, 2, \ldots, N.$$

With linear boundary conditions $Ay(a) + By(b) = c$ one obtains a nonlinear system of simple structure.

If no natural intial approximation is available an **imbedding technique**, described in Sec. 12.1.7, is often useful. One can introduce a aprameter in the differential equation, or sometimes it may be sufficient to use the step size h of the difference method as a parameter, i.e., one starts with a very crude grid, and refines it successively in using, e.g., the previously obtained solution and interpolation as initial approximation for the solution on the next finer grid.

13.6.5 Eigenvalue Problems

Many important eigenvalue problems in applied mathematics have the form

$$(p(x)y')' - q(x)y + \lambda r(x)y = 0, \tag{13.6.22}$$

subject to homogeneous boundary conditions

$$a_0 y(a) - a_1 p(a) y'(a) = 0, \qquad b_0 y(b) + b_1 p(b) y'(b) = 0, \tag{13.6.23}$$

where λ is a scalar parameter to be determined. This is called a **Sturm-Liouville problem**. Note that $y(x)$ has to be normed to be uniquely determined.

A more general form of the eigenvalue problem is

$$y' = f(x, y, \lambda), \qquad Ey(a) + Fy(b) - c = 0. \tag{13.6.24}$$

By introducing

$$y_0(x) = \lambda \text{ (constant)}, \qquad y_{s+1} = \int_a^x y^T y dx$$

and the differential equations

$$y_0' = 0, \qquad y_{s+1}' = y^T y,$$

this can be reduced to the standard form above for $\tilde{y} = (y_0, y_1, \ldots, y_{s+1})^T$.

Example 13.6.5.

For which value of λ does the boundary-value problem

$$y'' + \lambda y = 0, \qquad y(0) = y(1) = 0,$$

have solutions other than $y = 0$?

The general solution to the differential equation is

$$y(x) = a \cos \mu x + b \sin \mu x, \qquad \mu = \sqrt{\lambda},$$

From $y(0) = 0$ it follows that $a = 0$. Further since $y(1) = 0$ we have $\mu = n\pi$, $n = 0, \pm 1, \pm 2, \ldots$. Thus the **eigenvalues** are

$$\lambda = n^2 \pi^2, \qquad n = 1, 2, 3, \ldots.$$

Note that $n = 0$ gives the trivial solution $y = 0$; $n = -k$ gives the same solution as $n = k$. The solution of the differential equation when λ is an eigenvalue are called **eigenfunctions**. In this example the eigenfunctions $y(x) = b \sin n\pi x$ belong to the eigenvalue $\lambda = n^2 \pi^2$.

Eigenvalue problems occur in most areas of classical and modern physics (for eigen-vibrations, etc.) Example 13.6.5 comes up. e.g., in the computation of wave numbers for a vibrating string. Some other important problems in partial differential equations from physics can be reduced, by separation of variables, to eigenvalue problems for ordinary differential equations.

The difference method according to Sec. 13.6.4 gives an approximation to the eigenvalues which satisfies

$$\lambda(h) = \lambda + c_2 h^2 + c_3 h^3 + c_4 h^4 + \cdots,$$

where c_3 is zero in some cases (among others, in Example 13.6.3). Note the regularity assumptions made in Sec. 13.6.1.

Example 13.6.6.

For the problem in Example 13.6.5, the difference method with $h = 1/3$ gives the system of equations

$$\begin{aligned} -2y_1 + y_2 + \lambda h^2 y_1 &= 0 \\ y_1 - 2y_2 + \lambda h^2 y_2 &= 0. \end{aligned}$$

This is a homogeneous system of equations, two equations and two unknowns, which has a nontrivial solution if and only if λh^2 is an eigenvalue of the matrix

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}.$$

Thus $\lambda h^2 - 2 = \pm 1$, with solutions

$$\begin{aligned} \lambda_1 &= 9, \quad (\text{Exact value} \quad \pi^2 = 9.8696), \\ \lambda_2 &= 27, \quad (\text{Exact value} \quad 4\pi^2 = 39.48). \end{aligned}$$

The higher eigenvalues cannot even be estimated using such a course grid.

By similar calculations with various values of h we get the following results for the smallest eigenvalue:

h			Richardson	Error
$\frac{1}{2}$	8			
$\frac{1}{3}$	9	$\Delta/\frac{5}{4} = 0.8$	9.8	-0.0696
$\frac{1}{4}$	9.3726	$\Delta/\frac{7}{9} = 0.4791$	9.8517	-0.0179

A second Richardson extrapolation will cancel the $O(h^4)$ error term and gives

$$\lambda = 9.8517 + \frac{1}{3}0.0517 = 9.8689$$

correct to four decimal places. Motivate this extrapolation!

There are computational methods for solving eigenvalue problems for much larger matrices; see Sec. 10.10. By using Richardson extrapolation one can, however, obtain good accuracy with a reasonable number of points. The same general procedure can be used with differential expressions of the form of (13.6.22) with the difference expression of the form of (13.6.21).

The shooting method can also be used on eigenvalue problems, see Problems 3 and 4 below.

PROBLEMS

1. (a) Put

$$hy_0' = A\mu\delta(y_0 + Bh^2 y'') + R_T,$$

and determine A and B so that the remainder term is of as high order as possible. Give an asymptotically correct expression for R_T.

(b) Let $y(x)$ be the solution to the differential equation problem

$$y'' = f(x)y, \qquad y(0) = \alpha, \qquad y'(0) = \beta.$$

Put $y(kh) = y_k$, $k = -1, 0, 1, \ldots$. With Numerov's formula and the formula in (a), one gets a system of equations for determining y_{-1} and y_1. Give an asymptotically correct expression for the error in the determination of y_1 obtained from this system of equations.

(c) Apply the formula in (b) to compute $y(0.1)$ with $h = 0.1$ in the case

$$y'' = e^x y, \qquad y(0) = 0, \qquad y'(0) = 1.$$

2. For what positive values of the step size h does Numerov's method produce bounded solutions when applied to the differential equation

$$y'' = -y?$$

3. Consider the application of Numerov's method to the linear equation

$$y'' = p(x)y + q(x).$$

Show that with the notation of (13.6.8)

$$f_n = \big(p(x_n)v_n + q(x_n)\big)/\big(1 - \frac{h^2}{12}p(x_n)\big).$$

4. Write programs for the solution of $y'' = f(x, y)$ with the difference method (13.6.4) and the summed form (13.6.5). Apply them to the equation $y'' = -y$, compare with the exact solution, and print out the errors. Perform a series of numerical experiments in order to get acquainted with the accuracy obtained with Richardson extrapolations and see the effect of rounding errors.

5. Consider the initial-value problem

$$y'' = (1 - x^2)y, \qquad y(0) - 1, \qquad y'(0) = 0.$$

(a) Show that the solution is symmetric about $x = 0$.

(b) Determine $y(0.4)$ using Numerov's method without any special start formula, with step lengths $h = 0.2$ and $h = 0.4$. Perform Richardson extrapolation.

6. The function $y(x)$ is defined by the problem

$$y''' = yx, \qquad y(0) = 0, \qquad y'(0) = 1, \qquad y''(0) = 1.$$

To determine $y(1)$ one puts

$$y'''(x_n) \approx \frac{\mu\delta^3 y_n}{h^3} = \frac{y_{n+2} - 2y_{n+1} + y_{n-1} - y_{n-2}}{2h^3},$$

where $x_n = nh$, and derives the recursion formula $y_0 = 1$,

$$y_{n+2} = 2y_{n+1} + 2h^3 x_n y_n - 2y_{n-1} + y_{n-2}, \qquad n \geq 2.$$

The initial values y_1, y_2, y_3 needed in the recursion formula can be obtained, for example, by using the classical Runge-Kutta method with step h.

(a) What is the order of accuracy of the method?

(b) We give below computed values $y(h_k) = y(1, h_k)$ using the above method and step lengths $h_0 = 0.1$, $h_k = h_0/2^k$, $k = 0, 1, \ldots, 7$:

$$1.54568, 1.54591, 1.54593, 1.54592, 1.52803, 1.50045, 1.51262, 1.48828.$$

What is puzzling about these results? Explain what the trouble is.

7. In using the shooting method on the problem

$$y'' = \frac{1}{2}y - \frac{2(y')^2}{y}, \qquad y(0) = 1, \qquad y(1) = 1.5,$$

the following results are obtained using $y'(0) = p$:

$$p = 0: \quad y(1) = 1.543081, \qquad p = -0.05: \quad y(1) = 1.425560.$$

What value of p should be used on the next "shot"?

8. (From Collatz [3, 1960].) The displacement u of a loaded beam of length $2L$ satisfies under certain assumptions the differential equation

$$\frac{d^2}{ds^2}\left(EI(s)\frac{d^2u}{ds^2}\right) + Ku = q(s), \qquad -L \leq s \leq L,$$

with boundary conditions

$$u''(-L) = u'''(-L) = 0, \qquad u''(L) = u'''(L) = 0.$$

For a certain beam we have:

$$I(s) = I_0(2 - (s/L)^2), \quad q(s) = q_0(2 - (s/L)^2), \quad K = 40EI_0/L^4.$$

One wants to know the displacement at $s = 0$.

(a) Introduce more suitable variables for this problem, and write it as a system of two second-order equations with boundary conditions. Prove, and make use of the symmetry property $u(s) = u(-s)$. (Assume that the system has a unique solution.)

(b) Propose a difference approximation for this problem, where $h = L/N$, N an arbitrary positive integer. Count the number of equations and unknowns. Verify that for $N = 1$ one gets

$$u(0) = 13/280c = 0.046429c, \qquad c = q_0L^4/(EI_0).$$

(c) In a computation one obtained

$$N = 2: \quad u(0) = 0.045752c, \qquad N = 5: \quad u(0) = 0.045332c.$$

Perform Richardson extrapolation, first with $N = 1$ and $N = 2$, and then with $N = 2$ and $N = 5$.

(d) How should one number the variables to get a small band width in the matrix?

9. (a) Compute approximately, the smallest eigenvalue λ for which the problem

$$\frac{d}{dx}\left((1+x^2)\frac{dy}{dx}\right) + \lambda y = 0, \qquad y(-1) = y(1) = 0$$

has a nontrivial solution. Use a difference method with step size $h = 2/3$ and then $h = 2/5$, and perform Richardson extrapolation. (*Hint*:Utilize the symmetry of the eigenfunctions about $x = 0$.)

(b) Use the same difference method with $h = 1/5$ to solve the differential equation with initial-values $y(0) = 1$, $y'(0) = 1$, for $\lambda = 3.50$ and $\lambda = 3.75$. Use inverse interpolation to compute the intermediate value of λ for which $y(1) = 0$, and make a new computation with the value of λ so obtained. Then improve the estimate of the eigenvalue using Richardson extrapolation.

(c) Compute the next smallest eigenvalue in the same way.

10. One seeks the solution of the eigenvalue problem

$$\frac{d}{dx}\left(\left(\frac{1}{1+x}\right)\frac{dy}{dx}\right) + \lambda y = 0, \qquad y(0) = y(1) = 0$$

by integrating, for a few values of λ, an equivalent system of two first-order differential equations with initial values $y(0) = 0$, $y'(0) = 1$, with the classical Runge-Kutta method. Computations using three different λ, each with three different step sizes, gave the following values of $y(1) \cdot 10^4$:

h \ λ	6.76	6.77	6.78
$\frac{1}{10}$	16.126	5.174	−5.752
$\frac{1}{15}$	16.396	4.441	−6.490
$\frac{1}{20}$	15.261	4.304	−6.627

Compute, for each value of λ, a better value for $y(1)$ using Richardson extrapolation. Then use inverse interpolation to determine the value of λ which gives $y(1) = 0$.

11. The eigenvalue for a stretched circular membrane is

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{du}{dr}\right) + \lambda u = 0,$$

with $u(1) = 0$, $u(0)$ and $u''(0)$ finite.

(a) Set up a difference equation with grid points

$$r_i = \frac{2i+1}{2N+1} = r_0 + ih, \qquad i = -1, 0, 1, \ldots, N, \qquad h = \frac{2}{2N+1}.$$

(Thus the origin lies between the two grid points r_{-1} and r_0.)

(b) Determine the smallest eigenvalue, first for $N = 1$ and then for $N = 2$. Perform Richardson extrapolation under the assumption that the global error is proportional

to h^2.

(c) For large values of N, one would like to use a standard program for computing eigenvalues of symmetric matrices. How does one put the above problem in a suitable form?

Remark: The origin is a singular point for the differential equation, but not for its solution. The singularity causes no trouble in the above treatment.

12. Show that (13.6.15) is a first-degree equation when (13.6.10) is a linear differential equation, even if the coefficients depend on x. Show also that multiple shooting yields a linear system for the determination of $y(0)$ when the differential equation is linear.

13.7 Qualitative Theory of ODEs

13.7.1 Some Results from the Qualitative Theory of ODEs

The topic of the qualitative theory of differential equations is how to draw conclusions about some essential features of the solutions of a system of ODEs, even if the solutions cannot be expressed explicitly in analytic form. In a way it is the opposite to the study of ODEs by numerical methods. Nevertheless the ideas and results from the qualitative theory can be very useful in many ways, for example:

- at the planning of numerical experiments,
- for an intelligent interpretation of the results of a simulation,
- for finding out, whether an unexpected result of a simulation is reasonable, or due to a bug in the program, or due to the use of too large time steps, or some other cause.

The reader must find his own switch between computational and analytical techniques, but some ideas from the qualitative theory of ODEs are useful in the bag of tricks.

All ODE systems in this section are assumed to satisfy a Lipschitz condition etc., so that there is no trouble about existence and uniqueness. We begin by a simple and useful example.

Example 13.7.1.

Consider a *single* autonomous ODE, $y' = f(y)$, where the graph of $f(y)$ is shown in the left part of Fig. 13.7.1. The equation has 3 critical points. Since $y(t)$ increases if $f(y) > 0$ and decreases if $f(y) < 0$, we see from the arrows of the figure, that $y(t) \to y_1$ if $y(0) < y_2$, and $y(t) \to y_3$ if $y(0) > y_2$, as $t \to \infty$. See the right part of the figure. With an intuitively understandable terminology (that is consistent with the formal definitions given below), we may say that the critical points y_1, y_3 are **stable** (or **attracting**), while the critical point y_2 is **unstable** (or **repelling**).

This discussion can be applied to any single autonomous ODE. Notice that a critical point p is stable if $f'(p) < 0$, and unstable if $f'(p) > 0$. It is left to the

Figure 13.7.1. *Graph of $f(y)$.*

reader to figure out what can happen if $f'(p) = 0$. Also find an example with one stable and two unstable critical points.

By Taylor's formula, $f(y) \approx f'(p)(y-p)$ if $f'(p) \neq 0$, it is seen that the time scale (of the attraction or repulsion) for a motion that starts near p is, at the beginning, is measured by $|f'(p)|^{-1}$. In the case of repulsion, the neglected terms of this Taylor expansion will play a bigger role, as time goes by.

Now we shall consider a general autonomous system.

Theorem 13.7.1. *Let $V \subset \mathbf{R}^s$ be a closed set with a piecewise smooth boundary. A normal pointing into V is then defined by a vector-valued, piecewise smooth function $n(y)$, $\quad y \in \partial V$.*

Assume that there exists a function $n_1(y)$ that satisfies a Lipschitz condition for $y \in \mathbf{R}^s$, such that

(a) $\|n_1(y)\| \leq K$ *for* $y \in \mathbf{R}^s$,

(b) $n(y)^T n_1(y) \geq c > 0$ *for* $y \in \partial V$.

Consider an autonomous system $y' = f(y)$, and assume that

$$n(y)^T f(y) \geq 0 \quad \forall y \in \partial V, \tag{13.7.1}$$

and that $y(a) \in V$. Then the motion stays in V for all $t > a$.

COMMENTS:

- V is, for example, allowed to be a polyhedron or an unbounded closed set.
- $n_1(y)$ is to be thought of as a smooth function defined in the whole of $\mathbf{R}^s$,. On ∂V it should be a smooth approximant to $n(y)$.

Proof. Sketch: Consider Fig. 13.7.3. The statement is almost trivial, if the inequality in (13.7.1) is strict. To begin with, we therefore consider a modified problem,

$$y' = f(y) + pn_1(y), \qquad p > 0,$$

Figure 13.7.2.

with the solution $y(t;p)$. Then

$$n(y)^T y' \geq n(y)^T p n_1(y) \geq pc > 0, \qquad y \in \partial V.$$

In other words: at every boundary point, the velocity vector for the modified problem points into *the interior of* V. Therefore, an orbit of the modified problem that starts in V can never escape out of V, i.e., $y(t;p) \in V$ for $t > a, p > 0$. By Theorem 13.8.2, $y(t;p) \to y(t)$, as $p \to 0$. Since V is closed, this proves the statement. □

COROLLARIES:

A. Comparison Theorem

Let $y(t)$ be the solution of a single non-autonomous equation,

$$y' = f(t,y), \qquad y(a) = c. \tag{13.7.2}$$

If a function $z(t)$ satisfies the inequalities, $z'(t) \leq f(t,z(t))$, $\forall t \geq a$, $z(a) \leq y(a)$, then $z(t) \leq y(t)$ $\forall$ $t \geq a$.

Figure 13.7.3.

Demonstration. You can either convince yourself by a glance at Fig. 13.7.3, or deduce it from the previous theorem, after rewriting (13.7.2) as an autonomous system, and define $V = \{(t,y) : y \geq z(t)\}$. □

There are variants of this result with reversed inequalities, which can easily be reduced to the case treated. Another variant: *if strict inequality holds in at*

least one of the two assumptions concerning $z(t)$, then strict inequality holds in the conclusion, i.e., $z(t) < y(t)$, $\forall t > a$.

B. Positivity Theorem.

Assume that for $i = 1, 2, \ldots, s$,

(a) $y_i(0) \geq 0$,

(b) $f_i(y) \geq 0$, *whenever* $y_i = 0$, *and* $y_j \geq 0$ *if* $j \neq i$.

Then $y_i(t) \geq 0$ for all $t > a$.

Hint to a proof: Choose $V = \{y : y_i \geq 0, \quad i = 1, 2, \ldots, s\}$. □

Another variant: If (a) is replaced by the condition $y_i(0) > 0$, and (b) is unchanged, then $y_i(t) > 0$, $\forall t > a$, (but $y_i(t)$ may tend to zero, as $t \to \infty$).

In many applications, the components of y correspond to physical quantities known to be non-negative in nature, e.g. mass densities or chemical concentrations. A well designed mathematical model should preserve this natural non-negativeness, but since modelling usually contains idealizations and approximations, it is not self-evident that the objects of a mathematical model possess all the important properties of the natural objects. The positivity theorem can sometimes be used to show that it is the case.

It is important to realize that a numerical method can violate such natural requirements, for example if the step size control is inadequate, see Example 13.2.4.

Another branch of the qualitative theory of ODEs, is concerned with the **stability of critical points**, not to be confused with the stability of numerical methods. Let p be a critical point of the non-autonomous system, $y' = f(t, y)$, i.e., $f(t, p) = 0$, $\forall t \geq c$.

Definition 13.7.2.

The critical point p is **stable**, *if for any given $\epsilon > 0$ there exists a $\delta > 0$, such that, for all $a \geq c$, if $\|y(a) - p\| < \delta$ then $\|y(t) - p\| < \epsilon$, $\forall t > a$. If a critical point is not stable, it is called* **unstable**. *The critical point p is* **asymptotically stable**, *if it is stable and $\lim_{t\to\infty} y(t) = p$.*

For a linear homogeneous system $y' = A(t)y$ it follows that the stability of the origin is the same as the boundedness of all solutions, as $t \to \infty$. If A is constant, this means that $\|e^{At}\| \leq C$, $\forall t \geq 0$.

Theorem 13.7.3.

Let A be a constant square matrix. The origin is a stable critical point of the system $y' = Ay$, if and only if the eigenvalues of A satisfy the following conditions:

(i) *The real parts are less than or equal to zero.*

(ii) *There are no defective eigenvalues on the imaginary axis.*

The stability is asymptotic if and only if all eigenvalues of A have strictly negative real parts.

Proof. Hint: Express e^{At} in terms of the Jordan canonical form of A. □

This theorem is not generally valid for linear systems with variable coefficients. There are examples (see Problem 12 b), where the equation $y' = A(t)y$ has unbounded solutions, in spite that $\Re\lambda(A(t)) \leq -1$ (say) for all t.

Another important fact is that *stability and boundedness are not equivalent for nonlinear problems.* We saw in Example 13.7.1 that a solution that started a little above the unstable critical point y_2 became bounded by y_3.

Theorem 13.7.4.

Consider the system $y' = Ay + g(t,y)$, where $\|g(t,y)\|/\|y\| \to 0$, uniformly in t. (The constant matrix A is the Jacobian at the origin, for all t.)
If the origin is asymptotically stable for the linear approximation $y' = Ay$, it is so also for the non-linear system.

If A has at least one eigenvalue with positive real part, then the origin is unstable for the non-linear system.

If $\max \Re\lambda(A) = 0$, then the stability question for the non-linear problem cannot be settled by the discussion of the linear approximation $y' = Ay$.

The first two statements will be proved by a logarithmic norm technique (see Sec. 13.1.4. The last statement is exemplified in Problem 10. If the critical point is located at p, one has only to replace y by $y - p$ in the formulation.

We shall, in *real two-dimensional examples*, occasionally use an established terminology for a critical point, based on the eigenvalues of the Jacobian. For the linear system $y' = Ay$ we have the following:

- If they are real and of the same sign, the point is a stable or unstable **node**.
- If they are real and of opposite sign, the point is a **saddle point**.
- If they are conjugate complex with a non-zero real part, the point (or rather an orbit in its neighborhood) is a stable or unstable **spiral point**.
- If they are pure imaginary, the point is a neutrally stable **center**.
- If A is singular, one or both eigenvalues are zero, the critical point is not unique. The orbits are rays or, if $A = 0$, just the critical points.

If you are not familiar with this terminology, see Problem 9.

By Theorem 13.7.4 the behavior in *the neighborhood of a critical point of a non-linear system* is approximately the same, except that a neutrally stable center can also become a stable or unstable spiral point in the nonlinear case. There is also a case named **elliptic sector** that has no counterpart in a linear problem, see Problem 5.

We omit a detailed discussion of the case of a singular A, where one has to consider, whether or not the Jacobian has the same rank at the critical point as in its neighborhood.

The above definitions of stability etc. are essentially due to Liapunov. In some texts our notions are named *uniform* stability etc., since one considers also a more

general case, where δ may depend on the initial time a, so that δ is not uniformly bounded away from zero.

Notice that, in the unstable linear autonomous case, the solution is bounded, if the initial value is restricted to the subspace spanned by the eigenvectors belonging to the eigenvalues with negative real parts. Some authors use the term *conditionally stable* for this case, and applies this notion also to the nonlinear case, where in general the set of exceptional initial values will be a nonlinear manifold. We shall not use this terminology.

Liapunov's name is also associated with a technique for investigating stability. For the sake of simplicity, we restrict the presentation to autonomous systems. A more general treatment is found in Lefschetz [1963]. We begin by a generalization of two notions earlier used for quadratic forms only.

Definition 13.7.5.

A real-valued function $V(y)$ is **positive definite** *in an open environment Ω of a critical point p of the system $y' = f(y)$, if*

(a) $V(y)$ and $V'(y)$ are continuous in Ω,

(b) $V(p) = 0$,

(c) $V(y) > 0$ for $y \in \Omega$, $y \neq p$.

$V(y)$ is **negative definite** *if $-V(y)$ is positive definite.*

Definition 13.7.6.

A positive definite function $V(y)$ is called a **Liapunov function**, *if $V'(y)f(y) \leq 0$ in Ω.*

The importance of $V'(y)f(y)$ is explained by the following equation:

$$dV(y)/dt = V'(y)dy/dt = V'(y)f(y), \text{ for } y = y(t).$$

Theorem 13.7.7. (Liapunov)

If there exists a Liapunov function $V(y)$ in some neighborhood Ω of a critical point p, then the critical point is stable.
If $V'(y)f(y)$ is negative definite in Ω, then the critical point is asymptotically stable.

Proof. There is a beautiful proof of this result, see e.g. Lasalle and Lefschetz, [1961,p.37]. □

Corollary 13.7.8.

If $V'(y)f(y) = (y-p)^T C(y-p) + o(\|y-p\|^2$, where C is a negative definite matrix, then the critical point p is asymptotically stable. If asymptotic stability can be proved for the linear approximation, by means of a quadratic form, this quadratic form is a Liapunov function also for $y' = f(y)$.

It is often by no means trivial to construct a Liapunov function for a given problem. In physical problems energy considerations may give a good hint.

Example 13.7.2.

The equation for a damped or undamped **pendulum** reads (after a scaling of the time variable): $\phi'' + a\phi' + \sin\phi = 0$, $(a \geq 0)$. We can write this in the form $y' = f(y)$, if we set $y = [y_1, y_2]^T = [\phi,\ \phi']^T$, $f(y) = [y_2,\ -ay_2 - \sin y_1]^T$. We shall give the main features of a study of the stability of the critical point at $y = 0$. The details are left for Problem 10d.

The sum of the potential and the kinetic energy is $E(y) = 1 - \cos y_1 + \frac{1}{2}y_2^2$. We choose $V(y) = E(y) + \eta y_1 y_2$, where η is a small positive quantity. Then $V'(y)f(y) = -\eta y_1 \sin y_1 - (a - \eta)y_2^2 - a\eta y_1 y_2$.

In the *damped case*, $a > 0$, we choose $\eta \ll a$. $V'(y)f(y)$ is a negative definite function in a sufficiently small neighborhood Ω, since we obtain a negative definite quadratic form, if $y_1 \sin y_1$ is replaced by y_1^2. Then, by the corollary above, the origin is *asymptotically stable*. The orbit in the y-plane spirals in towards the origin. Phase plane plots for a damped pendulum are shown in the right part of Fig. 13.7.2.

If we had chosen $\eta = 0$, the quadratic form, which approximates $V'(y)f(y)$, would have become semi-definite only. That does not prove asymptotic stability.

In the *undamped case*, $a = 0$, we take $\eta = 0$ and find that $V'(y)f(y) = 0$, hence the origin is *stable*. Note that this means that $dV(y(t))/dt = 0$, hence $V(y)$ *is constant during the motion*. (This is not unexpected, since in this case $V(y)$ equals the total energy.) If the starting point is sufficiently close to the origin, the motion will be periodic along a closed level curve for $V(y)$. See Fig. 13.7.1. If we let the starting point approach the origin, the period of the the solution tends to the period for the linear approximation, $y' = Ay$, where A is the Jacobian at the critical point. (In this example the period of the linear system is 2π.)

At a simulation the period can be computed by means of the times for the intersections between the orbit and some ray. (For larger systems there will be a plane or a hyperplane instead of a ray.)

In numerical analysis a primary interest is to estimate the difference between a perturbed solution $z(t)$ and an unperturbed solution $y(t)$ of a system of ODEs. In the particular case where we consider the effect of a perturbation at the initial point only, the variable transformation $u(t) = y(t) - z(t)$ makes the origin a critical point for a differential system for the function $u(t)$. (Note however that this system usually becomes non-autonomous even if the original system for y is autonomous.) We usually discuss this in terms of a norm $\|u(t)\|$, and in the next section techniques will be developed, based on the notion of logarithmic norms, to make this type of analysis efficient. *A norm is a particular kind of Liapunov function*, except that differentiability is not required, e.g., the max-norm and the l_1-norm are not differentiable everywhere.

A positive definite quadratic form is a frequently used type of Liapunov function. Since such a form is the square of an inner-product norm, its usage as a Liapunov function is equivalent to the use of an appropriate norm. For the discussion of asymptotic stability and for most questions encountered at the study of

numerical methods the restriction to norms does not seem hindering, but norms are sometimes too crude for the delicate case of non-asymptotic stability.

In the problems with periodic solutions that we have encountered so far, there has been a whole family of periodic orbits. There is another type of periodic solutions in nonlinear problems, called **limit cycles**.

Example 13.7.3. The complex differential equation

$$z' = iz - 0.5z(|z|-1)(|z|-2)(|z|-3),$$

which can be written as a system of two real differential equations, provides a simple example of limit cycles. Set $z = re^{i\phi}$, and separate real and imaginary parts: $r' = -0.5r(r-1)(r-2)(r-3), \quad \phi' = 1$. The first of these equations can be discussed like Example 13.7.1. There are unstable critical points at 0 and 2, while 1 and 3 are stable critical points. The second equation shows that a uniform rotation is superimposed on the development of $r(t)$.

Figure 13.7.4. *(a) Limit cycle for Example 13.7.3. (b) Equation run backward in time.*

So, for $0 < |z(0)| < 1$, r increases towards 1, while ϕ increases monotonically. The orbit is a spiral that approaches the unit circle from the inside. See the left part of Fig. 13.7.1.

Similarly, for $1 < |z(0)| < 2$ the orbits approach the unit circle from the outside. The unit circle is therefore said to be a stable limit cycle. The circle $|z| = 3$ is also a stable limit cycle, for initial values $|z(0)| > 2$. The point $r = 2$ is an unstable critical point for the real differential equation for r. Therefore the circle $|z| = 2$ becomes an unstable limit cycle for the complex differential equation.

If the problem is run backwards in time, the situation is opposite; the circle $|z| = 2$ becomes a stable limit cycle, while the other two become unstable. See the right part of Fig. 13.7.1.

Figure 13.7.5. *The limit cycle of the Brusselator problem.*

The previous simple example should not make you believe that limit cycles are always circles. Fig. 13.7.5 shows the limit cycle of a famous problem in chemistry, named the Brusselator problem, see Computer Exercise 6. Limit cycles play a central role in the theory of autonomous ODEs in $\mathbf{R^2}$, according to the following classical theorem from around 1900. We quote the formulation of Hairer et al., and refer to Lefschetz [1963] or Coddington and Levinson [2] for a proof.

Theorem 13.7.9. (The Poincaré-Bendixson Theorem)
Each bounded solution of an autonomous system $y' = f(y)$ *in* $\mathbf{R^2}$ *must*

(i) *tend to a critical point for an infinity of points* $t_i \to \infty$*; or*

(ii) *be periodic; or*

(iii) *tend to a limit cycle.*

The complicated formulation of the first alternative is related to the possibility of a critical point on the limit cycle, see Hairer et al. [7, 1993, p.123].

There may be limit cycles also in larger systems of ODEs, but there is no general result like the Poincaré-Bendixson theorem. For instance, a motion in three dimensions can become chaotic, and its limiting set can have a fractal structure, a so-called **strange attractor**. A famous example of this is due to E. N.Lorenz, see Fig 13.7.5b and Computer exercise 7. A very illuminating discussion and a numerical study is presented by Hairer et al. [7, 1993, p.117 ff.].

13.7.2 More Applications of The Logarithmic Norm

Now we shall use the concept of logarithmic norm for proving the existence and uniqueness of a solution of a system of non-linear algebraic equations that may not

be easily brought to a form required by the theorems of Sec. 13.1. We first note that such results are needed, because *a nonlinear system can have more than one solution, even if the Jacobian is non-singular everywhere.* A simple example is the system,

$$e^{y_1}\cos y_2 = b_1, \quad e^{y_1}\sin y_2 = b_2, \qquad (b_1, b_2) \neq 0.$$

The Jacobian determinant equals $\exp(2y_1) \neq 0$. The general solution is $(y_1, y_2) = (\ln r, \phi)$, where (r, ϕ) are the polar coordinates for the point with Cartesian coordinates (b_1, b_2). y_2 is determined only modulo 2π.

The result, Theorem 13.7.10, is essentially due to Desoer and Haneda [4, 1972]. Such systems may occur at every time step of the treatment of a stiff system of ODEs by an implicit method. The systems can be written in the form $F(y) = 0$, where

$$F(y) = \beta h f(y) - y + \gamma, \tag{13.7.3}$$

where β, γ and the time step h are constant during a time step. In practice a damped and modified Newton method (see Sec. 12.1.6) usually works well. Since global uniqueness is not to be expected for such systems, it can, however, happen that an unwanted solution of the system is computed, unless the time step and the error of the initial guess are small enough. The following theorem provides sufficient conditions for existence and uniqueness.

Theorem 13.7.10.

Let y_0 be a given point in $\mathbf{R}^s$, and set

$$D_r = \{y \in \mathbf{R}^s \quad : \quad \|y - y_0\| \leq r\}.$$

Consider the system $F(y) = 0$, where $F : D_r \to \mathbf{R}^s$ is in C^1. Assume that 1. $\|F(y_0)\| < r\delta$, $(\delta > 0)$.

2. $\mu(F'(y)) \leq -\delta$ *for* $y \in D_r$.

Then, the system $F(y) = 0$ has exactly one solution y^* in D_r.

Proof. We first prove *uniqueness.* Suppose that y_1, y_2 are two different roots in D_r to the system $F(y) = 0$. By (13.8.12), we can write, $F(y_1) - F(y_2) = J(y_1 - y_2)$, where the matrix J is a neighborhood average of F'. By (13.1.23) and Assumption 2, $\mu(J) \leq -\delta$. Then, by Theorem 13.1.11, statement C,

$$\|J(y_1 - y_2)\| \geq \delta \|y_1 - y_2\|.$$

Since $J(y_1 - y_2) = F(y_1) - F(y_2) = 0$ this contradicts the assumption that $y_1 \neq y_2$. Hence there is *at most one solution* in D_r to the system.

In order to prove the *existence,* we *embed* our system in a one-parameter family of systems,

$$F(y(t)) = (1 - t)F(y_0), \qquad 0 \leq t \leq 1. \tag{13.7.4}$$

For $t = 0$, the system has the solution $y(0) = y_0$. For $t = 1$, we have our original system, $F(y) = 0$. We shall show that this system has a solution $y(t) \in D_r$, that is

a continuous function of t. We differentiate (13.7.4),

$$F'(y)\frac{dy}{dt} = -F(y_0), \qquad y(0) = y_0.$$

$$\frac{dy}{dt} = -F'(y)^{-1}F(y_0), \qquad y(0) = y_0. \tag{13.7.5}$$

A function $y(t)$ that satisfies this initial value problem, satisfies also (13.7.4). The existence theorem of Peano, mentioned in Comment 1 after Theorem 13.1.1, guarantees a differentiable solution $y(t)$ as long as $F'(y)^{-1}$ exists and is a continuous function of y. As long as $y \in D_r$,

$$\|F'(y)^{-1}\| \le |\mu(F'(y)|^{-1} \le \delta^{-1}, \tag{13.7.6}$$

by Theorem 13.1.11, statement D, and Assumption 2. The continuity of $F'(y)^{-1}$ then follows from the relation,

$$A^{-1} - B^{-1} = B^{-1}(B - A)A^{-1}, \quad \text{with} \quad A = F'(y_1),\ B = F'(y_2),$$

i.e., $F'(y_1)^{-1} - F'(y_2)^{-1} = F'(y_2)^{-1}(F'(y_2) - F'(y_1))F'(y_1)^{-1}$.

Now it only remains to show that $y(t)$ stays in D_r for $0 \le t \le 1$. In fact,

$$\|y(t) - y(0)\| = \|\int_0^t y'(s)ds\| \le \int_0^t \|F'(y(s))^{-1}F(y_0)\|ds < \int_0^t \delta^{-1}\, r\delta ds = rt.$$

Assumption 1 and (13.7.6) were used in the next to last step. Hence $y(t) \in D_r$ for $t \le 1$. For $t = 1$ we obtain, by (13.7.4), the result that *the system* $F(y) = 0$ *is satisfied by* $y = y(1)$. □

Note that for the function f in (13.7.3), Assumption 2 leads to a very liberal condition:

$$\mu(f'(y)) < \frac{(1-\delta)}{\beta h} \quad \forall y \in D_r.$$

We shall now collect some formulas that are useful, when one works with other norms than the max-norm and the l_1-norm, for which expressions were given above. Let T be a non-singular matrix, and let $\|\cdot\|$ be any vector norm. Then it is easily seen that

$$\|u\|_T = \|T^{-1}u\|. \tag{13.7.7}$$

satisfies the three conditions for a vector norm, stated at the beginning of Sec. 6.2.5.

Next, set $u = Tv$. Then

$$\frac{\|Bu\|_T}{\|u\|_T} = \frac{\|T^{-1}BTv\|}{\|v\|}$$

Since T is non-singular, $\max_u$ means the same as $\max_v$. It follows that the subordinate matrix norm is

$$\|B\|_T = \|T^{-1}BT\|.$$

For $B = I + \epsilon A$ we then obtain, $\|I + \epsilon A\|_T = \|T^{-1}(I + \epsilon A)T\| = \|I + \epsilon T^{-1}AT\|$, and it follows from the definition of the subordinate logarithmic norm that

$$\mu_T(A) = \mu(T^{-1}AT). \tag{13.7.8}$$

Let $D = \operatorname{diag}(d_i)$, $\max_i |d_i| = |d_{i'}|$. Then

$$\|Du\|_p^p = \sum_i |d_i u_i|^p \le \max_i |d_i|^p \|u\|_p^p.$$

Equality is obtained when $u = e_{i'}$, (one of the basis vectors). Hence $\|D\|_p = |d_{i'}|$. Now substitute $I + \epsilon D$ for D. Then $\|I + \epsilon D\|_p = |1 + \epsilon d_{i^*}|$, where i^* is the same for all sufficiently small ϵ (though not necessarily equal to i'). Then, by the definition of logarithmic norm, $\mu_p(D) = \Re d_{i^*}$, which must be equal to $\max \Re d_i$. It follows that the formulas

$$\|D\|_p = \max |d_i|, \qquad \mu_p(D) = \max \Re d_i, \tag{13.7.9}$$

are valid for any l_p-norm. *The same derivation also holds for weighted l_p-norms.*

Theorem 13.7.11.

Let $(u, v) = u^H H v$ be an inner product in $\mathbf{C}^s$, and set $\|u\| = \sqrt{(u,u)}$. Then the subordinate logarithmic norm equals

$$\mu(A) = \max_{(u,u)=1} \Re(u, Au) = \max\{\kappa : \det(A^H H + HA - 2\kappa H) = 0\}, \tag{13.7.10}$$

$$\mu(A) < 0 \;\leftrightarrow\; A^H H + HA \text{ negative definite} \tag{13.7.11}$$

For the l_2-norm,

$$\mu_2(A) = \max_i d_i, \tag{13.7.12}$$

where d_i denotes an eigenvalue of the hermitean part $B = \frac{1}{2}(A + A^H)$ of A, (Note that (13.7.12) contains $\max d_i$, not $\max |d_i|$.)

Proof. In order to derive (13.7.10), consider the equation

$$\|I + \epsilon A\|^2 - 1 = \max_{(u,u)=1} \big((u + \epsilon Au, u + \epsilon Au) - 1\big).$$

The second member equals the maximum of

$$(u,u) + \epsilon(Au, u) + \epsilon(u, Au) + \epsilon^2(Au, Au) - 1 = \epsilon\big(2\Re(Au, u) + O(\epsilon)\big).$$

The first member can be written $(\|I + \epsilon A\| + 1) \cdot (\|I + \epsilon A\| - 1)$. Hence

$$(2 + O(\epsilon))\frac{\|I + \epsilon A\| - 1}{\epsilon} = 2 \max_{(u,u)=1} \Re(Au, u) + O(\epsilon),$$

The first part of (13.7.10) follows, as $\epsilon \to +0$. The derivations of (13.7.11) and the second part of (13.7.10) are left for Problem 21.

Finally, the l_2-norm is an inner product norm, where $(u,v) = u^H v$. Hence

$$\mu_2(A) = \max_{\|u\|_2=1} \Re u^H A u = \max_{\|u\|_2=1} u^H B u,$$

where $B = \frac{1}{2}(A + A^H)$ is the hermitean part of A. Let T be a unitary matrix that diagonalizes B, i.e., $T^H B T = \operatorname{diag}(d_i)$, and set $u = Tv$. Note that $\|v\|_2 = \|u\|_2 = 1$. Then

$$u^H B u = v^H T^H B T v = v^H \operatorname{diag}(d_i) v = \sum_i d_i |v_i|^2 \le \max_i d_i.$$

The same type of argument as was used in the derivation of (13.7.9) then shows that

$$\mu_2(A) = \max_{\|u\|_2=1} u^H B u = \max_i d_i.$$

(The maximum is obtained for $u = Te_i$, where e_i is one of the basis vectors.) □

A simple and useful upper bound of $\mu_2(A)$ is obtained by the combination of (13.7.12) with Theorem 13.1.11, statement B:

$$\mu_2(A) = \max \Re\lambda(B) \le \mu_\infty(B), \tag{13.7.13}$$

where B is the hermitean part of A.

The sharpness of the bounds obtained by Theorems 13.1.8 and 13.1.10 depends on the choice of norm. Set $\alpha(J) = \max\{\Re\lambda : \lambda \in \lambda(J)\}$. We know that $\mu(J) \ge \alpha(J)$. *A logarithmic norm is said to be* **efficient** *for the matrix J, if equality holds. For a Hermitean matrix, $\mu_2(\cdot)$ is efficient*, by (13.7.12), but is there an efficient norm for a general matrix? The analogous question for operator norms was answered by Theorem 10.2.9. The proof of the following result is omitted, since it is very similar to the proof of that theorem.

Theorem 13.7.12.

Given a matrix $A \in \mathbf{R}^{n\times n}$, and set $\alpha(A) = \max\{\Re\lambda : \lambda \in \lambda(A)\}$. Denote by $\|\cdot\|$ any l_p-norm (or weighted l_p-norm), $1 \le p \le \infty$. Set $\|x\|_T = \|T^{-1}x\|$, and recall that, by (13.7.8), $\mu_T(A) = \mu(T^{-1}AT)$. Then the following holds:

(a) *If A has no defective eigenvalues with $\Re\lambda = \alpha(A)$, then there exists a matrix T such that $\mu_T(A) = \alpha(A)$.*

(b) *If A has a defective eigenvalue with $\Re\lambda = \alpha(A)$, then for every $\epsilon > 0$ there exists a matrix $T(\epsilon)$, such that $\mu_{T(\epsilon)}(A) \le \alpha(A) + \epsilon$.*
As $\epsilon \to 0$, the condition number $\kappa(T(\epsilon))$ tends to ∞ like ϵ^{1-m^*}, where m^* is the largest order of a Jordan block belonging to an eigenvalue λ with $\Re\lambda = \alpha(A)$.

Corollary 13.7.13.

If $\max \Re\lambda < \alpha^*$, then there exists an inner-product norm, such that the subordinate logarithmic norm is $\mu(A) < \alpha^*$.

For some classes of matrices, an efficient (or almost efficient) norm can be found more easily than by the construction used in the proof of Theorem 10.2.9. This may have other advantages as well, e.g. a better conditioned T. Consider a **weighted max-norm** $\|x\|_w = \max_i |x|_i/w_i = \|T^{-1}x\|_\infty$, where $T = \operatorname{diag}(w_i)$. Then

$$\mu_w(A) = \mu_\infty(T^{-1}AT) = \max_i \Re a_{ii} + \sum_{j,\ j\neq i} |a_{ij}|w_j/w_i. \tag{13.7.14}$$

Note that $\kappa(T) = \max w_i / \min w_i$.

Set $\tilde{A} = [\tilde{a}_{ij}]$, where $\tilde{a}_{ii} = \Re a_{ii}$, $\tilde{a}_{ij} = |a_{ij}|$ for $i \neq j$. Note that $\mu(A) = \mu(\tilde{A})$, when $\mu(\cdot)$ is subordinate to a weighted max-norm or a weighted l_1-norm. Also note that the inequality $\mu_w(A) \leq \beta$ is equivalent to the inequalities $\tilde{A}w \leq \beta w,\ w > 0$.

If A is irreducible, a modified form, see Problem 14, of the *Perron-Frobenius Theorem* (Theorem 10.2.12) tells that there exists a positive eigenvector w, such that $\tilde{A}w = \alpha(\tilde{A})w$, hence *the logarithmic norm $\mu_w(\cdot)$ is efficient for the matrix $\tilde{A}$.* It is in general not efficient for A itself, since $\alpha(A)$ may be less than $\alpha(\tilde{A})$, but still it can be useful also for A.

The latter result can be extended to some reducible matrices. e.g. to *any upper triangular $n \times n$ matrix A, such that $\Re a_{ii} < \Re a_{nn} = \alpha(A)$ for all $i < n$.* Then a positive vector w such that $\tilde{A}w \leq \alpha(A)w$ can be found by solving the inequalities $\Re(a_{nn} - a_{ii})w_i \geq \sum_{j,\ j>i} |a_{ij}|w_j$, for $i = n-1,\ n-2, \ldots, 1$. As in the analogous case discussed in Sec. 10.2.4 one may obtain a smaller value of $\kappa(T)$ by choosing w_i larger than necessary, for some i. (For example: the usual max-norm is efficient, if A is very strongly diagonally dominant and$\Re a_{ii} < \Re a_{nn}$.) There are important extensions of this to block matrices, see Sec. 13.3.

In the application of these results to yield improved bounds for the solution of ODEs there is one more complication: if the Jacobian varies with time then the matrix T is also likely to do so. It is sufficient to study a pseudo-linear system, since a general non-linear system can be reduced to this case, as in Theorem 13.1.10.

Theorem 13.7.14.

Consider the pseudo-linear system

$$\frac{du}{dt} = J(t,u)u + r(t,u).$$

Let $T(t)$ be a smooth non-singular matrix-valued function. In addition to a given norm $\|\cdot\|$, we consider a time-dependent vector norm $\|y\|_T = \|T^{-1}y\|$. Assume that, for every $t \in [a,b]$, there exists a real-valued function $\mu^*(t)$, and a convex domain $D_t \subseteq \mathbf{R}^s$, such that

$$\mu_T(J(t,w)) + \mu(-T^{-1}T'(t)) + \mu(T^{-1}T'(t)) \leq \mu^*(t),\ \kappa(T(t))\|r(t,w)\| \leq \epsilon(t),$$
$$\forall w \in D_t.$$

Then $\|u(t)\| \leq \psi(t)$, where $\psi(t)$ is a solution of the scalar differential equation,

$$\frac{d\psi}{dt} = \mu^*(t)\psi + \epsilon(t), \qquad \psi(a) \geq \|u(a)\|,$$

as long as the bounds obtained from this guarantee that $u(t) \in D_t$. *(See e.g.* (13.1.28)*, if* μ^* *and* ϵ *are constant.)*

Proof. Set $u = Tz$. Then $Tz' + T'z = JTz + r$, i.e.,

$$z' = (T^{-1}JT - T^{-1}T')z + T^{-1}r.$$

Hence, $\|z(t)\| \le \zeta(t)$, where

$$\zeta' = (\mu_T(J) + \mu(-T^{-1}T'))\zeta + \|T^{-1}r\|, \quad \zeta(a) \ge \|z(a)\|.$$

Now set $\eta = \|T\|\zeta$. Note that $\|u\| = \|Tz\| \le \|T\|\|z\| \le \|T\|\zeta = \eta$. Also note that $\|T\|' \le \mu(T^{-1}T')\|T\|$, since $dT/dt = T(T^{-1}T')$. Then

$$\eta' = \|T\|'\zeta + \|T\|\zeta' \le \mu(T^{-1}T')\|T\|\zeta + (\mu_T(J) + \mu(-T^{-1}T'))\|T\|\zeta + \kappa(T)\|r\|,$$

i.e.,

$$\eta' \le (\mu(T^{-1}T') + \mu_T(J) + \mu(-T^{-1}T'))\eta + \epsilon(t) \le \mu^*(t)\eta + \epsilon(t).$$

Hence $\|u(t)\| \le \eta(t) \le \psi(t)$, where $\psi(t)$ is defined above. The argument is valid as long as $\|u(t)\| \in D_t$. □

The application of this theorem is particularly simple, when $T(t)$ is a diagonal matrix. According to the remarks above, the sharpness of bounds obtained by an appropriately chosen diagonal matrix is related to the size of $\alpha(\tilde{J}) - \alpha(J)$.

Notice the similarities and the differences of this theorem and Theorem 13.2.1 (where, in a way, $S(t)$ corresponds to $T(t)$). One difference is the presence of the term $\mu(T^{-1}T')$ in the condition for μ^*. This is due to the fact that the norm $\|\cdot\|_T$ is here only an internally used aid for the derivation of a sharp bound that is to be expressed in terms of the original (external) norm $\|\cdot\|$. In Theorem 13.2.1, however, the matrix S performs a transformation to a norm that is used also in the result. The other characteristic feature of Theorem 13.2.1, namely the transformation of the independent variable ("age" instead of "time"), can be used as a preprocessing also to an application of Theorem 13.7.14 or Theorem 13.1.10, whenever it is appropriate.

The logarithmic norm can also be used to derive inequalities with reversed sign to the inequalities given above. The derivations are analogous. In some cases simple substitutions are enough. The details are left for Problem 10. There are similar modifications of other properties of $\mu(A)$.

Theorem 13.7.15.

A. *If* $u' = Ju + r$ *then* $\|u\|' \ge -\mu(-J)\|u\| - \|r\|$ *for* $t \ge 0$.
If $-\mu(-J) \ge \alpha$, *and* $\|r\| \le \epsilon$, *then* $\|u(t)\| \ge \psi(t)$, *where* $\psi' = \alpha\psi - \epsilon$, $\psi(0) \le \|u(0)\|$. *Moreover,* $\|e^{Jt}\| \ge e^{-\mu(-J)t}$, $\forall t \ge 0$.

B. *For any choice of norm,* $-\mu(-A) \le \min \Re\lambda(A)$, *and, for any* $\epsilon > 0$, *there is a norm such that* $-\mu(-A) \ge \min \Re\lambda(A) - \epsilon$.

C. $\mu(A) + \mu(-A) \geq \max \Re\lambda(A) - \min \Re\lambda(A)$,
$\mu_2(A) + \mu_2(-A) = \max \Re\lambda(B) - \min \Re\lambda(B)$, *where* B *is the hermitean part of* A.

We shall now prove Theorem 13.7.4 of Sec. 13.7.1 as an application of the logarithmic norm technique.

Proof. Proof of Theorem 13.7.4: We consider nonlinear systems of the form $y' = Ay + g(t, y)$, where A is a constant matrix, and $\|g(t, y)\|/\|y\| \to 0$, uniformly in t, as $y \to 0$.

In *the first part* it is assumed that all eigenvalues of A have strictly negative real parts, and we shall prove that the origin is asymptotically stable for the nonlinear system. By Theorem 13.7.12 we can then, for some $\eta > 0$, find a norm such that $\mu(A) < -\eta$. We then choose a positive ϵ small enough that $\|g(t, y)\| \leq \eta\|y\|$ when $\|y\| \leq \epsilon$. Finally we choose $\|y(0)\| \leq \epsilon$. Then

$$\|y(t)\|' \leq \mu(A)\|y(t)\| + \|g(t, y)\| \leq -\tfrac{1}{2}\eta\|y(t)\|,$$

hence $\|y(t)\| \leq \epsilon e^{-\eta t/2}$, which proves asymptotic stability.

Comment: Since this norm can be an inner-product norm, this proof also shows that a quadratic Liapunov function that proves asymptotic stability for the linear approximation, proves asymptotic stability also for the nonlinear equation. This was mentioned above as a corollary of Theorem 13.7.7, which is unproven in this text.

In *the second part*, it is assumed that A has at least one eigenvalue with a positive real part, and we shall prove that the origin is unstable for the nonlinear system. We can then make a coordinate transformation $y = Tu$, $g = Th$, that brings the operator A to block diagonal form, $T^{-1}AT = \text{blockdiag}(J_1, J_2)$, where (say)

$$\Re\lambda(J_1) \geq \alpha_1 > 0, \quad \Re\lambda(J_2) \leq \alpha_2 < \alpha_1.$$

The transformed system reads, in partitioned form,

$$u_1' = J_1u_1 + h_1(t, u), \quad u_2' = J_2u_2 + h_2(t, u).$$

Our proof is indirect. Suppose that the origin is stable, i.e., for any positive ϵ, we can find a $\delta > 0$ such that $\|u_i(0)\| \leq \delta$, $i = 1, 2$, implies that $\|u_i(t)\| \leq \epsilon$, $\forall t > 0$. For any $\theta > 0$ we can, by Theorems 12.2.15B and 12.2.13, choose norms so that

$$-\mu(-J_1) \geq \alpha_1 - \theta, \quad \mu(J_2) \leq \alpha_2 + \theta.$$

Next we choose two numbers θ and η, both less than $\frac{1}{8}\min(\alpha_1 - \alpha_2, \alpha_1)$. For later use, note that

$$\alpha_1 - \theta - 2\eta \geq \alpha_2 + \theta + 2\eta, \quad \alpha_1 - \theta - 2\eta \geq 5\alpha_1/8.$$

Finally ϵ should be small enough that $\|h_i(t, u)\| \leq \eta(\|u_1\| + \|u_2\|)$, $i = 1, 2$. This is possible due to the assumption originally made for $\|g(t, y)\|$. Then, by

Theorem 13.7.15 (statement A) and formula (13.8.9),

$$\|u_1\|' \geq (\alpha_1 - \theta)\|u_1\| - \eta(\|u_1\| + \|u_2\|),$$
$$\|u_2\|' \leq (\alpha_2 + \theta)\|u_2\| + \eta(\|u_1\| + \|u_2\|).$$

It follows that

$$\begin{aligned}\|u_1\|' - \|u_2\|' &\geq (\alpha_1 - \theta - 2\eta)\|u_1\| - (\alpha_2 + \theta + 2\eta)\|u_2\| \\ &\geq (\alpha_1 - \theta - 2\eta)(\|u_1\| - \|u_2\|).\end{aligned}$$

We choose $\|u_1(0)\| = \delta \ll \epsilon$, $u_2(0) = 0$. Then $\|u_1(t)\| - \|u_2(t)\| \geq 0$ for $t \geq 0$. Hence, $(\|u_1\| - \|u_2\|)' \geq \frac{5}{8}\alpha_1(\|u_1\| - \|u_2\|)$. We obtain,

$$\|u_1(t)\| - \|u_2(t)\| \geq \delta e^{5\alpha_1 t/8} \to \infty \quad t \to \infty.$$

This contradicts our hypothesis that the origin is stable. □

Review Questions

1. (a) Determine $y_0(t), y_1(t)$, so that, under conditions to be stated, the function $\tilde{y}(t) = y_0(t) + \epsilon y_1(t)$ is at an $O(\epsilon^2)$-distance from a particular solution to the ODE system $\epsilon y' = F(y)$, $0 < \epsilon \ll 1$ that is attractive for other solutions that start in its neighborhood.

(b) For a single ODE of the form $\epsilon y' = F(y)$, $0 < \epsilon \ll 1$, show that, under appropriate conditions, the solution $y(t)$ is approximately an $O(\epsilon)$-delay of a solution of the reduced problem, i.e., the algebraic equation obtained for $\epsilon = 0$. Give an example, where the conditions are no longer valid in the neighborhood of some points, and tell what happens.

2. Derive the non-homogeneous *linear* variational equation associated with a differential system $dy/dt = f(t, y; p)$, $y(a) = c(p)$, where p is a vector of parameters.

What is the *nonlinear* variational equation? Exemplify how it is used.

3. Consider a differential system $y' = f(y)$. Formulate a general theorem, that guarantees that a motion that starts in a domain V will remain there forever. Exemplify how the theorem is to be applied on a domain with singular points on the boundary, e.g., a closed circular cone. Formulate and exemplify the use of its corollaries, in the text called the Comparison theorem and the Positivity Theorem.

4. (a) Consider the linear autonomous system $y' = Ay$. Formulate in terms of the spectrum of A necessary and sufficient conditions for the stability and asymptotic stability of the origin.

(b) Give the main features of an example that shows that a nonautonomous linear system can have unbounded solutions, even if, for every t, the eigenvalues of $A(t)$ are less than some negative constant.

5. (a) Define the basic notions of the Liapunov stability theory: stability, asymptotic stability, Liapunov function. Formulate, in terms of Liapunov functions, the basic theorems about the stability and asymptotic stability of a critical point.

(b) Exemplify the distinction between stability and boundedness for a nonlinear ODE.

(c) Show, with a reference to a theorem in the text (i.e. the corollary of Theorem 13.7.12) that a critical point is asymptotically stable for a non-linear system, if it is asymptotically stable for the linear approximation. Does the statement remain true, if you remove the words "asymptotically"? Give a proof or a counterexample.

6. Give an example of a problem with a limit cycle. Draw a simple phase plane sketch. What is the difference between a limit cycle and a periodic solution of the type exemplified by the undamped pendulum and the predator-prey problem? Quote the Poincaré-Bendixson theorem.

Problems

1. The following set of autocatalytic chemical reactions was studied by H. H. Robertson [1966]:

$$\begin{aligned} A + A &\longrightarrow C + A, && \text{rate coeff.} = 3 \cdot 10^7; \\ A + C &\longrightarrow B + C, && \text{rate coeff.} = 1 \cdot 10^4; \\ B &\longrightarrow A\ , && \text{rate coeff.} = 0.04. \end{aligned}$$

According to the law of mass action, see, e.g., Lin-Segel [1974], p.302, this leads to the following ODE system, where y_1, y_2, y_3 denote the concentrations of the species A, B, C, respectively.

$$\begin{aligned} y_1' &= -10^4 y_1 y_3 - 3 \cdot 10^7 y_1^2 + 0.04 y_2, && y_1(0) = 0, \\ y_2' &= 10^4 y_1 y_3 - 0.04 y_2, && y_2(0) = 1, \\ y_3' &= 3 \cdot 10^7 y_1^2, && y_3(0) = 0. \end{aligned}$$

(a) Show that all variables are positive (or at least non-negative) for $t > 0$, as they should, because of they are concentrations.

(b) Show that $y_1(t) + y_2(t) + y_3(t) = 1$, by adding the three equations etc.

Comment: Such **invariants** or *first integrals* can often be derived either from some physical principle or by some mathematical manipulation (see the next problem). They can often be used for *simplification or stabilization* of computations.

2. (a) Usually the ODEs derived from the law of mass action, have the following form, $y_i' = P_i(y) - y_i Q_i(y)$, $i = 1, 2, \ldots, s$, where $P_i(y)$, $Q_i(y)$ are functions

of the vector y, usually polynomials, which are non-negative for every vector with non-negative components. Show that $y(t)$ is non-negative for all $t > 0$, if the initial values are so.

(b) The ODEs derived from the law of mass action, can often also be written in the form, $y' = Ar(y)$, where A is a constant rectangular matrix, and $y \in \mathbf{R}^s$, $r(y) \in \mathbf{R}^m$. Describe how linear invariants of the ODE system may be found, if there are any, by the manipulation of the matrix A.

(c) Rewrite the equations of Problem 1 in these forms.

3. The different types of neighborhood to a critical point in $\mathbf{R^2}$ may be illustrated on single complex equations of the form $z' = \lambda z$ with $\lambda = -1,\ 0,\ 1,\ -1+i,\ i,\ 1+i,$ *and* real systems of the form $y' = Ay$, with matrices of the form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \quad \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}$$

where one should distinguish between λ positive, negative or zero. (There are in all something like 16 types). Draw some sketches with or without a computer, and/or consult a modern text on ODEs, or Strang [1986].
For the system $y' = Ay$ in $\mathbf{R^2}$, find out how the trace and the determinant of A provide almost complete information about the type of a critical point.

4. (a) Determine the stability type (or instability) of the origin for $y' = ay^3$ and for the complex equation $z' = iz + a|z|^2 z,\ a \in \mathbf{R}$. Write the latter system as a real system in both Cartesian and polar coordinates. Use the latter for settling the stability question, and sketch the solution, for a positive and a negative value of a.

(b) Determine the stability or instability of the origin for the system $y_1' = y_2 - y_1^3,\quad y_2' = 0$. Is it true that this shows that the origin can be stable for a nonlinear system, even though it is unstable for the linear approximation?

(c) Consider the equation $y' = -y^2 + r(t)$, $y(0) = \epsilon$, $(\epsilon > 0)$. Find upper and lower bounds for $y(t)$, valid for all $t > 0$, if $0 \le r(t) < \epsilon$? Can you get a lower bound, if you only know that $|r(t)| < \epsilon$? (This is related to trouble that has been encountered at the numerical solution of stiff problems, when the system has similar properties for small values of $\|y\|$ as this scalar example.)

(d) Work out the details of Example 13.7.2 (the pendulum). Study also the (in)stability of the other critical points.

5. (Gomory, Lefschetz). Consider the system $y_1' = y_1$, $y_2' = -y_2$. Show that the orbits are hyperbolas, which are, in polar coordinates, described by the equation $r^2 \sin 2\phi = \text{const.}]$ By the transformation by reciprocal radii, we obtain a curve family, $r^2 = \text{const.}] \sin 2\phi$. Sketch these curves. Show that they satisfy the system $y_1' = y_1^3 - 3y_1y_2^2$, $y_2' = 3y_1^2y_2 - y_2^3$. This exemplifies the *elliptic sector* type of critical point mentioned in Sec. 13.7.1.

6. (a) Set

$$A_{2m} = \begin{pmatrix} 0.5 & 0 \\ 1.5 & 0.1 \end{pmatrix}, \quad A_{2m+1} = A_{2m}^T, \quad m = 0,\ 1.\ 2, \ldots.$$

Show that the spectral radius of $A_{2m+1}A_{2m}$ is greater than unity, and that the recurrence relation $y_{n+1} = A_n y_n$ has solutions that are unbounded as $n \to \infty$, in spite that the spectral radius $\rho(A_n) = 0.5$, $\forall n$.

(b) Set $B(t) = \ln A_n$, $n \le t < n+1$, $n = 0,\ 1,\ 2, \ldots$. Show that the eigenvalues of $B(t)$ are strictly less than -0.6 for all $t \ge 0$, and that nevertheless the system $y' = B(t)y$ has solutions that are unbounded as $t \to \infty$.

(c) $B(t)$ is in this example discontinuous. Try to modify the example so that $B(t)$ obtains a continuous derivative everywhere.

(d) Let $A(t) = A + B(t)$, where all eigenvalues of A have negative real parts, and $\int_0^\infty \|B(t)\|dt$ is finite. Show that all solutions of the system $y' = A(t)y$ are bounded. Try also to relax the conditions on A and $B(t)$.

Hint: Use the corollary of Theorem 13.7.12.

(e) Consider the system $y' = A(t)y$, where

$$A(t) = \begin{pmatrix} \dfrac{2}{1+t^2} & \dfrac{2}{2+t^2} \\ \dfrac{1}{2+t^3} & \dfrac{1}{2+t^2} \end{pmatrix}.$$

Show that all solutions are bounded as $t \to \infty$, in spite that both eigenvalues are positive for every $t > 0$.

7. Consider the predator-prey problem.

$$y_1' = ay_1 - by_1y_2, \quad y_2' = cy_1y_2 - dy_2, \quad (a, b, c, d \ge 0).$$

(a) Show that an orbit that starts in the first quadrant will remain there.

(b) Show that the origin is a saddle point, and that $p = (d/c, a/b)$ is a stable center for the linear approximation (around p), and determine the period of the solutions of the linear approximation.

(c) If you divide the two differential equations, you obtain a single equation, $dy_2/dy_1 = \ldots$. Show that its general solution is $F(y) = \text{const.}]$, where $F(y) = a\ln y_2 - by_2 - cy_1 + d\ln y_1$. Show that $F(p) - F(y)$ is a Liapunov function. For what values of k does the equation $F(y) = k$ represent closed orbits of the problem? How can you be sure that they are closed?

8. There are many variations of Theorem 13.1.8. The following can be useful in the study of certain types of stiff problems, when $g(t)$ is positive and sufficiently smooth, and $\mu^* \times \big($the local time-scale of $g(t)\big) \ll -1$. Assume that $d\|u\|/dt \le \mu^*(\|u\| - g(t))$, $(\mu^* < 0,\ t \ge a)$.

(a) Show that

$$\|u(t)\| \le \psi_k(t) + \max_{a \le x \le t} g^{(k)}(x)(\mu^*)^{-k} + Ce^{\mu^*(t-a)},$$

where

$$C = \|u(0) - \psi_{k+1}(a)\|, \quad \psi_0(t) = 0, \quad \psi_k(t) = \sum_{p=0}^{k-1} g^{(p)}(t)(\mu^*)^{-p}.$$

(cf. (13.8.4).)

Hint: Show this first for $k = 0$. Then derive a similar differential inequality for $\|u(t)\| - \psi_k(t)$, where $g(t)$ is replaced by something else.

(b) Since $\psi_k(t)$ is independent of a, it seems strange that the maximization mentioned in the result should be over the whole interval $[a,t]$. Try to improve this. Also show that the first neglected term is a good error estimate if $|g^{(k)}(t)/g^{(k+1)}(t)| \ll |1/\lambda|$.

9. Derive a bound for the difference between a solution $y(t)$ of the differential system,

$$\epsilon y' = f(t,y), \qquad (0 < \epsilon \ll 1), \tag{13.7.15}$$

and the solution $z(t)$ of the reduced problem, $f(t,z) = 0$, in terms of ϵ and upper bounds for $\|\partial f/\partial t\|$ and $\mu(\partial f/\partial y)$ in some suitably defined domain. The latter is called μ^* and is assumed to be negative. Show that this difference is $O(\epsilon)$ for $t > (\epsilon/|\mu^*|)\ln(1/\epsilon)$, i.e. "after a fast transient" or "outside a thin boundary layer".

Hint: By the chain rule, $\partial f/\partial z \;\; z'(t) + \partial f/\partial t = 0$. Rewrite the reduced problem as a differential system: $\epsilon z'(t) = f(t,z) + \epsilon z'(t), \qquad (0 < \epsilon \ll 1)$.

10. Verify the statements of Theorem 13.7.15.

11. Set $B = (I - A)^{-1}(I + A)$.

(a) For inner product norms, show that if $\mu(A) < 0$ then $\|B\| < 1$.

Hint: First show that $y = Bx \;\; \Rightarrow \;\; y - x = A(y + x)$. Then show that $\|y\|^2 < \|x\|^2$, if $x \neq 0$.

(b) Show by an example that this is not generally true for the max-norm.

(c) Set $m = (1 + \mu(A))/(1 - \mu(A))$. For inner product norms, show that $\|B\| \leq m$, if $0 \leq \mu(A) < 1$, while this is not generally true if $\mu(A) < 0$.

12 Let $(u,v) = u^H H v$, $|u|^2 = (u,u)$, and let $\|\cdot\|$, $\mu(\cdot)$ be, respectively, the subordinate matrix norm and the subordinate logarithmic norm.

(a) Show that $\mu(A) < 0 \;\; \leftrightarrow \;\; A^H H + HA$ negative definite, and that $\mu(A) = \max\{\kappa : \det(A^H H + HA - 2\kappa H) = 0\}$.

Hint: Apply the results proved in Theorem 13.7.11. For the latter statement, either use the Lagrange multiplier rule, or reduce the problem to the l_2-case for the matrix RAR^{-1}, by the Cholesky decomposition $H = R^H R$ and the substitution $w = Ru$.

(b) Show that $\|A\|^2 = \max\{\kappa : \det(A^H H A - \kappa H) = 0\}$.

13. Consider the matrix differential equation, $U' = J(t)U$, $U(0) = I$.
Note that $U(t+\epsilon) = \big(I + \epsilon J(t)\big)U(t) + o(\epsilon)$. For any operator norm, note that

$$\|I + \epsilon J\| = 1 + \epsilon\mu(J) + o(\epsilon), \quad d\|U(t)\|/dt \leq \mu(J(t))\|U(t)\|.$$

Show that $\det(I + \epsilon J) = 1 + \epsilon \,\mathrm{trace}\,(J) + o(\epsilon)$, and that $d(\det U(t))/dt = \mathrm{trace}\, J(t)\ \det U(t)$. Note that if $y' = f(t,y)$, $J(t) = f'_y(t, y(t))$, then $\mathrm{trace}\, J(t) = \sum \partial f_i/\partial y_i = \mathrm{div}\ f$ at $y = y(t)$.
Show that these results hold for the matrix differential equation $U' = U\,J(t)$ too.

14. From the classical Perron-Frobenius theorem (Theorem 10.2.12) we have the following: If P is a matrix with *strictly positive* elements, then P has a positive eigenvalue r, that we call the *Perron value*, with the following properties:

(i) r is a simple root of the characteristic equation.

(ii) r has a strictly positive eigenvector w, that we call the *Perron vector*.

(iii) If λ is any other eigenvalue of P, then $|\lambda| < r$.

The first two properties hold also for a matrix A that is *irreducible* and has *non-negative* elements, but the inequality of Property (iii) becomes $|\lambda| \leq r$.

(a) Find a 2×2 irreducible matrix with non-negative elements that does not possess Property (iii), unmodified. Also find a 2×2 matrix with non-negative elements that does not possess Properties (i) and (ii).

(b) Derive from these results a modified form concerned with $\max \Re\lambda$ instead of $\max|\lambda|$, for any irreducible matrix B, such that $b_{ij} \geq 0$, when $i \neq j$.

Hint: Apply the Perron-Frobenius theorems to a matrix $A = I + \epsilon B$ with positive elements, and let $\epsilon \to 0$.

(c) Let A and B satisfy the respective conditions stated above. Take the coordinates of the Perron vector of A as weights in a weighted maximum norm. Prove that the subordinate matrix norm is efficient for the matrix A. Then formulate and prove the analogous result for the subordinate logarithmic norm of the matrix B.

(d) Let $A = [a_{ij}]$ be irreducible, and let $C = [c_{ij}]$ be a complex matrix, such that $|c_{ij}| \leq a_{ij}$, then $\rho(C) \leq \rho(A)$, where $\rho(\cdot)$ denotes the spectral radius. Formulate and show the analogous result for the matrix B, (where the non-negativity is required for the off-diagonal elements only).

Hint: Use the norms discussed in (c).

(e) Find a weighted max-norm that produces an efficient logarithmic norm for the matrix

$$\begin{pmatrix} -3 & 4 & 6 \\ 0 & -2 & 5 \\ 0 & 0 & -1 \end{pmatrix}.$$

Comment: This matrix is *reducible*, but it can be handled according to the remarks after Theorem 13.7.12.

15. Let $T(t)$ be a real orthogonal matrix for every t, with the derivative $T'(t)$. Show that $\mu_2(\pm T^{-1}T') = 0$. (The application of Theorem 13.7.14 is thus as simple and sharp as it should be in this case.)

Hint: Show that $T^{-1}T'$ is a skew-symmetric matrix.

Computer Execrcises

1. Treat Problem 13.1.8 (A population with crowding and toxines) with Runge's 2nd order method. Start with $y_1(0) \ll TOL$, $y_2(0) = 0$.
Run the following cases long enough to show the limits as $t \to \infty$:

$a = b = 0.5$; $a = b = 0.1$; $a = 0.5,\ b = 0$; $a = 0.1,\ b = 0$.
Plot y_1 versus t on the same sheet for all the four cases.
Plot y_2 versus y_1 in the first two cases.

2. Treat Problem 13.1.2 with Runge's 2nd order method. You are likely to encounter some trouble with values of $y > 1$, due to the inevitable computational errors. How can you rewrite the problem in order to avoid this trouble (without using the known exact solution)?

3. Test experimentally the result of Problem 3 above, for some representative choices of λ and α.

4. Solve by computer the equation $y' = t^2 - y^2$, $y(0) = 1$. Choose the tolerance so that you can rely on 4 decimal places. Plot the difference between $y(t)$ and the approximations mentioned above in Problem 4.

5. Solve by computer the "rectifier problem", (13.8.10). In particular, determine the minimum value of $y(t)$. Choose TOL and various values of ϵ, so that you can test the hypothesis that $\min y(t) \sim C\sqrt{\epsilon}$. If it seems true, estimate C to about 1% relative accuracy. (Extrapolate appropriately to $\epsilon = 0$.)

Hint: Remember that the right hand side is zero at the minimum point.
Also prove theoretically that $y(t) > 0,\ \forall t > 0.1$.

6. The Brusselator problem is a chemical system with a limit cycle. A simple version, see Hairer et al. [7, 1993, p.112]: is described by the system,

$$y_1' = a + y_1^2 y_2 - (b+1)y_1, \quad y_2' = by_1 - y_1^2 y_2.$$

Choose $a = 1,\ b = 3$, and show that the critical point is unstable for this choice. Compute one orbit with the origin as starting point, and another orbit that starts near the critical point. You will find that both orbits approach the same limit cycle. Estimate its period to about 1% relative accuracy.

7. Lorenz's example of a chaotic motion. See Fig. 13.7.5b. The following equations occurred in Theoretical Meteorology, see Hairer et al. [7, 1993, p.117]:

$$y_1' = -\sigma y_1 + \sigma y_2, \quad y_2' = -y_1 y_3 + r y_1 - y_2, \quad y_3' = y_1 y_2 - b y_3,$$

with initial conditions: $y_1(0) = -8$, $y_2(0) = 8$, $y_3(0) = r - 1$.
Take e.g. $b = 8/3$, $\sigma = 10$, $r = 28$, and run the problem over the interval $0 \le t \le 30$, with $TOL = 0.001$ and 0.002. Plot y_1 versus t. You are likely to find big differences in the results.
Plot also the projections of the orbit into the coordinate planes, and try to obtain an idea of the behavior of the orbit in $\mathbf{R^3}$.
Finally plot the intersection of the orbit with the plane $y_3 = r - 1$, a so-called *Poincaré section*, a useful picture for the theoretical analysis of a non-linear system.

8. The equation of an undamped pendulum $\phi'' + \frac{1}{\pi}\sin\pi\phi = 0$ can be rewritten as a system, $y_1' = \frac{1}{\pi}y_2$, $y_2' = -\sin(\pi y_1)$. The left part of Fig. 13.7.6 is a phase plane plot with 14 orbits for this system.
Similarly, the equation of a damped pendulum $\phi'' + 0.1\phi' + \sin\phi$ can be written as a system, $y_1' = y_2$, $y_2' = -0.1y_2 - \sin y_1$. The right part of Fig. 13.7.6 is a

Figure 13.7.6. *Phase plane plots for an undamped and a damped pendulum.*

phase plane plot with initial values, $y_1(0) = 0$, $y_2(0) = \sqrt{2a}$ for 15 values of a: $a = 0.5 : 0.5 : 5$ and $6 : 1 : 10$.

(a) Run the undamped case with $y_1(0) = 0$, $y_2(0) = -0.05, 0.95, 1.95$, and determine the period to about 1% accuracy. Explain theoretically the change of the character of the orbits that occurs at $y_2(0) = 2$.

(b) For the damped pendulum, determine by numerical experiment (to about 1% accuracy) the initial speeds needed at the bottom point in order that the pendulum should be able to do $1, 2, 3, 4, 5$ complete revolutions, before it starts to oscillate back and forth around the bottom point.

Hint: Run the system backwards in time from one (or more) appropriate starting point(s). Repeat the run with other tolerances and slightly different starting points in order to estimate the accuracy.

9. (a) Run the system $y_1' = (1 + ay_1)(y_2 - y_1)$; $y_2' = \frac{1}{2}y_1 - y_2$; $y_3' = y_2$, with initial values $y_1(0) = 0$; $y_2(0) = 1$; $y_3(0) = 0$, for a few values of the parameter a. Choose the tolerance and the final value of t, so that $\lim_{t\to\infty} y_3(t)$ can be determined to three correct decimals. The result will probably surprise you. Make a conjecture about $\lim y_3(t)$, and try to prove it. Try also to generalize the result to a more general system of ODEs.

(b) The following example may give a hint to the proof of the conjecture. The differential equation $y' = -e^y y$ is separable, but the expression for the exact solution is rather difficult to handle. Show instead by a rather simple manipulation of the differential equation that $\int_0^\infty y(t)dt = 1 - e^{-y(0)}$.

10. The Mathieu equation reads, $u'' + (a + b\cos 2\pi t)u = 0$, see, e.g., Coddington and Levinson [2, 1955, pp. 218-220]. Determine experimentally, if $a = 1$, for what values of b all solutions of this equation are bounded. (Or, more ambitiously, find the "stability region" for $(a, b) \in \mathbf{R}^2$.)

Hint: Rewrite the equation as a system, and start at $t = 0$. Note that its fundamental matrix $U(t)$ satisfies the conditions,

$$U(t) = U(t-1)U(1) = \ldots = U(t^*)U(1)^n,$$

where $t = n + t^*$, $0 \leq t^* < 1$. The spectrum of $U(1)$ is therefore the crucial thing. Note that $\det U(t) = 1$, according to Problem 22. So, the question is reduced to finding out experimentally for what values of b, a certain condition is satisfied by $U(1)$. What condition?

13.8 General Concepts and Theorems for ODEs

13.8.1 Variational Equations and Perturbation Expansions

The iteration used in the proof of Theorem 13.1.1 was based on the rewriting of the differential equation $dy/dt = f(y)$ in terms of an integral operator that is bounded and, under appropriate conditions, contractive. Would it have been possible to use a different iterative scheme? For example, let $y_0(t)$ be a solution of the algebraic equation $f(t, y(t)) = 0$, and define

$$f(t, y_i(t)) = \frac{dy_{i-1}(t)}{dt}, \quad i = 1,\ 2,\ 3\ldots \tag{13.8.1}$$

(with appropriate conditions that make $y_i(t)$ unique).

The answer is *"No"*, because the differential operator $\frac{d}{dt}$ is an unbounded operator, at least in any function space that contains exponentials $y(t) = e^{\alpha t}$ for some arbitrarily large values of $|\alpha|$, (since for such functions $\|\frac{d}{dt}y\| = |\alpha|\|y\|$). The differential operator $\frac{d}{dt}$ is a bounded operator in some particular function spaces only, such as the space of polynomials of a fixed degree or a space of so-called band-limited functions, i.e. Fourier transforms of functions with a compact support.

Nevertheless the iterative scheme (13.8.1) is interesting for special classes of differential systems, called **separably stiff systems** or singular perturbation problems. First consider a special case:

$$y' = \lambda(y - ce^{\alpha t}), \quad \alpha \in \mathbf{C}. \tag{13.8.2}$$

This equation has the particular solution $y(t) = c(1 - \alpha/\lambda)^{-1}e^{\alpha t}$ Let $y_0(t) = ce^{\alpha t}$. Then $y_i(t) = c_i e^{\alpha t}$ where $c_0 = c$, $\lambda(c_i - c)\alpha e^{\alpha t} = c_{i-1}\alpha e^{\alpha t}$, $i = 1, 2, 3, \ldots$, i.e., $c_i = c + c_{i-1}\alpha/\lambda$. By induction

$$c_i = c\left(1 + \frac{\alpha}{\lambda} + \frac{\alpha^2}{\lambda^2} + \ldots + \frac{\alpha^i}{\lambda^i}\right) \to (1 - \frac{\alpha}{\lambda})^{-1},$$

if $|\alpha| < |\lambda|$, *i.e.,* y_i *converges towards the particular solution mentioned above.* Note that, if $\Re(\lambda) < 0$ any solution of (13.8.2) will approach this particular solution at the rate of $e^{\lambda t}$ as t increases.

When the iteration formula (13.8.1) is applied to the more general equation

$$y' = \lambda(y - g(t)), \quad y(a) = c, \quad \Re(\lambda) < 0, \tag{13.8.3}$$

with $y_0(t) = g(t)$, each iteration yields a new term in the expansion,

$$y(t) \sim g(t) + \lambda^{-1}g'(t) + \lambda^{-2}g''(t) + \ldots + \lambda^{-k+1}g^{(k-1)}(t) + \ldots. \tag{13.8.4}$$

We first note that this is (at most) a particular solution of (13.8.3), but it can then approximate any solution when $\Re(\lambda)(t-a) \ll -1$. Moreover, it is common that this expansion is *semiconvergent* only (in the sense of Sec. 3.1.8). The expansion is to be truncated, when the terms do not decrease rapidly any longer, in absolute value. Another derivation of the expansion is indicated in Problem 13.7.8, together with a remainder. It shows that the first neglected term provides a useful error estimate, as long as the second neglected term is much smaller, in absolute value, than the first, i.e., if $|g^{(k)}(t)/g^{(k+1)}(t)| \gg |1/\lambda|$ or, in other words, if the local time scale $\tau_q(t; g, k) \gg |1/\lambda|$. It is worth noting that it often happens, in the applications we have in mind, that this expansion is *not* alternating, and that, for a fixed t, $\tau_q(t; g, k) \to 0$ slowly, as $k \to \infty$. Take, for example, $g(t) = 1/t$. Then $\tau_q(t; g, k) = t/(k+1)$, and the expansion reads

$$y(t) \sim \frac{1}{t} - \frac{1}{\lambda t^2} + \frac{2!}{\lambda^2 t^3} - \ldots \pm \frac{(k-1)!}{\lambda^{k-1}t^k} + \ldots.$$

This reminds of the alternating semiconvergent expansion studied in Example 3.2.14, but now the most interesting case is when $\lambda < 0$, $t > 0$, i.e., when all terms are positive. Now the first neglected term is a useful error estimate only if $k + 1 \ll |\lambda t|$.

Note, assume that $\lambda < 0$ that

$$g(t + \lambda^{-1}) \approx g(t) + \lambda^{-1}g'(t) + \frac{1}{2}\lambda^{-2}g''(t) + \ldots.$$

By comparison of this with the expansion (13.8.4), we find that the graph of $y(t)$ can be looked upon as a translation $g(t - |\lambda|^{-1})$ of the graph of $g(t)$, with an error of $O\big(|\lambda\tau|^{-2}\big)$, where τ is the local time scale of $g(t)$, provided that the transient proportional to $e^{\lambda(t-a)}$ has died out, i.e., for $t - a > |\lambda|^{-1} \ln 1/TOL$, or $t - a > |\lambda|^{-1} \ln |\lambda|$ depending on the criterion used for deciding that a term is negligible.

Equation (13.8.4) is valid also if we substitute a matrix A for λ, such that $\Re\lambda < 0$ and $|\lambda\tau| \gg 1$ for all eigenvalues of A. Now consider a nonlinear system, where ϵ corresponds to $1/|\lambda|$,

$$\epsilon y' = F(t, y), \quad \epsilon > 0. \tag{13.8.5}$$

Assume that $F(t, y)$ and its partial derivatives of low order are bounded, and that there exists a solution $y_0(t)$ of **the reduced problem** $F(t, y) = 0$, such that the Jacobian $J(t) = \partial F/\partial y$ at $y = y_0(t)$ is non-singular and has a negative logarithmic norm $\mu(J(t)) < \mu^* < 0$. (In fact, the non-singularity follows from the latter assumption, see Theorem 13.1.11.) Let then $y_1(t)$ be defined by the equation

$$\epsilon y_0'(t) = F(t, y_1(t)). \tag{13.8.6}$$

Take the total derivative of the equation $F(t, y_0(t)) = 0$ with respect to t, i.e., $\partial F/\partial t + J(t)y_0' = 0$. This determines $y_0'(t)$. By Taylor's formula, we can now write (13.8.6) as follows:

$$\epsilon y_0'(t) = F(t, y_1(t)) - F(t, y_0(t)) = J(t)(y_1(t) - y_0(t)) + O(\|y_1 - y_0\|^2). \quad (13.8.7)$$

It can then be shown that

$$y_1(t) = y_0(t) + \epsilon J(t)^{-1} y_0'(t) + O(\epsilon^2). \quad (13.8.8)$$

and that $y_2(t) - y_1(t) = O(\epsilon^2)$. It is then conceivable that $y(t) - y_1(t) = O(\epsilon^2)$, when a transient, that decays faster than $e^{\mu^*(t-a)}$, has died out.

If (13.8.5) is a *single nonlinear equation*, $J(t) \leq \mu^* < 0$ then we conclude from (13.8.8) that

$$y(t) = y_1(t) + O(\epsilon^2) = y_0(t - \epsilon/|J(t)|) + O(\epsilon^2) \quad (13.8.9)$$

i.e., *the graph of $y(t)$ is approximately a translation of the graph of $y_0(t)$, after the transient, but the delay $\epsilon/|J(t)|$ depends on time.*

A few things are worth to be noted for the case of a single equation:

(a) The left hand side of (13.8.7) is smaller than the absolute value of each of the terms $J(t)y_1(t)$, $J(t)y_0(t)$, by a factor that can be described as the ratio of the smallest local time constant of the system to the local time scale of the approximate solution $y_0(t)$. So, when the solution of a single differential equation has become stiff, the differential equation describes *a moving approximate balance between positive and negative terms on the right hand side.*

(b) The graphs of $y(t)$ and $y_0(t)$ intersect at points where $y'(t) = 0$. (Why is that no contradiction to "the translation point of view"?)

(c) The asymptotics is valid only when the delay $\epsilon/|J(t)|$ is much smaller than the local time scale. In particular, it becomes unreliable when t approaches a point, where $J(t)$ is zero.

These points of view, applied to individual equations of a system, where the other variables are considered as driving terms, can often provide a good insight in what is going on in the system. This makes it natural *to extend the use of the term stiffness to individual variables or subsystems of a larger system.*

The function defined by a few iterations according to (13.8.1) is close to a solution of (13.8.6) if $\epsilon \ll \tau(t)$. If $\mu^* < 0$, it is **attracting** to other solutions of the same system, as t increases. If the Jacobian has a positive eigenvalue, the iterative process may still be semi-convergent, but the resulting function is **repelling** to the other solutions of the system. In non-linear problems the expressions for the successive iterates quickly become messy, but Problem 13.7.8 indicates that rather simple bounds can be obtained.

Example 13.8.1.

Figure 13.8.1. *The solution of* $y' = a(1/t - y)$, $y(1) = 0$ *for* $a = 100, 10, 5$.

Fig. 13.8.1 shows the solution of $y' = a(1/t - y)$, $y(1) = 0$ for $a = 100, 10, 5$. The curve $y = 1/t$ is not drawn, but it proceeds from the upper left corner through the top point of the leftmost curve and continues then immediately to the left of the leftmost curve. It is seen that, after the transient, the other two curves are delayed about a^{-1} after the left curve, as predicted above.

Figure 13.8.2. *Solution of "the rectifier equation".*

Example 13.8.2.

Fig. 13.8.2 shows the solution of "the rectifier equation"

$$\epsilon \frac{dy}{dt} = \left((\sin 2\pi t)^2 - y^2 \right), \quad y(0.1) = 0.5, \tag{13.8.10}$$

for $\epsilon = 0.0001, 0.01, 0.02$. The reduced problem has two solutions: $y_0(t) = \pm \sin 2\pi t$. Note that $f'_y(t,y) = -2y/\epsilon$, hence $y_0(t) = |\sin 2\pi t|$ is attracting, while the branch $y_0(t) = -|\sin 2\pi t|$ is repelling. The points where $y_0(t) = 0$ are branch points, where $J(t) = 0$. The smallest time constant is $\epsilon/|y|$, the local time scale is approximately $\frac{1}{2\pi}$, and $y(t) \approx |\sin 2\pi(t - \epsilon/(2y))|$, except in the transient and in the neighborhood og the branch-points. It appears that the asymptotics becomes unreliable when $y = O(\sqrt{\epsilon})$, where $y(t)$ has a minimum, with a sharp turn. It may be conjectured that $\min y(t) \sim c\sqrt{\epsilon}$, $(\epsilon \to 0)$. It is not quite trivial to determine c by an asymptotic analysis. If one does it numerically instead, by running the problem for different values of ϵ (Computer exercise 13.7.5), one should avoid to choose $TOL > \epsilon$ (say). It has happened also with widely used programs for stiff problems and a careless choice of TOL that the computed solution becomes close to $\sin 2\pi t$ instead of $|\sin 2\pi t|$. This can be very misleading, in particular since the erroneous solution is much smoother and looks more trustworthy than the correct solution! Roughly speaking, the step size has become so big for $t > 0.25$ that the computed points never come close enough to $y = 0$, for the program to recognize the change of local time scale. It may also fail to recognize the repelling nature of the wrong branch for $y < 0$, since the stability region of many numerical methods for stiff problems contains part of the right half plane where $\Re\lambda h > 0$. (See Example 13.2.8.)

Let $z(t)$ be a function that satisfies the differential system,

$$\frac{dz}{dt} = f(z) + r(t),$$

where $r(t)$ is a piecewise continuous perturbation. Let $y(t)$ be a solution of the system $dy/dt = f(y)$. Set

$$u(t) = z(t) - y(t).$$

Then $u = u(t)$ satisfies the differential equation,

$$\frac{du}{dt} = f(y(t) + u) - f(y(t)) + r(t), \tag{13.8.11}$$

called the exact or the nonlinear variational equation.

Lemma 13.8.1.

The non-linear variational equation (13.8.11) *can be written in* **pseudo-linear** *form*

$$\frac{du}{dt} = J(t,u)u + r(t),$$

where $J(t,u)$ is a neighborhood average of the Jacobian matrix,

$$J(t,u) = \int_0^1 f'\big(y(t) + \theta u\big)d\theta. \tag{13.8.12}$$

Proof. By the chain rule,

$$\frac{\partial f(y(t) + \theta u)}{\partial \theta} = f'(y(t) + \theta u)u,$$

hence

$$f(y(t)+u)-f(y(t))=\int_0^1 f'(y(t)+\theta u)u d\theta = J(t,u)u.$$

□

Systems of ODEs often contain *parameters*, and it may be of interest to study how the solution $y(t;p)$ depends on a parameter vector p. The following result can be derived formally by the application of the chain rule to (13.8.13). A more thorough treatment is found in the first three chapters of Coddington-Levinson [2].

Theorem 13.8.2.

Let $y=y(t;p)$ be the solution of the initial value problem,

$$dy/dt = f(t,y;p), \qquad y(a)=c(p), \tag{13.8.13}$$

where p is a vector of parameters. Assume that f is a continuous function of t, a differentiable function of y and p everywhere, and assume that $\|\partial f/\partial y\| \le L$ everywhere.

Then, $y(t;p)$ is a differentiable function of t and p. The matrix valued function $U(t)=\partial y/\partial p$ is determined by the non-homogeneous linear variational equation,

$$dU/dt - J(t)U = \partial f/\partial p, \qquad U(0)=\partial c/\partial p.$$

$J(t)=\partial f/\partial y$ is evaluated at $y=y(t;p)$.

If f is an analytic function of $(y;p)$ in some complex neighborhood of $(y_0;p_0)$, then $y(t;p)$ is an analytic function of p that can be expanded into powers of $\epsilon = p-p_0$ with a non-vanishing radius of convergence, sometimes called a **regular perturbation expansion**.

Comments: If the differentiability conditions etc. hold only in a subset D, then the first result remains true, as long as the orbits stay in D. It would have been more correct to write $J(t;p)$ instead of $J(t)$. We avoid this in order not to confuse with the neighborhood average $J(t,u)$ introduced above.

The regular perturbation expansion is called so, since the same number of initial values are required for $p=p_0$ as for $p=p_0+\epsilon$. It can be shown, see e.g. Lin and Segel [1976], that the coefficients in the expansion, which are functions of t, can be computed recursively, by the solution of the linear variational equation with the same matrix $J(t)$, but the right hand side now depends on the coefficients previously computed. In practice, it may be easier to compute numerical solutions for a few (judiciously selected) values of ϵ, *with the same step size sequence (!), in order to rely on approximate cancellation of truncation errors*, see a warning example in Hairer et al. [1997, p. 183]. A few coefficients can then be computed by a polynomial fitting program.

Notice that the expansion (13.8.4) does not fall into this category. Surely, it may be considered as a particular case of an ϵ-expansion (with $\epsilon=|1/\lambda|$), applicable to the nonlinear system (13.8.5), i.e., $\epsilon y' = F(t,y)$. There is, however, a great difference. The problem obtained for $\epsilon=0$, is an algebraic system (i.e., $F(t.y)=0$

or $g(t) - y = 0$, respectively) that cannot accept arbitrary initial values, as the problem does for $\epsilon \neq 0$. This is called a **singular perturbation** problem. The recurrence relation for the coefficients is different, as exemplified by (13.8.6). If $0 < \epsilon \ll 1$ and $\mu(\partial F/\partial y) < 0$, the motion quickly forgets its initial (or boundary) values, during a transient (or in a boundary layer). Singular perturbation problems are a particular type of stiff problems.

13.8.2 Difference Equations and Matrix Power Boundedness

Recursion formulas of the type

$$y_{n+k} = f(y_n, y_{n+1}, \ldots, y_{n+k-1}, n), \tag{13.8.14}$$

play a large part in the numerical solution of differential equations. Its solution is uniquely determined when k initial values $y_0, \ldots, y_{k-1}$ are given, and these can be chosen arbitrarily—at least if the function f is defined in the whole space. Such recursions can also be written in the form of a kth order difference equation

$$\Delta^k y_n = g(y_n, \Delta y_n, \ldots, \Delta^{k-1} y_n, n). \tag{13.8.15}$$

The properties of such equations were discussed in Sec. 3.2.3.

We next consider nonhomogeneous **linear systems of first-order difference equations**, written in vector-matrix form:

$$y_{n+1} = A_n y_n + x_n,$$

where $y_n, x_n \in \mathbf{R}^s$ and $A_n \in \mathbf{R}^{s\times s}$. If the initial value y_0 is given, then by induction we obtain

$$y_n = P_{n,0} y_0 + \sum_{j=1}^{n} P_{n,j} x_{j-1}, \tag{13.8.16}$$

$$P_{n,j} = A_{n-1} A_{n-2} \ldots A_j, \qquad P_{n,n} = I. \tag{13.8.17}$$

This is a discrete analog of (13.1.15), (which may make the formula for the continuous case more intelligible).

If the matrices A_i are non-singular then we may write $P_{n,j} = P_n P_j^{-1}$. In particular, if all A_i are equal to A, then

$$y_n = A^n y_0 + \sum_{j=1}^{n} A^{n-j} x_{j-1}, \tag{13.8.18}$$

This formula holds, of course, also in the scalar case.

The following analog to Theorem 13.1.8 is easily proved by induction.

Theorem 13.8.3.

The solutions of a "pseudo-linear" system of difference equations,

$$u_{n+1} = A(n, u_n) u_n + r(n, u_n),$$

satisfy the inequality, $\|u_{n+1}\| \le \|A(n,u_n)\| \cdot \|u_n\| + \|r(n,u_n)\|$. *Let* D_n *be a convex domain in* $\mathbf{R}^s$, *and assume that*

$$\|A(n,w)\| \le a_n, \quad \|r(n,w)\| \le b_n, \quad \forall w \in D_n.$$

Then, $\|u_n\| \le \psi_n$, *where* ψ_n *is a solution of a scalar difference equation,* $\psi_{n+1} = a_n\psi_n + b_n, \quad \psi_0 = \|u_0\|$, *as long as the bound derived from this guarantees that* $u_n \in D_n$.

If $a_n = a, b_n = b$, *independently of* n, *then*

$$\|u_n\| \le \psi_n = \begin{cases} a^n\|u_0\| + \dfrac{b(1-a^n)}{1-a}, & \text{if } a \ne 1; \\ \|u_0\| + bn, & \text{if } a = 1. \end{cases} \tag{13.8.19}$$

If $a < 1$, $\|u_n\| \le \max\{\|u_0\|, b/(1-a)\}$, $n \ge 0$.

Proof. The proof is left for Problem 3. □

The natural relation is via a step size parameter h, such that $nh = t, \quad a = e^{\mu^* h}$. In particular, $a = 1$ corresponds to $\mu^* = 0$.

Example 13.8.3.

In the analysis of numerical methods for differential equations, see e.g. Sec. 13.4, one sometimes encounters a vector sequence that satisfies a difference equation of the form,

$$y_{n+1} = (A + hB_n)y_n + r_n, \qquad n = 0, 1, 2, \ldots,$$

where h is a step length. We want to estimate $\|y_n\|$.

Assume that, for some l_p-norm, $\|B_n\| \le K$, $\|r_n\| \le ch^{p+1}$, $\forall n$, and that A is a constant matrix with spectral radius equal to 1, such that the eigenvalues of unit modulus are non-defective. Then, by Theorem 10.2.9, there exists a norm $\|y\|_T = \|T^{-1}y\|$, such that $\|A\|_T = 1$, for the subordinate matrix norm. Then $\|B_n\|_T \le K_1$, where

$$K_1 = \kappa(T)K, \quad \kappa(T) = \|T\| \cdot \|T^{-1}\|,$$

and $\kappa(T)$ is the condition number. It follows that

$$\|y_{n+1}\|_T \le (1 + K_1 h)\|y_n\|_T + ch^{p+1}\|T^{-1}\|.$$

This is the situation treated by Theorem 13.8.3, with $a_n = a = 1 + K_1 h, \quad b_n = b = ch^{p+1}\|T^{-1}\|$. By (13.8.19),

$$\|y_n\|_T \le (1 + K_1 h)^n \|y_0\|_T + \frac{(1 + K_1 h)^n - 1}{K_1 h} ch^{p+1}\|T^{-1}\|$$

Since $\|y\| \le \|T\| \cdot \|y\|_T, \quad \|y\|_T \le \|T^{-1}\| \cdot \|y\|$, we can return to the original norm:

$$\|y_n\| \le \|T\| \left((1 + K_1 h)^n \|y_0\| \cdot \|T^{-1}\| + \frac{(1 + K_1 h)^n - 1}{K_1} ch^p \|T^{-1}\| \right)$$

Finally, we use the definition of condition number and the following relations,

$$1 + K_1 h \le e^{K_1 h}, \quad nh = t, \quad e^{K_1 t} - 1 \le K_1 t e^{K_1 t}, \quad \text{for } t \ge 0,$$

in order to the obtain the result:

$$\|y_n\| \le \kappa(T) e^{K_1 t} (\|y_0\| + tch^p). \tag{13.8.20}$$

If we had not used to the T-norm internally, the result is likely to have become less sharp, in many cases even useless. Still the positive constant K_1 is for many applications a weak point of this result. A stronger bound may be obtained by the use of a sequence of similarity transformations T_n according to the following general theorem that is a discrete analog of Theorem 13.7.14. For the sake of simplicity it is formulated for a linear system, but the generalization to a pseudo-linear system is straight-forward (see Theorems 13.7.14 and 13.8.3). Recall that $\|A\|_T = \|T^{-1}AT\|$.

Theorem 13.8.4.

Consider the linear system $y_{n+1} = A_n y_n + r_n$. Let T_n be a sequence of non-singular matrices. Then $\|y_n\| \le \psi_n$, where ψ_n is defined by the difference equation

$$\psi_{n+1} = \kappa(T_{n+1}^{-1} T_n) \|A_n\|_{T_n} \psi_n + \kappa(T_{n+1}) \|r_n\|, \quad \psi_0 = \kappa(T_0) \|y_0\|.$$

If $\kappa(T_{n+1}^{-1} T_n) \|A_n\|_{T_n} \le a$, and $\kappa(T_{n+1}) \|r_n\| \le b$, then the bounds given in (13.8.19) are valid, and the behavior of ψ_n is illustrated by Fig. 13.1.3.

Proof. The proof is left for Problem 3. □

The question whether a matrix sequence $\{A^n\}_0^\infty$ is bounded or not is often of interest, e.g. in the application of (13.8.18). We also saw in the previous example that it is interesting to know whether there is a norm such that $\|A\| \le 1$. The following theorem, that is a discrete analog of Theorem 13.7.3, shows, among other things, that these two questions are equivalent.

Theorem 13.8.5. Power Boundedness of a Single Matrix.

Let A be a given square matrix with spectral radius $\rho(A)$. The following statements are equivalent:

(i) *The sequence $\{A^n\}_0^\infty$ is bounded.*

(ii) *All eigenvalues of A are located inside or on the unit circle, and there are no defective eigenvalues on the unit circle.*

(iii) *There exists an operator norm such that $\|A\| \le 1$.*

Proof. We shall establish the equivalence by showing that (i) implies (ii), (ii) implies (iii), and (iii) implies (i). In order to show that (i) implies (ii), we first

consider a Jordan box $J_m(\lambda) = \lambda I + N$, where $N^m = 0$.

$$\|J_m(\lambda)^n\|_\infty = \left\| \sum_{p=0}^{\min(n,m-1)} \binom{n}{p} N^p \lambda^{n-p} \right\| = \sum_{p=0}^{\min(n,m-1)} \binom{n}{p} |\lambda|^{n-p} \geq c n^{m-1} |\lambda|^n.$$

It follows that $\|A^n\|$ can be bounded only if, for every eigenvalue λ, $|\lambda| \leq 1$, and $m = 1$ if $|\lambda| = 1$. This is, in other words, condition (ii).

Next, (ii) implies (iii), by Theorem 10.2.9. Finally, (iii) implies (i), since $\|A^n\| \leq \|A\|^n \leq 1$. □

Difference equations of order k can be written as a system of first-order difference equations. After the substitution $z_n := (y_n, y_{n+1}, \ldots, y_{n+k-1})^T$, the difference equation $y_{n+k} + a_1 y_{n+k-1} + \ldots + a_k y_n = 0$ can be written in the vector-matrix form

$$z_{n+1} = A z_n, \qquad A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_k & -a_{k-1} & -a_{k-2} & \cdots & -a_1 \end{pmatrix}, \tag{13.8.21}$$

where the matrix A is a **companion matrix**. In the literature such matrices appear in different forms, depending, e.g., on the naming of the coordinates of z_n or the coefficients of the difference equation, see, e.g., Problem 10.1.7 With our convention, the difference equation and the matrix A have the same characteristic equation, i.e. the roots of the characteristic equation of the difference equation are the eigenvalues of its companion matrix. The eigenvector belonging to the eigenvalue λ reads $(1, \lambda, \lambda^2, \ldots, \lambda^{k-1})^T$. It is unique, hence *any multiple eigenvalue of a companion matrix A is defective.* The verification of these statements is left for Problem 1.

Theorem 13.8.6.

The following **root conditions** *is necessary and sufficient, for the boundedness of all solutions of the difference equation*

$$y_{n+k} + a_1 y_{n+k-1} + \ldots + a_k y_n = 0,$$

or equivalently the power boundedness of the companion matrix (13.8.21)*, as $n \to \infty$:*

(i) *All roots of the characteristic equation should be located inside or on the unit circle;*

(ii) *The roots on the unit circle should be simple.*

Proof. This follows from Theorem 13.8.5, since any multiple eigenvalue of a companion matrix is defective. □

In many areas, notably in the analysis of finite difference methods for partial differential equations, it is important to know, if there is a constant C, such that $\|A^n\| \le C$ hold for an *infinite family of matrices*. It turns out that statement (ii) of Theorem 13.8.5 is not sufficient, and statement (iii) must also be modified, as shown by the following example and theorem.

Example 13.8.4.

A matrix family is defined by $A(\delta) = \begin{pmatrix} 1-\delta & \delta^{1/2} \\ 0 & 1-\delta \end{pmatrix}$, $0 \le \delta \le 1$, where using (10.2.14)

$$A(\delta)^n = \begin{pmatrix} (1-\delta)^n & n(1-\delta)^{n-1}\delta^{1/2} \\ 0 & (1-\delta)^n \end{pmatrix}.$$

For each δ statement (ii) of Theorem 13.8.5 is satisfied, but that tells only that $\|A(\delta)^n\| \le C(\delta)$. We see, however, that

$$\max_{\delta} \|A(\delta)^n\|_\infty \ge \|A(1/n)^n\|_\infty \ge (1+\sqrt{n})(1-1/n)^n \sim n^{1/2}e^{-1} \to \infty, \ n \to \infty.$$

(Since all norms in a finite-dimensional space are equivalent the same final conclusion holds in nay norm.) This is *not* due to the fact that $1-\delta$ is a defective eigenvalue of $A(\delta)$, for if we replace one of the diagonal elements of $A(\delta)$ by $(1-\frac{1}{2}\delta)$, then no member of the matrix family has a defective eigenvalue, but $\|A(\delta)^n\|$ would become even larger. (Note that $A(0) = I$.)

The following important theorem gives necessary and sufficient conditions for the *power boundedness for a family of matrices*, cf. Theorem 13.8.5.

Theorem 13.8.7. The Kreiss Matrix Theorem: Discrete Case.

Consider a matrix family $\mathcal{F} \subset \mathbf{R}^{s\times s}$ for a fixed s. The following four statements are equivalent.

(a) *There exists a constant C such that for all $A \in \mathcal{F}$ and all $n = 1,\ 2,\ 3, \ldots,$*

$$\|A^n\| \le C.$$

(Note that the bound C must be the same for all $A \in \mathcal{F}$.)

(b) *There exists a constant C_1 such that for all $A \in \mathcal{F}$ and all $z \in \mathbf{C}$ with $|z| > 1$, the* **resolvent** *$(A - zI)^{-1}$ exists and*

$$\|(A-zI)^{-1}\| \le \frac{C_1}{|z|-1}, \quad \text{resolvent condition.}$$

(c) *There exist constants C_2, C_3, and to each $A \in \mathcal{F}$, a matrix S such that the condition number $\kappa(S) \le C_2$ and $B = S^{-1}AS$ is upper triangular with*

$$|b_{ij}| \le C_3 \min(1-|b_{ii}|, 1-|b_{jj}|), \quad \text{for } i \ne j.$$

(d) *There exists a constant C_5 and, for each $A \in \mathcal{F}$ a matrix T, such that*

$$\|A\|_T \leq 1, \quad \kappa(T) \leq C_5.$$

Proof. Sketch. The general plan is to show that (a) $\Rightarrow$(b) $\Rightarrow$(c) $\Rightarrow$(d) $\Rightarrow$(a).

We first note that for each $A \in \mathcal{F}$ the eigenvalues are located in the closed unit disc, hence for $|z| > 1$, $A - zI$ is invertible and

$$\|(A - zI)^{-1}\| = \|\sum_{n=0}^{\infty} A^n z^{-n-1}\| \leq C \sum_{n=0}^{\infty} |z|^{-n-1} = \frac{C}{|z| - 1}.$$

Hence (a) $\Rightarrow$ (b). The proof that (b) $\Rightarrow$(c) is the most difficult part, and we refer to Richtmyer and Morton [10, 1967]. The proof that (c) $\Rightarrow$(d) is a more complicated variant of the proof of Theorem 10.2.9, and again we refer to Richtmyer and Morton [10, 1967] for details. Note the important assumption about $\kappa(T)$. (This is automatically satisfied in the single matrix case.) Finally, (d) $\Rightarrow$(a), since

$$\|A^n\| \leq \kappa(T)\|A^n\|_T \leq C_5\|A\|_T^n = C_5.$$

□

In recent applications of this theorem to the study of difference methods to partial differential equations, the resolvent condition (b) is the easiest one to verify. It is therefore interesting that LeVeque and Trefethen has found a direct proof that (b)$\Rightarrow$(a), with $C = 2esC_1$ based on the representation of A^n by a Cauchy integral. (Note that the resolvent appears in this integral.) Their proof is reproduced in Hairer and Wanner [8, 1991, p.349].

There is a continuous version of this theorem that can be heuristically derived from the above discrete version as follows. (Historically, the continuous version was found first.) Set $A = e^{hB}$, $nh = t$. Then $A^n = e^{tB}$, $\|A^{\|} \approx e^{h\mu(B)}$, and the outside of the unit disk in the discrete case corresponds to the positive half-plane, $\Re z > 0$ in the continuous case.

Theorem 13.8.8. The Kreiss Matrix Theorem: Continuous Case.

Consider a matrix family $\mathcal{F} \subset \mathbf{R}^{s\times s}$ for a fixed s. The following four statements are equivalent.

(a) *There exists a constant C such that*

$$\|e^{tB}\| \leq C, \quad \forall B \in \mathcal{F}, \quad \forall t \geq 0.$$

(b) *There exists a constant C_1 such that for all $B \in \mathcal{F}$ and all $z \in \mathbf{C}$* with $\Re z > 0$, *the resolvent $(B - zI)^{-1}$ exists and*

$$\|(B - zI)^{-1}\| \leq \frac{C_1}{\Re z}, \quad \text{resolvent condition.}$$

(c) *There exist constants C_2, C_3, and to each $B \in \mathcal{F}$, a matrix S such that the condition number $\kappa(S) \leq C_2$ and $K = S^{-1}BS$ is upper triangular with*

$$|k_{ij}| + C_3 \max(\Re k_{ii},\ \Re k_{jj}) \leq 0, \quad \text{for } i \neq j.$$

(d) *There exists a constant C_5 and, for each $B \in \mathcal{F}$ a matrix T such that*

$$\mu_T(B) \leq 0, \quad \kappa(T) \leq C_5.$$

13.8.3 More Results on Stability Theory

There are other algebraic criteria and algorithms than given in Sec. 13.2.3 for the investigation of the root condition of a polynomial, perhaps most useful at the study the the root condition and related matters, for algebraic equations containing parameters. They can be used either numerically (floating point operations on vectors of polynomial coefficients) or algebraically (symbolically). There is a short discussion of these two approaches to problems with two parameters, in Example 13.8.7.

We first consider an analogous problem connected with the left half-plane This problem has an interest in its own right for differential equations, but it means more to us that the two problems can be transformed into each other by a Möbius transformation,

$$z = \frac{\zeta + 1}{\zeta - 1}, \quad \zeta = \frac{z + 1}{z - 1}, \quad |\zeta| < 1 \leftrightarrow \Re z < 0, \quad |\zeta| > 1 \leftrightarrow \Re z > 0, \tag{13.8.22}$$

and we set

$$R(z) = \left(\frac{z-1}{2}\right)^k \rho\left(\frac{z+1}{z-1}\right), \Rightarrow \ \rho(\zeta) = (\zeta - 1)^k R\left(\frac{\zeta + 1}{\zeta - 1}\right), \tag{13.8.23}$$

and similarly for $\sigma(\zeta)$, $S(z)$. The polynomials R and S have no common factors (like ρ and σ). In the theory of multistep methods these formulas are called the **Greco-Roman transformation**. It can be seen as a linear coordinate transformation in the linear space of multistep methods (ρ, σ) that provides many simplifications, see Problem 18 and Sec. 13.4.

We need some notions and notations. $\mathbf{C}^+ = \{q : \Re q > 0\}$. $\bar{\mathbf{C}}^+$ is the closure of $\mathbf{C}^+$. It includes also the imaginary axis and ∞. Similarly, $\mathbf{C}^- = \{q : \Re q < 0\}$. The degree p of a polynomial is denoted $\deg p$.

A k-th degree polynomial p, with zeros α_j, $j = 1, 2, \ldots, k$ is called a **Hurwitz polynomial**, if $\alpha_j \in \mathbf{C}^-$, $\forall j$. The closure of the set of Hurwitz polynomials are the polynomials for which $\alpha_j \in \bar{\mathbf{C}}^-$, $\forall j$. The following simple condition is often helpful for the proof that a given polynomial is *not* a Hurwitz polynomial.

Theorem 13.8.9.

A necessary condition for a real polynomial to be a Hurwitz polynomial is that all coefficients are different from zero and have the same sign. This condition is

sufficient for linear and quadratic polynomials only. For the closure of Hurwitz polynomials, a coefficient is also allowed to be zero.

Proof. In the factorization $c\prod(z-\alpha_j)$, a root α_j is either negative, or it occurs together with its complex conjugate in a factor of the form $(z-\alpha)(z-\bar{\alpha}) = z^2+az+b$ where $a > 0$, $b > 0$. The expansion of the product will therefore have positive coefficients only. The statement about sufficiency will appear as a consequence of the necessary and sufficient conditions presented in Theorem 13.8.10 or, more explicitly, in Problem 9 (a). The last statement follows from continuity considerations. □

A rational function f is called a **positive function**, if $\Re z > 0 \rightarrow \Re f(z) > 0$. In other words: it maps $\mathbf{C}^+$ into itself. A useful property follows directly: The function $f \circ g$, *defined by the equation* $f \circ g(z) = f(g(z))$, *is positive if* f *and* g *are so.* Notice that $1/z$ is a positive function, since $\Re\ 1/z = \Re\bar{z}/|z|^2 = \Re z/|z|^2$. Hence *if the function* f *is positive then* $1/f$ *is also positive.* In particular, the rational function S/R is positive if R/S is positive.

When the attribute **real** is added to terms like polynomial, rational function, positive function etc., it means that it is real on the real axis. In other words, it has real coefficients, when expanded about a point in $\mathbf{R}$.

Since $\rho(\zeta)/\sigma(\zeta) = R(z)/S(z)$, it follows from Theorem 13.2.6 that the function $q = R(z)/S(z)$ *maps* $\mathbf{C}^+$ *onto the interior of the complement of* S. In particular, *a linear multistep method is A-stable, iff* $R(z)/S(z)$ *is positive real.* This leads to a connection between Hurwitz polynomials and positive functions.

If R/S *is a positive rational function, then the polynomial* $R - qS$ *is a Hurwitz polynomial for any constant* $q \in \mathbf{C}^-$. An even more important connection is expressed by the following theorem.

Theorem 13.8.10.
Let $p_0(z) = c_0z^k+c_2z^{k-2}+c_4z^{k-4}\cdots$, $\quad p_1(z) = c_1z^{k-1}+c_3z^{k-3}+c_5z^{k-5}\cdots$, *be two real polynomials. Set* $p(z) = p_0(z) + p_1(z)$, *and assume that* p_0 *and* p_1 *have no common divisor. Then* p *is a Hurwitz polynomial, if and only if*

$$p_0/p_1 \quad \textit{is a positive function.} \tag{13.8.24}$$

Proof. We first note, that the statement in (13.8.24) is, by the second and fourth statements of (13.8.22), equivalent to the condition $|p_0(z)/p_1(z)+1|/|p_0(z)/p_1(z)-1| > 1$, i.e., $|p_0(z)+p_1(z)|/|p_0(z)-p_1(z)| > 1$,, i.e.,

$$|p(z)| > |p(-z)|,\ \forall z \in \mathbf{C}^+. \tag{13.8.25}$$

This strict inequality immediately shows that $p(z) \neq 0$ for $z \in \mathbf{C}^+$. Moreover, when z is pure imaginary, the real part is equal to either $p_0(z)$ or $p_1(z)$ (depending on the parity of k) and the other is the imaginary part. Anyway, a pure imaginary zero of $p(z)$ would be a common zero of $p_0(z)$ and $p_1(z)$, and this contradicts the assumption. Hence the "if" part is proved.

The converse is more interesting. Assume that $p(z)$ is a Hurwitz polynomial. A root α_j is either negative, or it occurs together with its conjugate, both in $\mathbf{C}^-$. We

have $p(z) = c_0 \prod (z - \alpha_j)$ and shall establish relation (13.8.25). Then $|p(z)/p(-z)|$ is a product of factors *either* of the form,

$$|z - \alpha|/|-z - \alpha| = |z - \alpha|/|z + \alpha| > 1, \quad \forall z \in \mathbf{C}^+,$$

or of the form

$$\left| \frac{(z-\alpha)(z-\bar{\alpha})}{(-z-\alpha)(-z-\bar{\alpha})} \right| = \left| \frac{z-\alpha}{z+\bar{\alpha}} \right| \left| \frac{z-\bar{\alpha}}{z+\alpha} \right| > 1, \ \forall z \in \mathbf{C}^+.$$

(Draw a picture with five points: $z \in \mathbf{C}^+$, $\alpha \in \mathbf{C}^-$, $\bar{\alpha}$, $-\alpha$, $-\bar{\alpha}$.) This proves (13.8.25) and hence the 'only if' part. □

The division algorithm, described at the end of Sec. 3.4.1 (Continued fractions), applied to the polynomials $p_0(z)$, $p_1(z)$ defined above, yields a sequence of alternatingly odd or even polynomials,

$$\frac{p_m(z)}{p_{m+1}(z)} = d_m z + \frac{p_{m+2}(z)}{p_{m+1}(z)}, \quad m = 0, 1, 2, \ldots, n-1, \quad \deg p_{m+1} = \deg p_m - 1.$$

Since the algorithm is, in fact, identical to the Euclid algorithm for finding common divisors of the polynomials p_0 and p_1, it will, if the assumption of the theorem is satisfied, terminate when p_{m+1} has become a non-zero constant. If $c_0 > 0$, a repeated use of Theorem 13.8.11 (below) tells that *p_0/p_1 is a positive function, hence $p = p_0 + p_1$ is a Hurwitz polynomial, if and only if $d_m > 0$ for all m*, the **Routh criterion**. This algorithm, the **Routh algorithm**, is easily programmed, with vectors for the polynomial coefficients, see Problem 17. If $d_m = 0$, within a tolerance, then p_m is the greatest common divisor of p_0 p_1. This should be examined further. It contains the zeros of p on the imaginary axis, (these are common zeros of p_0 and p_1), but it may also contain real zeros, symmetrically located with respect to the imaginary axis.

The Routh criterion was developed by Routh in 1877, with a completely different derivation. An extension to polynomials with complex coefficients was made in 1946 by E. Frank. See Problem 16 for a different approach, due to Hurwitz 1895.

Theorem 13.8.11.

Let $f(z) = az + g(z)$, where $g(z)$ is regular in $\mathbf{C}^+$ and bounded at ∞. Then $f(z)$ is a real positive function, if and only if $a > 0$, and $g(z)$ is either a positive function or identically zero.

Proof. The "if" part is obvious; we shall prove the deeper "only if" part. First note that, since $g(z)$ is bounded, $\arg f(z) = \arg a + \arg z + o(1)$, as $z \to \infty$. We then note that $\Re w > 0 \ \leftrightarrow \ |\arg w| < \pi/2$. It follows that $|\arg f(z)| < \pi/2$, for all z, such that $|\arg z| < \pi/2$, if and only if $\arg a = 0$, i.e., iff $a > 0$. It then follows that $\liminf_{z \to iy} \Re g(z) = \liminf_{z \to iy} \Re f(z) - 0 \geq 0$ for $z \in \mathbf{C}^+$ and for all real y, (including $y = \pm\infty$). Then, by the minimum principle for harmonic functions, see e.g. Dinghas [5, 1961, p. 303], $\Re g(z) > 0$ in $\mathbf{C}^+$, unless $g(z) = 0$ everywhere. □

Theorem 13.8.12.

A rational function $f = u/v$ can be a positive function only if $|\deg u - \deg v| \leq 1$. If $|\deg u - \deg v| = 1$, the leading coefficients of u and v have a positive ratio.

Proof. Suppose that $f(z) \sim az^\alpha$, $z \to \infty$. Analogously to the previous proof we have $\arg f(z) = \arg a + \alpha \arg z$. If $|\alpha| > 1$ the right hand side covers an open sector of width greater than π and a positive function cannot do this. The rest is proved like the beginning of the previous proof. □

Results of a similar type can be obtained by a substitution like $z := 1/z$ or by applying the ideas of the proof. (See Problems.)

The Routh algorithm provided a clear criterion for the positiveness of an odd rational function. For more general rational functions the following result has for a long time been well known in circuit theory, where positive functions play an important role in the study of passive circuits.

Theorem 13.8.13.

A rational function $R(z)/S(z)$, where R, S have no common divisor, and where the leading terms of R and S are different from zero, is a positive function, iff

(i) $\Re\big(R(iy)/S(iy)\big) \geq 0$, *for all real y, such that $S(iy) \neq 0$;*

(ii) *$R + S$ is a Hurwitz polynomial.*

Proof. Think of a boundary locus construction $q = R(z)/S(z)$ with z traversing the imaginary axis from $-i\infty$ to $i\infty$ (instead of ζ traversing the unit circle). Condition (i) tells that $\mathbf{C}^-$ is to the left of the boundary locus, and hence the number of unstable roots of the equation $R(z) - qS(z) = 0$ is the same for all $q \in \mathbf{C}^-$. Condition (ii) tells that this number is 0, by a test at the point $q = -1$. □

For a real rational function Condition (i) can also written $\Re R(iy)S(-iy) \geq 0$. We can also formulate this

$$P(y^2) \equiv \Re R(iy)S(-iy) \geq 0, \tag{13.8.26}$$

where P is a polynomial of kth degree at most. It often simplifies the matters to use also the necessary condition that *both R and S belong to the closure of Hurwitz polynomial*, (although it should be possible to derive these from the others).

Example 13.8.5.

Consider the real function $f(z) = R(z)/S(z)$, where

$$R(z) = az + b,\ S(z) = cz + d,\ |c| + |d| > 0,\ ad - bc \neq 0.$$

R and S are in the closure of Hurwitz polynomials, hence all non-zero coefficients have the same sign, and the other conditions imply that there is at least one non-zero coefficient in both the denominator and the numerator. Condition (13.8.26)

becomes $P(y^2) = \Re(aiy+b)(-ciy+d) = acy^2+bd$, and this adds no new constraints, nor does Condition (ii). *A necessary and sufficient condition for the positiveness of a function $(az+b)/(cz+d)$, that is not identically a constant or $0/0$, is that the non-zero coefficients have the same sign.*

Next we shall find the positivity condition for the real function,

$$\frac{R(z)}{S(z)} \equiv \frac{z+a_0}{b_2z^2+b_1z+b_0}, \quad b_2 \neq 0,$$

where r, s have no common divisor. All non-zero coefficients must have the same sign. Since $a_1 = 1$, they must be non-negative and $b_2 > 0$. Condition (i) then tells that either $b_0 > 0$ or $a_0 > 0$ The function R/S is positive at the same time as S/R. We consider S/R in order to apply Theorem 13.8.11. We have

$$S(z)/R(z) = b_2 + (b_1 - b_2a_0)z + b_0)/(z+a_0).$$

By Theorem 13.8.11, and the first part of this example, we obtain the following necessary and sufficient conditions:

$$b_2 > 0, \quad b_0 + a_0 > 0, \quad b_1 - b_2a_0 \geq 0,$$

(in addition to the condition $s(-a_0) \neq 0$ for no common divisor.)

Finally we consider

$$\frac{R(z)}{S(z)} \equiv \frac{z^2+a_1z+a_0}{b_2z^2+b_1z+b_0},$$

with no common divisors. We have the non-negativity conditions for all coefficients, and Condition (i) adds the strict inequalities $b_0 + a_0 > 0$, $b_1 + a_1 > 0$. Condition (13.8.26) becomes after a short calculation, if we set $y^2 = t$, $a = a_1b_1 - a_0 - b_0$, $b = a_0b_0$,

$$P(t) = t^2 + at + b \geq 0, \quad \forall t \geq 0. \tag{13.8.27}$$

Note that $\min P(t) = P(0) = b$, if $a \geq 0$, or $b - \frac{1}{4}a^2$ if $a < 0$. Since we have the condition $b = a_0b_0 \geq 0$ already, we finally have, *in addition to the previous conditions, the new condition $b \geq a^2/4$ if $a < 0$.*

For rational functions of higher degree, a classical device, named Sturm chains, can be applied to check the validity of the inequality (13.8.26), after it has been rephrased as proving that

$$P(0) \geq 0, \quad p(t) \equiv P(t) + \epsilon \neq 0, \quad t \in (0, \infty), \tag{13.8.28}$$

where ϵ is to be chosen a little larger than what is needed to compensate for the errors of the elements of the two sequences, due to rounding.

Theorem 13.8.14. Sturm chains.

Let p be a polynomial with derivative p'. Set $p_0 = p$, $p_1 = p'$, and consider the Euclid algorithm,

$$p_{i-1} = q_i p_i - p_{i+1}, \quad i = 1, 2, \ldots, m.$$

Here q_i is the quotient at the division p_{i-1}/p_i, hence $\deg p_{i+1} < \deg p_i$. The algorithm terminates when $p_{m+1} = 0$; p_m is the greatest common divisor of the polynomials p, p'. The sequence of polynomials $p_0, p_1, \ldots, p_m$ is called a **Sturm chain**.

If $p(b) \neq 0$, $p(c) \neq 0$, then the number of distinct roots of the equation $p(t) = 0$ in the open real interval (b, c) equals the difference between the number of sign changes in the sequence $p_0(b), p_1(b), \ldots, p_m(b)$ and in the sequence $p_0(c), p_1(c), \ldots, p_m(c)$. A multiple root is counted as one root.

For $c = \infty$ the leading coefficient of the polynomial p_i is to be substituted for $p_i(c)$. Similarly for $b = -\infty$, with appropriate choice of signs.

Comment: A zero in these sequences can be ignored at the counting of sign changes. If, e.g., $p_i(b) = 0$, $i > 0$, the recurrence relation shows that there is exactly one sign change in the subsequence $p_{i-1}(b)$, $p_i(b)$, $p_{i+1}(b)$, both for $p(b) = \epsilon$ and $p(b) = -\epsilon$, unless two consecutive numbers in the sequence are zero, but in that case the whole sequence would be zero, and this is this is inconsistent with the assumption.

Proof. For a complete proof, see, e.g., Gantmacher [6, 1959], Vol.2, p.175]. □

Example 13.8.6.

The following is a classical application of Sturm chains for the separation of the real roots of an algebraic equation. Given $p(t) = 5t^5 - 3t^3 - 2t + 1$. The Sturm chain becomes (rounded to 4 decimals):

$$\begin{aligned}
p_1(t) &= p'(t) = 25t^4 - 9t^2 - 2 \\
p_2(t) &= 1.2t^3 + 1.6t - 1 \\
p_3(t) &= 42.3333t^2 - 20.8333t + 2 \\
p_4(t) &= -1.8339t + 1.0279 \\
p_5(t) &= -3.6221.
\end{aligned}$$

The number of sign changes are

	4	3	2	1	1
for $t =$	$-\infty$	0	0.5	1	∞

After counting the number of "lost sign changes" we conclude that $p(t)$ has 1 negative and 2 positive zeros.

Example 13.8.7.

Consider the situation which occurred in Example 13.8.5, i.e., we want to find conditions for $p(t) = t^2 + at + b$ to be positive for $t \geq 0$. A necessary condition

is $b \geq 0$. In an algebraic treatment, the Sturm chain becomes $p(t),\ p_1(t) = 2t + a,\ p_2(t) = a^2/4 - b$. At $t = 0$ the sign sequence is $(+,\ a,\ a^2/4 - b)$, and at $t = \infty$, it is $(+,\ +,\ a^2/4 - b)$.

If $a^2/4 - b < 0$, there is 1 sign change at both places, for all a.

If $a^2/4 - b > 0$, there is no sign change at $t = \infty$, and so is the case at $t = 0$ if $a > 0$, but if $t = 0,\ a < 0$, there are two sign changes.

We reach the same conclusion as in Example 13.8.5: $p(t) > 0$ for $t \geq 0$ iff $b > 0$, and $a > 0$ or $b > a^2/4$.

In a problem with (say) two parameters, it is not hard to work entirely with a numerical version of the Sturm algorithm, instead of a symbolic version, if the purpose is to obtain a plot of the the set $\{(a, b) : p(t; a, b) > 0 \quad \forall t > 0.\}$. It is not hard to find suitable stepsize control, interpolation techniques, etc., for a program that you can be happy with yourself in an interactive environment. You can even allow yourself to use one or two big values of t, (though not so big that you risk overflow) instead of having a special branch in your code for handling ∞. It is less trivial to design software that works well under any conditions (good step size control, good criteria for deciding whether a quantityof small absolute value is to be counted as positive or not, signals for ill-conditioned cases, etc.). It seems likely that the complexity grows more slowly with $\deg p$ with this approach, than with symbolic computation with automatic handlingof the inequality logic. (We are still discussing a two parameter problem.) This particular example seems easy both ways. This discussion is applicable to the other algorithms in this section.

There is also a criterion due to Schur 1916, that directly tests whether or not all zeros of a polynomial are located inside the unit circle, without the Greco-Roman transformation.

Theorem 13.8.15. The Schur criterion.

Let $P(\zeta) = c_0\zeta^n + c_1\zeta^{n-1} + \ldots + c_n$, *and set* $\hat{P}(\zeta) = \bar{c}_n\zeta^n + \bar{c}_1\zeta^{n-1} + \ldots + \bar{c}_n$. *We write* $P \in \mathrm{Sch}$, *iff all zeros of* P *are strictly inside the unit circle. Let* $p_0(\zeta)$ *be a* k*th degree polynomial, and define recursively,*

$$p_{i+1}(\zeta) = \frac{\hat{p}_i(0)p_i(\zeta) - p_i(0)\hat{p}_i(\zeta)}{\zeta}.$$

The algorithm terminates when $|p_i(0)| \geq |\hat{p}_i(0)|$.

The following statements are the basis of the algorithm:

(i) $(p_i \in \mathrm{Sch}) \leftrightarrow \big((p_{i+1} \in \mathrm{Sch}) \wedge (|p_i(0)| < |\hat{p}_i(0)|)\big)$,

(ii) $p_0 \in \mathrm{Sch}$, *iff* p_i *is equal to a constant at the termination.*

Proof. Sketch. If $|\hat{p}_i(0)| > |p_i(0)|$, then $\deg p_{i+1} = \deg p_i - 1$.

The statement (i) is proved by the Rouché theorem of complex analysis, that tells that $\zeta p_{i+1}(\zeta)$ and $p_i(\zeta)$ have the same number of zeros inside the unit circle,

if $|-p_i(0)\hat{p}_i(\zeta)| < |\hat{p}_i(0)p_i(\zeta)|$ on the unit circle. Since $|\hat{p}_i(\zeta)| = |p_i(\zeta)|$ on the unit circle, the latter condition is equivalent to the inequality $|\hat{p}_i(0)| > |p_i(0)|$.

If $P(\alpha) = 0$ then $\hat{P}(1/\bar{\alpha}) = 0$, and vice versa. If $\deg P > 0$, the modulus of the product of the zeros of P equals $|P(0)/\hat{P}(0)|$. Hence p_i cannot belong to Sch, if the termination criterion $|p_i(0)| \geq |\hat{p}_i(0)|$ is satisfied, before p_i has become a constant. □

Figure 13.8.3. *Mapping of an equidistant point set on the unit circle.*

The mapping of an equidistant point set on the unit circle is often very far from equidistant. The right part of Fig. 13.8.3 is a magnification generated by 53 points around $\zeta = -1$, out of the 256 points which generated the left figure. More than 20% of the points are mapped into the strip $-0.0005 < \Re q < 0.0025$. that contains less than 2% of the length of the locus. The left figure gives an impression of the nature of the singularity at $q = 0$, that contradicts the expectation given by analytic considerations. This inspired the drawing of the magnification. Similarly, it is to be expected that the algorithms described in this section may sometimes be needed as a supplement to a drawing of the boundary locus also for a single method (without parameters).

Problems

1. (a) Verify the statements made above concerning the companion matrix defined by (13.8.21), its characteristic equation, eigenvalues, eigenvectors etc.

Hint: An easy way to find the characteristic equation and the eigenvector is to solve the equation $Az = \lambda z$ from the top to the bottom. The last equation gives the condition that λ must satisfy!

(b) Similarly, find the inverse of the companion matrix A is by solving the equation $Ax = y$ from the top to the bottom. This gives you $x = A^{-1}y$.

(c) Let λ be a double eigenvalue of a companion matrix. Show that a principal vector is $(0,\ 1,\ 2\lambda, \ldots,\ (k-1)\lambda^{k-2})^T$. Formulate a generalization to eigenvalues of higher multiplicity.

(d) Rewrite analogously the differential equation $y^{(k)} + a_1 y^{(k-1)} + \ldots + a_k y = 0$.

(e) What bounds do you obtain for the roots of an algebraic equation by the consideration of the maximum norm or the subordinate logarithmic norm of its companion matrix? Consider also the l_1-norm and weighted variants of both norms, e.g. with weights $w_i = c^i$, for some suitable choice of c.

2. Does (13.8.20) remain valid, if the condition on r_n is replaced by the more liberal condition, $\|r_n\| \leq ch^{p+1}(1 + K_1 h)^{n+1}$?

3. Prove Theorems 13.8.3 and 13.8.4.

Hint: These theorems are analogous to Theorems 13.8.2 and 13.7.14, see the comment after Theorem 13.8.3. Concerning Theorem 13.8.4, note, e.g., that if $T_n = T(t_n)$, where $T(t)$ is a smooth function, then for $t = t_{n+1}$

$$\|I - h_n T^{-1} T'(t)\| \approx 1 + h_n \mu(-T^{-1} T'(t).$$

4. The slopes k_n in the knots of a cubic spline is, by Theorem 4.6.1, in the case of m subintervals of equal length h, determined by the difference equation,

$$k_{n+1} + 4k_n + k_{n-1} = d_{n+1} + d_n, \quad n = 0, 1, 2, \ldots, m,$$

where $d_n = (y_n - y_{n-1})/h$ is given. Several alternatives for providing the boundary values k_0 and k_m are described in Sec. 4.6.2. Show that, if m is not too small, the effect on k_n of the errors of the boundary slopes is approximately

$$\delta k_n \approx \delta k_0 (-2 - \sqrt{3})^{-n} + \delta k_m (-2 - \sqrt{3})^{-m+n},$$

and that we can determine δk_0 by setting $\delta k_1 \approx (-2 + \sqrt{3})\delta k_0$ into equation (4.6.11) or (4.6.12), and analogously for δk_m. For what values of m are the errors of these results less than (say) $0.01 \max(|\delta k_0|,\ |\delta k_m|)$?
For what values of n is $|\delta k_n|$ itself less than (say) this bound? (Make an appropriate assumption about m in order to get a simple result.)
Estimate also how these errors are propagated to the values of the spline itself. (See (4.6.4).)

5. (a) Show that all solutions of the difference equation

$$y_{n+1} - 2\lambda y_n + y_{n-1} = 0$$

are bounded, as $n \to \infty$, if $-1 < \lambda < 1$, while for any other λ in the complex plane there exists at least one solution which is unbounded.

(b) Let A be a diagonalizable matrix. Give, in terms of the eigenvalues of A, a necessary and sufficient condition for the boundedness as $n \to \infty$ of all solutions of the difference equation

$$y_{n+1} - 2Ay_n + y_{n-1} = 0$$

6. (a), (b) (c) See Problems 9 and 10 of Sec. 8.5 in the old Dahlquist-Björck (unfinished).

7. Difference equation for power series coefficients of a rational function, and the partial fraction decomposition. Relate to Padé, epsilon and Shanks.(unfinished)

8. A short introduction to the z-transform (unfinished).

9. (a) Assume that the matrices A_n are non-singular, and that all solutions of the linear system $u_{n+1} = A_n u_n$ satisfy the inequality

$$\|u_n\| \le k_0 \|u_j\|, \quad \forall j, \quad 0 \le j \le n.$$

Set $P_n = A_{n-1} A_{n-2} \dots A_1$, see (13.8.16) and Problem 13 of Sec. 13.1. Show that $\|P_n P_j^{-1}\| \le k_0$.

(b) Assume that $B_n(u)$ is a matrix such that

$$\|B_n(u)\| \le c_1, \ \forall n \ge 0 \ \text{ if } \|u_n\| \le c_2(n).$$

Show that all solutions of the pseudo-linear system $u_{n+1} = \big(A_n + B_n(u)\big)u_n$ satisfy the inequality, $\|u_n\| \le c_0\|u_0\| + c_0 \sum_{j=1}^{n} c_1 \|u_{j-1}\|$, as long as this inequality implies that $\|u_n\| \le c_2(n)$.

(c) A difference analog of the Gronwall-Bellman Lemma: Let $\{g_n\}$, $\{k_n\}$, be two scalar sequences, such that $k_n \ge 0$, $\forall n \ge 0$, and set

$$K_j = (1 + k_{j-1})(1 + k_{j-2}) \cdots (1 + k_0).$$

Assume that a (scalar) sequence $\{y_n\}$ for $n \ge 0$ satisfies the inequality $y_n \le g_n + \sum_{j=1}^{n} k_{j-1} y_{j-1}$. Show that

$$y_n \le g_n + \sum_{j=1}^{n} (K_n / K_j) k_{j-1} g_{j-1},$$

Show also that if $k_n = k$ and $g_n = g$ for all $n > 0$, then $y_n \le (1+k)^n g$. Apply the result to find a simple bound for $\|u_n\|$ in Problem (a).

Hint and note: See the hint and note of Problem 13 of Sec. 13.1.

10. (a) Fig. 13.8.4a shows the boundary locus for a consistent linear 8-step method, about which it is known that $\sigma(\zeta)$ has one unstable root. Find the stability region, if it exists. Is the method zero-stable?

(b) Fig. 13.8.4b shows the boundary locus of the linear 3-step method generated by the polynomials, $\rho(\zeta) = (\zeta - 1)(\zeta^2 + 1)$, $\sigma(\zeta) = 2\zeta^3$. Is the method consistent? Is it zero-stable? Is it strongly zero-stable? Is it ∞-stable?

(c) Show that the explicit linear 2-step method,
$y_{n+2} + 4y_{n+1} - 5y_n = h\big(4f(y_{n+1}) + 2f(y_n)\big)$ is linearly consistent of order $\bar{p} = 3$ but, unfortunately, it has no stability region. Would you recommend it to your best friend?

Figure 13.8.4. *Boundary locus for two methods.*

11. Set

$$r(z) = \sum_{i=0}^{k} a_i z^i, \qquad s(z) = \sum_{i=0}^{k} b_i z^i, \tag{13.8.29}$$

A linear k-step method can also be expressed in the following form, where $\nabla = 1 - E^{-1}$ is the backward difference operator:

$$r(\nabla) y_{n+1} = h s(\nabla) f(y_{n+1}).$$

For a one-leg method the corresponding equation reads

$$r(\nabla) y_n = h f\big(s(\nabla) y_n\big).$$

(a) Prove the formula $\rho(\zeta) = \zeta^{k-1} r(1 - \zeta^{-1})$, and the analogous formula for the polynomials σ and s.

(b) Show how to plot the boundary locus, when the coefficients of r and s are given, without calculating the coefficients of ρ and σ.

(c) The most widely used methods for stiff problems are probably the BDF methods, based on the truncated expansion of the differentiation operator given in section 4.5.3: $hD = \ln E = -\ln(1 - \nabla)$.

$$r(\nabla) = \nabla + \tfrac{1}{2}\nabla^2 + \frac{1}{3}\nabla^3 + \ldots + \frac{1}{k}\nabla^k, \quad s(\nabla) = 1.$$

Plot the stability regions of these methods for $1 \leq k \leq 7$. Give a (non-rigorous) answer to the following questions by a look at the plots. For which values of k is the method zero-stable? For which values is it ∞-stable? For which values is it A-stable?

(d) Show that $\nabla^j = (1 - E^{-1})^j$, $E^{-j} = (1 - \nabla)^j$. Set $\tilde{\nabla}_k = (\nabla^k, \nabla^{k-1}, \ldots, \nabla, 1)$,

$\tilde{E}_k = (E^{-k},\ E^{-k+1}, \ldots,\ E^{-1}, 1)$. Determine a matrix P_k such that $\tilde{\nabla}_k = \tilde{E}_k P_k$, and show that $P_k^{-1} = P_k$. (The latter can be shown with almost no computation.) Find a recurrence formula for the construction of the matrix $P_k,\ k = 1, 2, 3, \ldots$

Set $\tilde{a} = (a_k,\ a_{k-1}, \ldots, a_0)^T$, $\tilde{\alpha} = (\alpha_0,\ \alpha_1, \ldots,\ \alpha_k)$, and show that $\tilde{\nabla}_k \cdot \tilde{a} = \tilde{E}_k \cdot \tilde{\alpha}$ and $P_k \tilde{a} = \alpha, \quad P_k \tilde{\alpha} = a$. ($P$ stands for Pascal.)

(e) Compute $\rho(\zeta),\ \sigma(\zeta)$, for the BDF method, for $k \leq 3$.

12. Some multistep plots with exceptions: 2-step methods with degenerate S, or rotating twice (unfinished).

13. Some uncomplicated Runge-Kutta plots and linear consistency analysys: Heun2 the same as Runge 2 on linear problems; Heun3 acc. to HNW p. 133 (unfinished).

14. Runge-Kutta plots with complications: several level curves for Kutta-Simpson; the boundary locus not unicursal. (unfinished.)

15. (a) With the notations of the description of the Routh algorithm in Sec. 13.8.3, show that, *if* $c_0 > 0$, then the first conditions that come out of the Routh algorithm, applied algebraically, are

$$c_1 > 0,\ c_1 c_2 - c_0 c_3 > 0,\ c_3(c_1 c_2 - c_0 c_3) + c_1(c_0 c_5 - c_1 c_4) > 0, \ldots,$$

and show that $c_j > 0$, for $1 \leq j \leq 3$ are consequences, if we set $c_j = 0$ for $j > 3$.

(b) Test by the Routh algorithm the suspicion that the polynomial $z^5 + z^4 + mz^3 + z^2 + nz + 1$ cannot be a Hurwitz polynomial for any values of the parameters $m,\ n$.

Hint: When one works algebraically it often simplifies to divide the denominator or the numerator by a factor that has to be positive, if the rational function is a positive function. Moreover, the algebra can be simplified by appropriate substitutions.

16. (a) Hurwitz showed in 1895 the criterion for a kth degree to be a "Hurwitz polynomial" is that the first k principal minors of the matrix H with elements $h_{ij} = c_{2j-i},\ i, j = 1, 2, \ldots k$, should be positive. Here the c_ν are the polynomial coefficients; the notation is the same as in Problem 15 and Theorem 13.8.10, with the conventions that $c_0 > 0$, and $c_\nu = 0$ for $\nu < 0$ and $\nu > k$. Note that each row contains the coefficients of either $p_0(z)$ or $p_1(z)$. Write down the matrix for $k = 5$.

Interpret *the Routh algorithm as an elimination process* for bringing the matrix H to upper triangular form.

(b) Is it true that the condition $c_4 > 0$ together with the three inequalities mentioned in Problem 15 (a), are sufficient for the Hurwitz property, if $k = 4$?

(c) Is it true, that it does not matter for the application of the Hurwitz criterion, if the coefficients are ordered as in our description, with c_0 as the leading coefficient, or the other way around, with c_k as the leading coefficient? Is the analogous thing true for the Schur algorithm?

17. (a) Write a program for the Routh algorithm for a polynomial with numer-

ically given coefficients, as described in Sec. 13.8.3, and test it on suitable polynomials, including cases where there are roots on the imaginary axis.

(b) Write a program for the Schur algorithm, as described in Sec. 13.8.3, and test it on suitable polynomials, including cases where there are roots on the unit circle.

(c) Write a program for the Sturm chain, as described in Sec. 13.8.3, and test it on suitable polynomials. Test also the sensitivity to perturbations of the coefficients.

18. (a) Convince yourself about the validity of the relations in (13.8.22) and (13.8.23) which are not definitions.

(b) The following is a short MATLAB program for the *Greco-Roman transformation*. It computes by recurrence relation the $k \times k$ matrix called `gr` which maps the coefficient vector of $R(z)$ to the coefficient vector of $\rho(\zeta)$. The vectors are column vectors with the leading coefficient last.

```
gr = 1; a1 = 1;
for n = 1 : k,
    bg = [0; a1] − [a1; 0]; z = zeros(1, n);
    c = [gr; z] + [z; gr]; a1 = bg; gr = [bg, c];
end;
```

Read the program, and make sure you understand the algorithm. A semicolon (inside brackets) means partitioning in the vertical direction, while a comma means means partitioning in the horisontal direction. Note that there is dynamic memory allocation, e.g., `a1` is a column, the length of which increases from 1 to k. Is it true that the inverse of `gr` equals 2^{-k}`gr`
Note that the algorithm actually computes all transformation matrices for the orders $n = 1, 2, \ldots, k$. Test it on some simple cases, either in MATLAB or after translation to another language.

References

[1] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods.* John Wiley, New York, NY, 1986.

[2] E. A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations.* McGraw-Hill, New York, 1955.

[3] L. Collatz. *The Numerical Treatment of Differential Equations. 3rd ed.* Springer-Verlag, Berlin, 1960.

[4] C. A. Desoer and H. Haneda. The measure of a matrix as a tool to analyze computer algorithms for circuit analysis. *IEEE Trans.*, CT–19:480–486, 1972.

[5] A. Dinghas. *Vorlesungen über Funktionentheorie.* Springer, Berlin, 1961.

[6] F. R. Gantmacher. *The Theory of Matrices. Vols. I and II.* Chelsea Publishing Co., New York, NY, 1959.

[7] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems. 2nd. ed.* Springer-Verlag, Berlin, 1993.

[8] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential Algebraic Problems.* Springer-Verlag, Berlin, 1991.

[9] Klamkin, 1987

[10] R. D. Richtmyer and K. W. Morton. *Difference Methods for Initial Value Problems. 2nd. ed.* John Wiley-Interscience, New York, 1967.

Chapter 14

Partial Differential and Integral Equations

14.1. Introduction

Computers have made it possible to treat many of the partial differential equations of physics, chemistry, and technology on a much larger scale than previously. The most common methods are **difference methods**, i.e., derivatives with respect to one or more of the variables are approximated by difference quotients. Other important methods, namely, the Galerkin method, the Ritz method and the **finite element method** (FEM) will be presented in Secs. 14.3.1 and 14.3.2.

Numerical weather prediction is one of the most well-known applications of partial differential equations. Here, a model of flow in the atmosphere is described by a system of partial differential equations, with two or three space variables and a time variable. The most recently available observational data from a large area of the earth is the initial condition. On a large computer,it takes a few hours to compute the subsequent flow in the atmosphere for, say, three days ahead in time. The models of the atmosphere which are used are in many ways less complete than one would like them to be.

Partial differential equations with four independent variables x, y, z, t can be time-consuming to handle also on the largest available computers. Fortunately, one can often reduce the number of dimensions by making use of *symmetry properties* of the problem, or by so-called *separation of variables*. Problems with two independent variables are usually tractable.

For the most part, this chapter is only intended to give some insight into the possibilities that exist and some feeling for the difficulties with numerical stability which can occur. Some simple, general ideas are illustrated in the examples—these examples could be treated by more effective methods.

We start by developing some formulas for multidimensional interpolation. and differentiation. These will be used to derive simple methods for the numerical

solution of partial differential and integral equations.

14.1.1. Multidimensional Interpolation The simplest way to generalize interpolation to functions of several variables is to use repeated one-dimensional interpolation, i.e., to work with one variable at a time. The following formula for **bilinear interpolation**,

$$f(x_0+ph, y_0+qh) \approx (1-q)\phi(y_0) + q\phi(y_0+k),$$
$$\phi(y) = (1-p)f(x_0, y) + pf(x_0+h, y).$$

is the simplest example. After simplification it can written as

$$f(x_0+ph, y_0+qh) \approx (1-p)(1-q)f_{0,0} + p(1-q)f_{1,0} + (1-p)qf_{0,1} + pqf_{1,1}, \tag{14.1.1}$$

where we have used the notation $f_{ij} = f(x_0+ih, y_0+jk)$, $i,j \in \{0,1\}$. This formula is exact for functions of the form $f(x,y) = a+bx+cy+dxy$, and from equation 4.2.7 we obtain the error bound,

$$\max_{(x,y)\in R} \frac{1}{2}(p(1-p)h^2|f_{xx}| + q(1-q)h^2|f_{yy}|), \qquad 0 \le p,q \le 1,$$

where $R = \{(x,y) : x_0 \le x \le x_0+h, y_0 \le y \le y_0+k\}$. The formula for bilinear interpolation can easily be generalized by using higher order interpolation in the x and/or y direction.

In the following we consider explicitly only the case of two dimension, since corresponding formulas for three and more dimensions are analogous.

A **rectangular grid** in the (x,y)-plane with grid spacings h,k in the x and y directions, respectively, consists of points $x_i = x_0+ih$, $y_i = y_0+ik$. In the following we use the notation $f(x_i,y_j) = f_{ij}$. (For interpolation formulas valid for irregular triangular grids, see Sec. 8.4.3)

Central difference approximations for partial derivatives using function values can be obtained by working with one variable at a time,

$$\frac{\partial f}{\partial x} = \frac{1}{2h}(f_{i+1,j} - f_{i-1,j}) + O(h^2), \quad \frac{\partial f}{\partial y} = \frac{1}{2k}(f_{i,j+1} - f_{i,j-1}) + O(k^2).$$

For second order derivatives

$$\frac{\partial^2 f}{\partial x^2} = \frac{1}{h^2}(f_{i+1,j} - 2f_{ij} + f_{i-1,j}),$$

and a similar formula holds for $\partial^2 f/\partial y^2$.

Formulas of higher accuracy can also be obtained by operator techniques, based on an operator formulation of Taylor's expansion (see Theorem 4.6.6,

$$f(x_0+h, y_0+k) = \exp\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right) f(x_0, y_0) \tag{14.1.2}$$

From this we obtain

$$f(x_0+h,y_0+k) = f_{0,0} + \Big(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\Big)f_{0,0} + \Big(h^2\frac{\partial^2}{\partial x^2} + 2hk\frac{\partial^2}{\partial x\partial y} + k^2\frac{\partial^2}{\partial y^2}\Big)f_{0,0} + O(h^2+k^2).$$

An interpolation formula valid for all quadratic functions can be obtained by replacing in Taylor's formula the derivatives by difference approximations valid for quadratic polynomials,

$$\begin{aligned} f(x_0+ph,y_0+qh) \approx f_{0,0} &+ \frac{1}{2}p(f_{1,0}-f_{-1,0}) + \frac{1}{2}q(f_{0,1}-f_{0,-1}) \\ &+ \frac{1}{2}p^2(f_{1,0}-2f_{0,0}+f_{-1,0}) \\ &+ \frac{1}{4}pq(f_{1,1}-f_{1,-1}-f_{-1,1}+f_{-1,-1}) \\ &+ \frac{1}{2}q^2(f_{0,1}-2f_{0,0}+f_{0,-1}). \end{aligned} \tag{14.1.3}$$

This formula uses function values in nine points. (The proof of the expression for approximating the mixed derivative $\dfrac{\partial^2}{\partial x\partial y}f_{0,0}$ is left as an exercise, Problem 2.

Cubic Hermite interpolation treating one variable at a time (see Example 4.6.3):

$$\begin{aligned} f(x_0+ph,y_0+qh) &= (1-q)\phi(y_0) + q\phi(y_1) \\ &+q(1-q)\Big[(1-q)(h\phi_y'(y_0)-\Delta\phi(y_0)) - q(h\phi_y'(y_1)-\Delta\phi(y_0))\Big]. \end{aligned}$$

$$\begin{aligned} \phi(y) &= (1-p)f(x_0,y) + pf(x_1,y) \\ &+p(1-p)\Big[(1-p)(hf_x'(x_0,y)-\Delta f(x_0,y)) - p(hf_x'(x_1,y)-\Delta f(x_0,y))\Big]. \end{aligned}$$

This formula requires that the quantities $f, \partial f/\partial x, \partial f/\partial y$, and $\partial^2 f/\partial x\partial y$ are given at the four points (x_i,y_j) for $0 \le i,j \le 1$. Hence 16 quantities are needed to specify the bicubic polynomial.

REVIEW QUESTIONS

1. How is bilinear interpolation performed? What is the order of accuracy?

PROBLEMS

1. Compute by bilinear interpolation $f(0.5, 0.25)$ when

$$f(0,0) = 1, \quad f(1,0) = 2, \quad f(0,1) = 3, \quad f(1,1) = 5.$$

2. Derive a formula for $f''_{xy}(0,0)$ using f_{ij}, $|i| \leq 1, |j| \leq 1$, which is exact for all quadratic functions.

14.2. Partial Differential Equations

14.2.1. Initial Value Problems In many applications, one wants to study some process in both time and space. In physics, a partial differential equationis often derived by first using laws of nature to derive approximate *difference* equations involving relevant physical quantities and small increments in time and space, and then passing to the limit to get partial differential equations. Construction of a numerical method often involves going "backward", i.e., constructing difference equations from the differential equation. However, the art of the numerical analyst is to construct the difference scheme so that it not only is *consistent* with the differential equation, but is also *accurate* and *stable*. These concepts have already been discussed e.g., in Secs. 13.2 and 13.4.

We shall here give a simple example, which nevertheless is fairly typical: the problem is to compute the variation of temperature in a homogeneous wall, where initially the temperature is higher on one side than on the other. The curves which we shall compute, i.e., the temperatures in the wall at various times, will look approximately like those shown in Fig. 14.2.1. Here $x = 0$ and $x = 2$ corresponds to the two sides of the wall.

FIG. 14.2.1. *Temperature curves for a homogeneous wall.*

We divide the wall into $N - 1$ layers of thickness $h = 2/N$, and denote the midpoints of the layers by $x_i = ih$, see Fig. 14.2.1.

We want to compute the temperature at equidistant times $t_j = jk$, where k is the time step. We denote the temperature by $U_{ij} = u(x_i, t_j)$. The first physical law we shall use is that which says that the flow of heat past $x = \alpha$ is equal to a constant time κ times $\partial u/\partial x$ at $x = \alpha$. The constant κ is called the heat

Fig. 14.2.2. *Flow of heat.*

conductivity. Applying this to the layer with center x_i, the difference between the amount flowing in at the left and that flowing out at the right is

$$-\kappa \frac{\partial u}{\partial x}\Big|_{x=x_{i-1/2}} + \kappa \frac{\partial u}{\partial x}\Big|_{x=x_{i+1/2}}.$$

Approximating these quantities by difference quotients, we get that the amount of heat flowing into the ith layer during the time $k = t_{j+1} - t_j$ is

$$\begin{aligned} W &= k\kappa\Big(-\frac{u_{ij} - u_{i-1,j}}{h} + \frac{u_{i+1,j} - u_{i,j}}{h}\Big) \\ &= \frac{k}{h}\kappa(u_{i+1,j} - 2u_{ij} + u_{i-1,j}). \end{aligned} \tag{14.2.1}$$

Another physical law tells us that the amount of heat which flows into the layer with midpoint x_i is approximately proportional to the change in temperature at the midpoint of that layer. The amount of heat W which comes in is thus $(u_{i,j+1} - u_{ij})$ times a constant, which really consists of three constants—the specific heat c, the density ρ, and the thickness of the layer, h:

$$W = (u_{i,j+1} - u_{ij})c\rho h. \tag{14.2.2}$$

Equations (14.2.1) and (14.2.2), we get

$$u_{i,j+1} - u_{ij} = \frac{\kappa}{c\rho}\frac{k}{h^2}(u_{i+1,j} - 2u_{ij} + u_{i-1,j}). \tag{14.2.3}$$

From this difference equation, approximate temperatures at time t_{j+1} can be computed if the temperature curve at time t_j is known, and if the temperature at the boundaries $x = 0$ and $x = 2$ is prescribed for all time t (**boundary conditions**). Also, dividing by k and letting $h, k \to 0$, (and applying (1.2.5)) we get a partial differential equation, known as the **heat equation**,

$$\frac{\partial u}{\partial t} = \frac{\kappa}{c\rho}\frac{\partial^2 u}{\partial x^2}. \tag{14.2.4}$$

For simplicity in what follows, we assume that $\kappa = c\rho$.

Boundary conditions can be prescribed in several ways, depending on the physical situation to be simulated. Here we shall assume, as an example, that

the temperature at $x = 0$ is given for all time t by $u(0,t) = 1,000\sin(8\pi t/3)$, and also that at $x = 2$ we have perfect isolation, i.e., no heat flow. Mathematically, we have then $(\partial u/\partial x)(2,t) = 0$. We also assume the **initial conditions** $u(x,0) = 1,000\sin(\pi x/4)$.

From the first boundary condition we get

$$u_{0,j} = 1,000\sin(8\pi t_j/3).$$

One way to formulate the boundary condition at $x = 2$ is to introduce a new point x_{N+1}; then

$$0 = \frac{\partial u}{\partial x}(2,t_j) \approx \frac{u_{N+1,j} - u_{N-1,j}}{2h},$$

or $u_{N+1,j} = u_{N-1,j}$, for all j. Since $u_{N-1,0}$ is known and $u_{N+1,0} = u_{N-1,0}$, it is clear that we can now compute $u_{i,j}$ for all i,j using (14.2.3). If we choose $h = 1/4$ and $k/h^2 = 1/2$ (i.e., $k = 1/32$), then we get the simple algorithm

$$u_{i,j+1} = \frac{1}{2}(u_{i-1,j} + u_{i+1,j}). \tag{14.2.5}$$

This algorithm can be pictured by the computational molecule shown in Fig. 14.2.1.

FIG. 14.2.3. *Computational molecule*

With the above values of k and h, the initial and boundary conditions become:

$$u_{i,0} = 1,000\sin\left(\frac{\pi i}{16}\right), \qquad u_{0,j} = 1,000\sin\left(\frac{\pi j}{12}\right), \qquad u_{9,j} = u_{7,j}.$$

The computations can be arranged in a scheme as below; the reader is recommended to complete it.

$j \setminus i$	0	1	2	3	4	5	6	7	8	9
0	000	195	383	555	707	832	924	981	1000	981
1	258	192	375	545	694	816	906	962	981	962
2	500	316	368	534	680	800	889	944	962	944
3	707	434	425							
4	866									

One question which is especially important in the numerical solution of partial differential equationsis: how is a disturbance propagated? To answer this, we

can make an **experimental perturbation calculation**. Since the difference equation is linear, the difference between the perturbed and the unperturbed solution can also be computed by equation (14.2.5) with homogeneous boundary conditions. Thus a disturbance (perturbation) of magnitude 1 in the computation of $u_{3,1}$ is propagated according to the following scheme:

$j \setminus i$	0	1	2	3	4	5	6	7	8	9
1	0	0	0	1	0	0	0	0	0	0
2	0	0	0.5	0	0.5	0	0	0	0	0
3	0	0.25	0	0.5	0	0.25	0	0	0	0
4	0	0	0.375	0	0.375	0	0.125	0	0	0
5	0	0.1875	0	0.375	0	0.25	0	0.0625	0	0.0625

The effect of the disturbances is spreading out and decreases. Notice the slight asymmetry due to the influence of the boundary values.

We can apply some of the ideas about stability and methods of ordinary differential equations(see Chapter 13) by approximating (14.2.4) by a system of *ordinary* differential equations. Again, put $h = 2/N$, $x_i = ih$, and let $u_i(t)$ denote an approximation to $u(x_i, t)$. Approximate $\partial^2 u/\partial x^2$ by a central difference as previously, but retain the time derivative. This procedure is sometimes called the **method of lines**. Then one gets a system of ordinary differential equations:

$$\frac{du_i}{dt} = \frac{1}{h^2}(u_{i+1} - 2u_i + u_{i-1}), \qquad i = 1, 2, \ldots, N. \tag{14.2.6}$$

The initial and boundary value conditions give

$$u_0(t) = 1,000 \sin \frac{8\pi t}{3}, \qquad u_{N+1}(t) = u_{N-1}(t).$$

these equations can be written in vector form

$$\frac{du}{dt} = Au + f(t), \tag{14.2.7}$$

where

$$A = h^{-2}\begin{pmatrix} -2 & 1 & 0 & \ldots & 0 & 0 \\ 1 & -2 & 1 & \ldots & 0 & 0 \\ 0 & 1 & -2 & \ldots & 0 & 0 \\ & \vdots & & & & \\ 0 & 1 & -2 & \ldots & -2 & 1 \\ 0 & 0 & 0 & \ldots & 2 & -2 \end{pmatrix}, \quad f(t) = h^{-2}\begin{pmatrix} u_0(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

When h is small, this is a stiff system. By Gershgorin's theorem (Theorem 8.3.1) the eigenvalues are located in the disk:

$$\left|\lambda + \frac{2}{h^2}\right| \leq \frac{2}{h^2}. \tag{14.2.8}$$

We conclude that the eigenvalues have non-positive real parts and that the spectral radius is less than $4/h^2$. In fact, the eigenvalues are all real and negative, and one of them is close to $-4/h^2$.

Euler's method applied to this system gives exactly (14.2.3) (with $\kappa/(c\rho) = 1$). It can be shown, see Problem 5, that the computations are stable when $4k/h^2 \leq 2$, i.e., if $k/h^2 \leq 1/2$. A condition of this type was first obtained by Courant, Friedrich and Lewy 1928 in a pioneering investigation of the *convergence* of the solutions of partial differential equations, as the grid spacings h, k tend to zero. If one chooses $k/h^2 > 1/2$, then with time the approximate solution becomes a more and more ragged function of x, which has very little in common with the solution of the differential equation.

The trapezoidal method applied to the above system is called the **Crank-Nicolson method**. Notice that the matrix $-A$ is tridiagonal, and positive definite so Gaussian elimination can be used without pivoting to solve the resulting linear equation. For a nonlinear partial differential equation an iterative solution is not burdening. This method is more accurate than Euler's method, and it follows from Example 13.3.4 that it is stable for all positive values of k and h. Hence, it is possible to take longer time steps than with Euler's method, but there may appear slowly decreasing oscillations in the computed values which require smoothing.

FIG. 14.2.4. *Computational molecule for the explicit midpoint method.*

One might think that it would be better to use the explicit midpoint method, which here gives

$$u_{i,j+1} - u_{i,j-1} = \frac{2k}{h^2}(u_{i+1,j} - 2u_{ij} + u_{i-1,j}).$$

For $k/h^2 = 1/2$ we get the computational molecule of Fig. 14.2.1. The "weak instability" mentioned in Sec. 13.6.2 here leads to catastrophe. (Not even the oscillation damping modification of the midpoint method would help here.)

$j \setminus i$	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0
2	0	0	1	-2	1	0	0	0	0	0
3	0	1	-4	7	-4	1	0	0	0	0
4	0	-6	17	-24	17	-6	1	0	0	0
5	0	30	68	89	-68	31	8	1	0	1
5				-338						

If there is a risk of numerical instability with a given method, it can usually be discovered by a perturbational calculation of this type.

The explicit midpoint method is useless in this problem because its stability region is only a segment of the real axis, while the matrix A has negative eigenvalues of large magnitude. The latter is typical of **parabolic problems**, one example of which is the heat equation. Nevertheless, the explicit midpoint method, or the **leap-frog** method as it is usually called in this context, has found important applications to partial differential equations, for solving **hyperbolic systems**. There are many problems of this type where the eigenvalues of the corresponding matrix A are all imaginary.

A systematic treatment of the design and analysis of stabler and accurate difference methods would require mathematical tools which are beyond the scope of this book. We refer the reader to the literature mentioned in Sec. 15.7. We hope, however that the reader who encounters an initial value problem for a more difficult partial differential equationwill derive some guidance from the points of view which we have applied above. See also the examples of nonlinear ordinary differential equationsin Chapter 13.

14.2.2. Boundary Value Problems One important equation of physics is **Poisson's equation**

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \tag{14.2.9}$$

often written as $\nabla^2 u = f(x, y)$. (Note that ∇ here does *not* mean backward difference operator here, but stands for the gradient operator $(\partial/\partial x, \partial\partial y)$; ∇^2 is called the Laplacean operator.) Poisson's equation, e.g., governs, e.g., the electric or magnetic field when the charge density or pole strength are non-zero. Poisson's equation and the special case when $f \equiv 0$, **Laplace's equation**, are the best known examples of **elliptic** partial differential equations.

To get a solution to (14.2.9) we need boundary conditions. Usually u or $\partial u/\partial n$, or some linear combination of u and $\partial u/\partial n$, are prescribed on the boundary of some closed region D. (Here $\partial u/\partial n$ means the derivative of u taken in the direction of the normal to the boundary of D.) The solution is to be obtained for the points in the interior (or exterior) of D. (Initial value problems for elliptic equations are extremely ill-conditioned, see Problem 11.)

Consider a square grid with mesh width $\Delta x = h$, $\Delta y = k$. The simplest difference approximation of $\nabla^2 u$ is the **five-point** operator ∇_5^2,

$$\nabla_5^2 u_{ij} \equiv \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{k^2}. \tag{14.2.10}$$

Fig. 14.2.2 gives the computational molecule for $h^2\nabla_5^2$. From Taylor's formula it follows, if $h = k$, that

$$\nabla_5^2 u_{ij} = \nabla u(x_i, y_j) + \frac{1}{12}\Big(\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4}\Big) + O(h^4).$$

FIG. 14.2.5. *The operator for* $h^2\nabla_5^2$.

Another difference operator, denoted ∇_x^2, see Figure 13.1.4, is

$$\nabla_x^2 u_{ij} \equiv \frac{u_{i+1,j+1} + u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} - 4u_{ij}}{2h^2}.$$

Suppose that the boundary consists of lines that are either parallel to the coordinate axes or form 45-degree angles with them, as in Fig. 14.2.2. (Curved boundaries are treated in the special literature, see, e.g., Smith [13].) If ∇^2 is approximated by ∇_5^2, then one gets the difference equation

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij} = h^2 f_{ij}. \tag{14.2.11}$$

Since the difference equation is based on approximation of the derivatives with a local truncation error which is $O(h^2)$ if $u \in C^4$, then one can expect that the truncation error in the solution is $O(h^2)$ and that Richardson extrapolation should be applicable. However, this depends in general on the given boundary conditions.

Enumerate the points in the net as in Fig. 14.2.2, and form a vector u of the desired function values. The pair of integers (i, j) is replaced by the points number. The difference equation can then be written in the form

$$Au = h^2 f, \qquad u = (u_1, u_2, \ldots, u_n).$$

For simplicity, we shall limit the discussion to the so-called **Dirichlet** problem, where the values of the function u are prescribed on the boundary. Then the following hold:

Fig. 14.2.6. *Ordering of mesh points in Poisson's equation.*

1. The vector f consists partly of values f_{ij} and partly of values of u on the boundary.
2. No point has more than four neighbors. Hence A will have at most five elements per row. All diagonal elements in $A = (a_{pq})$ are equal to -4.
3. If point p is a neighbor of point q then $a_{pq} = 1$, otherwise $a_{pq} = 0$. From this it follows that $a_{pq} = a_{qp}$, i.e., the matrix A is symmetric.

Fig. 14.2.7. *Structure of matrix A.*

A typical structure of A is shown in Fig. 14.2.2, where the lines parallel to the main diagonal indicate the location of the nonzero elements. (cf also Example 11.1.1.) Clearly, A is a band matrix. In the particular example shown in Fig. 14.2.2 with 28 points, $a_{pq} = 0$ for $|p-q| > 6$. If h is halved, then the band width is approximately doubled, while the order of the matrix is approximatively quadrupled. When the number of points is not to large, Gaussian elimination can be used; see Sec. 6.4.5. By using Richardson extrapolation (if applicable) one can often get sufficient accuracy with a reasonable small number of points.

In many cases, however, h must be chosen so small that Gaussian elimination is no longer practicable. In such cases iterative methods or the conjugate gradient method are more attractive to use; see Sec. 11.2. A particularly simple method to solve equation (14.2.11) is the SOR method described in Sec. 11.1.3, which means that the formula

$$u_{ij}^{(n+1)} = u_{ij}^{(n)} + \omega\frac{1}{4}(u_{i+1,j}^{(n)} + u_{i-1,j}^{(n)} + u_{i,j+1}^{(n)} + u_{i,j-1}^{(n)} - 4u_{ij}^{(n)} - h^2 f_{ij})$$

is used, where the points (x_i, y_j) are traversed in the same way as the numbering in Fig. 14.2.2. The process converges if $0 < \omega < 2$. For the case shown in the

figure, $\omega \approx 1.4$ would give reasonably fast convergence. When there are more grid points, then ω should be chosen closer to 2, see Theorem 11.1.8. In the literature, certain rules are given for the ordering of the points and the choice of ω. In more difficult problems, one must experiment to find the best value, though this can be done in a systematic way, see Young [15].

A good strategy for boundary value problems is often to start with a coarse grid and use Gaussian elimination. If the accuracy is insufficient then one can use the result thereby obtained—possibly with extrapolation— to construct a good initial approximation for an iterative method with a finer grid. In constructing this initial approximation one also needs values of u at the midpoints of those squares at whose corners u has been previously computed. To accomplish this, one can use the operator ∇_x^2; see Fig. 14.2.2 and Problem 5 at the end of this section.

FIG. 14.2.8. *The operators* $2h^2\nabla_x^2$ *and* $6h^2\nabla_9^2$.

By a linear combination we get the **nine-point** difference operator

$$\nabla_9^2 = \frac{2}{3}\nabla_5^2 + \frac{1}{3}\nabla_x^2. \tag{14.2.12}$$

Verify that the computational molecule is as shown in Fig. 14.2.2. This formula is advantageous in many situations because of the special form of the remainder term

$$\nabla_9^2 u_{ij} = \nabla^2 u + \frac{1}{12}h^2\nabla^4 u + O(h^4), \tag{14.2.13}$$

so for Laplace's equation this has fourth order accuracy. Also for Poisson's equation ∇_9^2 is more advantageous than ∇_5, since we have

$$\nabla^2 u = \nabla^2 u + \frac{h^2}{12}\nabla^4 u + O(h^4) = f + \frac{h^2}{12}\nabla^2 f + O(h^4)$$

Thus if one uses the difference equation

$$\nabla_9^2 u_{ij} = f_{ij} + \frac{h^2}{12}\nabla_9^2 f_{ij} \tag{14.2.14}$$

then one gets a truncation error which is $O(h^4)$ under certain smoothness conditions on u and f. With this operator, however, the SOR method must be slightly modified.

One possible strategy for boundary-value problems with linear differential equations is to start with a *coarse* grid, and use Gaussian elimination. Is the accuracy is unsufficient, then one can use the results thereby obtained—possibly with extrapolation— to construct a good *initial approximation* for an iterative method with a finer grid. To construct this one needs values of u in the midpoints of those squares at whose corners u has previously been computed. To accomplish this, one can use the operator ∇_x^2 given above.

The methodology described here is also useful in more complicated elliptic differential equations. There are more effective iterative methods, e.g., the conjugate gradient method with convergence acceleration. For Poisson's equation Fourier methods can be used. For equidistant and constant coefficients methods based on the Fast Fourier Transform (FFT) are very efficient.

REVIEW QUESTIONS

1. Give a general description of a difference approximation for:
(a) the heat equation; (b) Poisson's equation.

PROBLEMS

1. (a) Set up a difference equation for the same problem as in Sec. 14.2.1, but take $k/h^2 = 1$.

(b) How is a disturbance of unit magnitude at the point $i = 3$, $j = 0$, propagated? How large is the effect for $i = j = 4$?

(c) What conclusion can one draw concerning the use of this value of k/h^2?

2. Set up a difference approximation for the parabolic problem

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\Big((1+x^2)\frac{\partial u}{\partial x}\Big), \qquad -1 \le x \le 1, \quad t \ge 0,$$

with initial conditions and boundary values given by

$$u(x,0) = 1,000 - |1,000x|, \qquad u(-1,t) = u(1,t) = 0, \quad t \ge 0.$$

Take $h = 0.4$, $k = 0.04$ and integrate to $t = 0.12$. Take advantage of the symmetry!
Hint: An $O(h^2)$ approximation to the difference operator is obtained by

$$\frac{\partial}{\partial x}\Big(g(x)\frac{\partial u}{\partial x}\Big)_{ij} \approx \frac{1}{h^2}\big(g_{i,j+1/2}(u_{i,j+1} - u_{i,j}) - g_{i,j-1/2}(u_{i,j} - u_{i,j-1})\big).$$

3. (a) Consider the linear system of $N-1$ ordinary differential equations

$$\frac{du}{dt} = Au.$$

Suppose that A has $N-1$ eigenvalues and linearly independent eigenvectors, $Av_j = \lambda_j v_j$, $j = 1, 2, \ldots, N-1$. Show that

$$u(0) = \sum_{j=1}^{N-1} \alpha_j v_j \quad \Rightarrow u(t) = \sum_{j=1}^{N-1} \alpha_j v_j e^{\lambda_j t}.$$

(b) Consider the heat equation with boundary conditions

$$\partial u/\partial t = \partial^2 u/\partial x^2, \qquad u(0,t) = u(1,t) = 0.$$

Use central difference approximation for $\partial^2 u/\partial x^2$, $h = 1/N$. Show that this gives a system where the eigenvalues of A are $\lambda_j = -2N^2(1 - \cos(\pi j/N))$.

The following problems are concerned with the system defined in (b).

(c) Show that $d\|u\|_2^2/dt \leq 0$, and hence $\|u(t)\| \leq \|u(0)\|$.

(d) Show that $\|u_{n+1}\|_2^2 \leq \|u_n\|_2^2$ if the trapezoidal method is used for solving the system. (This shows stability for any step size.)

(e) How many operations are needed for solving the system for u_{n+1} in each step by the trapezoidal method? Compare this to the explicit method.

(f) Generalize to arbitrary equations ($p(x) > 0$)

$$\partial u/\partial t = \partial\big(p(x)\partial u/\partial x\big)/\partial x, \qquad u(0,t) = u(1,t) = 0.$$

4. Consider the heat equation with reversed sign,

$$\frac{\partial u}{\partial t} = -\frac{\partial^2 u}{\partial x^2}, \qquad u(0,t) = u(1,t) = 0.$$

(It corresponds to integrating the usual heat equation backwards in time.)

(a) How is a disturbance of unit magnitude at $(0.5, 0)$ propagated if a difference approximation, analogous to that in Sec. 14.2, is used with $k/h^2 = 1/2$?

(b) Consider the system in Problem 3, where A is obtained by a central difference approximation of $-\partial^2 u/\partial x^2$. Show that the initial value problem is very ill-conditioned when N is large, $h = 1/N$.

5. In a numerical treatment of the Laplace equation $\nabla^2 u = 0$ in a square grid, the numerical values shown below were obtained in six of the grid points. In order to obtain a first approximation in a net with half the grid size, one needs, among others, the values a, b, c indicated below. Compute this with an $O(h^4)$ error—disregarding the errors in the given values.

101		105		117
	a	b	c	
109		109		117

6. (a) Derive a formula for $\partial^2 u/\partial x \partial y(0,0)$ using values u_{ij}, $|i| \leq 1$, $|j| \leq 1$, which is valid for all quadratic functions.

(b) Show that the interpolation formula

$$u_{pq} \approx u_{0,0} + \frac{1}{2}p(u_{1,0} - u_{-1,0}) + \frac{1}{2}q(u_{0,1} - u_{0,-1}) + \frac{1}{2}p^2(u_{1,0} - 2u_{0,0} + u_{-1,0}) + \frac{1}{2}pq(u_{1,1} - u_{1,-1} - u_{-1,1} + u_{-1,-1}) + \frac{1}{2}q^2(u_{0,1} - 2u_{0,0} + u_{0,-1}),$$

where $u_{pq} = u(x_0 + ph, y_0 + qk)$, is valid for all quadratic functions.

7. (a) Verify (14.2.13) and the computation molecule in Fig. 14.2.2.
(b) Show, for example, by operator techniques, that

$$\sum_{i=1}^{6} u_i - 6u_0 = \frac{3}{2}h^2\nabla^2 u_0 + \frac{3}{32}h^4\nabla^4 u_0 + O(h^6),$$

where ∇^2 is the Laplacian operator, and the u_i are the values in the neighbor points P_i, $i = 1, 2, \ldots, 6$ of P_0 in a regular hexagonal lattice with side h.

8. Compute the optimum overrelaxation factor for the five-point difference approximation to the Laplace equation on a 20×20 square grid. (See Example 11.1.5.) Compute the asymptotic rate of convergence for Gauss-Seidel and optimal SOR.

9. Compute approximately the smallest value of λ for which the equation

$$\nabla^2 u + \lambda u = 0$$

has a nontrivial solution which vanishes on the boundary of a triangle with vertices $(0,0)$,$(0,1)$, and $(1,0)$ (the fundamental mode of a triangular membrane). Use a difference method with $h = 1/4$ and $h = 1/5$ and apply Richardson extrapolation. Compare with the exact result $\lambda = 5\pi^2$.

10. Show that the difference equation

$$\nabla_9^2 u = \Big(a + \frac{(ah)^2}{12}\Big)u, \qquad a = \text{constant}$$

is an $O(h^4)$-approximation to the differential equation $\nabla^2 u = au$.

11. The displacement u of a loaded plate satisfies the differential equation

$$\nabla^4 u = f(x, y),$$

where f is the load density. Write the equation as a system

$$\nabla^2 u = v, \qquad \nabla^2 v = f.$$

For a rectangular plate whose four corners are $(-2,-1), (-2,1), (2,1), (2,-1)$, the boundary conditions are, in a certain symmetric loading case,

$$u = \frac{\partial u}{\partial n} = 0,$$

on the two sides $y = 1$ and $x = 2$, and

$$\frac{\partial u}{\partial n} = \frac{\partial v}{\partial n} = 0,$$

on the symmetry lines $y = 0$ and $x = 0$. Sketch a method for calculating the matrix of the system obtained when ∇^2 is approximated by ∇_5^2 on a square grid with mesh width $1/N$. Consider in particular the boundary conditions. What is the order and the band width of the matrix?

12. By Gauss integral theorem, the relation

$$\int_C \frac{\partial u}{\partial n} ds = 0$$

holds for any function which satisfies Laplace's equation everywhere inside a closed curve C. Formulate and prove a similar relation which holds for any solution of the difference equation $\nabla_5^2 = 0$ in a rectangular domain.

13. (a) What happens if one tries to solve an initial-value problem for the Laplace equation by the usual difference method? Investigate how a disturbance of unit magnitude is propagated, e.g., solve recursively $u_{i,j+1}$ from the difference equation with initial conditions

$$u_{i0} = 0, u_{i1} = \begin{cases} 0, & i \neq 0; \\ 1, & i = 0 \end{cases},$$

and boundary conditions $u_{-4,j} = u_{4,j} = 0$.

(b) Write the recurrence relation in vector form

$$u_{j+1} - 2Au_j + u_{j-1} = 0.$$

Compute the eigenvalues of A, and compare with the stability condition $-1 < \lambda < 1$.

14. Set up a difference approximation for the hyperbolic differential equation

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}, \qquad t \geq 0.$$

Initial conditions are: u and $\partial u/\partial t$ given for $t = 0$. Boundary conditions: $u(0,t) = u(1,t) = 0$. Proceed as in Problem 11 (b), and determine a stability condition.

14.3. Variational Methods

14.3.1. Collocation and Ritz Methods To begin with, we consider a *linear* differential equation, valid in a domain $D \subset \mathbf{R}^k$, with linear boundary conditions on ∂D, the boundary of D. We write the differential equation and boundary conditions symbolically in the form

$$Au = f \quad \text{in } D, \qquad Bu = g \quad \text{in } \partial D, \tag{14.3.1}$$

where A and B are linear operators. (Linearity means that $A(\alpha u + \beta v) = \alpha Au + \beta Av$ for all constants α, β and for all functions u, v for which Au, Av, Bu, Bv exist. We assume that this problem has a unique solution.

EXAMPLE 14.3.1.

$$-\frac{d^2 u}{dx^2} = f(x), \quad x \in [0,1], \qquad u(x) = 0, \quad x \in \{0\} \cup \{1\}.$$

EXAMPLE 14.3.2. The Dirichlet problem for Poisson's Equation on a region D.

$$-\nabla^2 u = f(x,y), \quad (x,y) \in D, \qquad u(x,y) = g(x,y), \quad (x,y) \in \partial D.$$

For a linear problem, it is no restriction to *assume that either* $f \equiv 0$ or $g \equiv 0$ in (14.3.1). One only needs to replace u by $u - v$, where v satisfies the differential equation or boundary condition; see Problem 13. (Actually it is sufficient that *either* the differential equation *or* the boundary condition be linear.)

There are a great variety of methods where one seeks an approximate solution in the form

$$u \approx c_1\psi_1 + c_2\psi_2 + \ldots + c_n\psi_n, \tag{14.3.2}$$

where the ψ_i are n given linearly independent functions which span an n-dimensional function space H_n. The methods differ in the choice of H_n as well as in the method of fitting the coefficients to the problem.

1. First, suppose that $g = 0$. It is convenient to choose H_n such that $Bu = 0$, $\forall u \in H_n$. In the classical applications the ψ_i were usually polynomials or trigonometric functions, multiplied by some appropriate function in order to satisfy the boundary conditions. Sometimes eigenfunctions of a simpler equation with the same boundary conditions were chosen. A different choice of the ψ_i is used in the **finite element method**; see the next section.

It then remains to approximate f by linear combinations of the functions $\phi_i = A\psi_i$. In Chap. 9 the discussion of such approximation problems was formulated for functions of one variable, but the *formalism* and Theorem 9.2.5 have a more general validity and can be used for minimizing $\|f - Au\|$, provided that a scalar product (u, v) and the norm or seminorm $\|u\| = (u, u)^{1/2}$ have been defined which satisfy the conditions given in Sec. 9.1.2. For example we could use

$$(u, v) = \sum_{i=1}^{m} u(P_i)v(P_i), \quad P_i \in D, \tag{14.3.3}$$

$$(u, v) = \int\int_D u(x, y)v(x, y)dx\, dy. \tag{14.3.4}$$

If a discrete point set $\{P_i\}_1^m$ is used, the method is called **collocation**. We then obtain a linear system with matrix elements

$$a_{ij} = A\psi_j(P_i).$$

The system is overdetermined if $m > n$.

In **Galerkin's method**, the coefficients are determined by the requirement that the residual function $(f - Au)$ should be orthogonal to all $v \in H_n$, in the sense of (14.3.4), i.e.,

$$(\psi_i, Au - f) = 0, \qquad i = 1, 2, \ldots, n.$$

This yields a linear system

$$A_n u_n = f_n, \tag{14.3.5}$$

where the elements of A_n and f_n are $(\psi_i, A\psi_j)$ and (ψ_i, f), respectively. The minimization of $\|f - Au\|$ mentioned earlier yields a similar system, but with elements $(A\psi_i, A\psi_j)$ and $(A\psi_i, f)$. (Verify these statements!)

If one wants to add one more term in (14.3.2), and if the LU decomposition of A_n is given, the solution of the system $A_{n+1}u_{n+1} = f_{n+1}$ can be obtained cheaply. (The same remark applies to collocation, if $m = n$.) Results for different values of

n can be useful for estimating the order of magnitude of the error and, perhaps, for convergence acceleration.

An operator A in an inner product space H is symmetric if $(u, Av) = (Au, v)$, $\forall u, v \in H$. A symmetric operator is positive definite if $(u, Au) > 0$ when $u \neq 0$. In our case, all $u \in H$ satisfy the boundary conditions $Bu = 0$.

EXAMPLE 14.3.3.

Let

$$(u, v) = \int_0^1 u(x)v(x)dx, \qquad u(x) = v(x), \quad x \in \{0\} \cup \{1\}.$$

Then $A = -d/dx^2$ is symmetric and positive definite. We have, namely,

$$(Au, v) = -\int_0^1 u''(x)v(x)dx = -u^p r(x)v(x)\Big|_0^1 + \int_0^1 u'(x)v'(x)dx.$$

Because of the boundary conditions

$$(Au, v) = -\int_0^1 u''(x)v(x)dx = \int_0^1 u'(x)v'(x)dx,$$

and (u, Av) yields the same result. Evidently, $(Au, u) > 0$, unless $u^{\partial}(x) = 0$, $\forall x$, but since $u(0) = 0$, this implies that $u(x) = 0$, $\forall x$.

It also holds that $A = -\nabla^2$ is symmetric and positive definite. We have the useful formula

$$(-\nabla^2 u, v) = (\nabla u, \nabla v), \tag{14.3.6}$$

and more generally

$$(-div(\rho \nabla u), v) = (\rho \nabla u, \nabla v), \tag{14.3.7}$$

If a is positive definite, a new scalar product $< \cdot, \cdot >$ and norm can be defined

$$< u, v >= (u, Av), \qquad \|u\|^2 =< u, u > . \tag{14.3.8}$$

(It is easy to verify the conditions in Sec. 9.1.2.)

THEOREM 14.3.1.

Suppose that A is positive definite. Let u^* be the solution of $Au = f$, $Bu = g$. Then the Galerkin method gives the best approximation in H_n to u^*, measured in the norm (14.3.8).

PROOF: By Theorem 9.2.5, the solution of this minimization problem is given by the condition $, v, u - u^* >= 0$, $\forall v \in H_n$. But

$$< v, u - u^* >= (v, Au - Au^*) = (v, Au - f),$$

and the Galerkin condition is $(v, Au - f) = 0, \forall v \in H_n$. As an exercise the reader can derive the the relation

$$(u, Au) - 2(f, u) =< u - u^*, u - u^* > - < u^*, u^* > .$$

It follows form this that $u = u^*$ minimizes the functional on the left hand side among all functions u for which the functional exists. It also follows that the minimization of this functional in H_n is obtained by the Galerkin method. ∎

There exist **variational principles** in many areas of physics. This usually means that problems in that area can be formulated as minimization problems for some energy integral or some other functional. The approximate solution of such problems by a function containing a finite number of parameters is known as the **Ritz method**. By the previous remarks, the Ritz method is often equivalent to the Galerkin method, but this is not always the case; see Collatz [3, 1966].

2. Next suppose that $f = 0$. Then, if possible, H_n should be built up by particular solutions of the differential equation, i.e., $Au = 0$, $\forall u \in H_n$. For the Laplace equation in two dimensions, the real and imaginary parts of any analytic function of $z = x + iy$ are particular solutions. Then minimization of $\|Bu - g\|$, collocation, or the Galerkin method can be used to fit the boundary conditions. In the previous formulas, A, f should be replaced by B, g, and the scalar products will be integrals (or sums) on the boundary.

3. Finally, one can use the methods described with a space H_n where neither the differential or the boundary conditions are satisfied, but then n must be large enough so that a good fit can be made to both the equations and boundary values.

In this general form, the ideas presented above can be applied to problems where both the differential equation and boundary conditions are *nonlinear*. If one of them is linear, H_n can be constructed according to the suggestions above. The nonlinear system of minimization problem obtained can be treated by the methods of Sec. 12.1.

14.3.2. Finite Elements Methods The finite element method (FEM) is a method for approximate treatment of physical problems, defined, for example, by differential equations with boundary conditions, or by variational principles. Compared to other methods it has been successful when the domain of the problem has a complicated shape or when the functions involved behave very differently in different parts of the domain. It is an important tool in structural mechanics, but there are numerous other applications; see, e.g., Johnson [1987].

In the finite element method the domain is represented by a collection of a finite number of Connected subdomains of simple shape (e.g., triangles). Functions are approximated *locally* over each subdomain by continuous functions uniquely defined in terms of the values of the function (and possibly some of its derivatives) at certain points, called **nodes**, inside or on the boundary of the subdomain. Our treatment in Sec. 8.5 3 of interpolation and double integrals in triangular domains is in the spirit of the finite element method.

In most cases, finite element methods can be looked upon as applications of the *Ritz or Galerkin methods with a particular type of local basis functions*. In order to explain this, we first consider the simplest particular case; piecewise

linear interpolation in one variable. In Sec. 4.6.5 we introduced the B-splines

$$B_{i,2}(x) = \begin{cases} (x - x_{i-1})/h_i & x \in [x_{i-1}, x_i); \\ 1 - (x - x_i)/h_{i+1} & x \in [x_i, x_{i+1}); \\ 0 & \text{otherwise.} \end{cases}, \qquad i = 1, \ldots, n-1,$$

where x_i are the nodes and $h_i = (x_i - x_{i-1})$. In each interval $E_i = [x_{i-1}, x_i)$ the function is represented by the broken line in Fig. 14.3.2 can be expressed in the form

$$u = \sum_{k=1}^{n} u_k \psi_k(x) = u_{i-1}\psi_{i-1}(x) + u_i \psi_i(x), \qquad x \in E_i, \tag{14.3.9}$$

since all other basis functions are zero outside the interval E_i.

FIG. 14.3.1. *Broken line approximation and basis function* $\psi_3(x)$.

In order to obtain higher degree interpolation, we can introduce internal nodes in the elements. A representation similar to (14.3.1) can be obtained by the Lagrange interpolation formula, when the ψ_i are piecewise polynomials, one for each node.

Functions defined by piecewise linear interpolation on triangular grids (see Eq. (8.5.11)) can also be expressed in the form of equation (14.3.1), with one term ψ_i for each node P_i, such that $\psi_i(P_j) = \delta_{ij}$ (the Kronecker delta), and $\psi_i(x) \neq 0$ only when x belongs to an element containing P_i, see Fig. 14.3.2.

FIG. 14.3.2. *Support for basis function* $\psi_i(x)$.

Now consider the finite element solution to a linear boundary value problem

$$Au(P) = f, \quad P \in D, \qquad u(P) = 0, \quad P \in \partial D.$$

The subdomains are run through in turn, and the contribution of each subdomain to A_n and f_n in (14.3.5) is computed. Equations similar to (14.3.7) are useful

here—they are even necessary if the ψ_i have discontinuous first derivatives on the boundaries of the subdomains. In structural mechanics A_n is called the stiffness matrix. Note that A_n will be sparse when n is large, since $(\psi_i, A\psi_j) = 0$, *unless there is an element containing both P_i and P_j*. The resulting system of linear equations is usually solved by *direct* methods, for sparse systems, see Sec. 6.5.

When the finite element method is applied to Poisson's equation, and other equations, which can be derived from a variational principle containing only derivatives of first order, then it is appropriate to work with basis functions, whose first derivatives are discontinuous on the boundaries of the subdomains (for example, piecewise linear functions). It may seem strange to use functions whose second derivative does not exist at the boundaries, for solving second order differential equations, but this works fine.

When solving plate equations the variational principle contains partial derivatives of second order. The differential equation then is of fourth order; in special cases one gets the biharmonic equation $\nabla^4 u = f$. Then the basis functions should have element continuous derivatives of second order. The first order derivatives must then be continuous everywhere. Details about how to construct such basis functions and much more about the finite element method can be found in monographs, e.g., Strang and Fix [1973], Johnson [1987].

REVIEW QUESTIONS

1. Describe in your own words:

(a) collocation; (b) the Galerkin method; (c) the Ritz method.

The first two methods can be applied in two different ways to a linear boundary value problem. How?

2. Describe an application of the finite element method to a two-dimensional problem.

PROBLEMS

1. Let D be the ellips $x^2/a^2 + y^2/b^2 = 1$. Consider the boundary-value problem

$$\nabla^2 u = 1, \quad (x, y) \in D, \qquad u = x^4 + y^4, \quad (x, y) \in \partial D.$$

(a) Reduce it to a problem with homogeneous boundary conditions.

(b) Reduce it to a Dirichlet problem for the Laplace equation.

(c) Which symmetry properties will the solution have? (You may assume that the problem has a unique solution.)

2. Let $p_n(x, y), q_n(x, y)$ be the real and imaginary parts of $(x + iy)^n$. By analytic function theory, p_n, q_n satisfy the Laplace equation. Consider a truncated expression

$$u \approx \sum_{n=0}^{N} (a_n p_n + b_b q_n).$$

(a) Which coefficients will certainly be zero, if $u(x, y) = u(-x, y)$, $\forall x, y$?

(b) How can you reduce the number of coefficients in the application of this

expansion to Problem 13 (b), by utilizing symmetry properties?

(c) Write a program for the calculation of the matrix and the right-hand side in an application of the collocation method to Problem 13 (b) with M given boundary points, $M \geq N/2 + 1$.

3. In the notation of Sec. 14.1.1, is it true that $(v, Au) = v_n^T A_n u_n$, whenever $u, v \in H_n$?

4. (a) In the notation of Theorem 14.2.1, show that

$$(u, Au) - 2(f, u) = < u - u^*, u - u^* > - < u^*, u^* > .$$

(b) Show that the solution of the Poisson equation, $-\nabla^2 u = f$, with boundary conditions $u = 0$ on ∂D, minimizes the integral

$$\int\int_D \left((\nabla u)^2 - 2fu\right) dxdy$$

among all functions, satisfying the boundary conditions.

(c) What differential equation corresponds to the minimization of the integral

$$\int\int_D \left(p(\nabla u)^2 + gu^2 - 2fu\right) dxdy,$$

where p, f, g are given functions of x, y; p and g are positive.

Hint: Use (14.3.7).

5. (a) Set up the equation for the finite element solution of

$$-u''(x) = f(x), \quad x \in [0, 1], \qquad u(0) - u(1) = 0,$$

with the piecewise linear basis functions appearing in (14.3.9). Divide into N subintervals of equal length. Treat f as a piecewise constant function. Compare with the finite difference solution.

(b) The same question for the differential equation

$$-\frac{d}{dx}\left(p\frac{du}{dx}\right) + gu - f = 0.$$

14.4. Integral Equations

14.4.1. Fredholm Integral Equations Problems in partial differential equations can sometimes be transformed into integral equations. Many mathematical models, e.g., in statistical physics, also lead directly to integral equations or integro-differential equations.

The equations

$$\int_a^b K(x, y)u(y)dy = f(x), \tag{14.4.1}$$

$$u(x) - \int_a^b K(x, y)u(y)dy = f(x), \tag{14.4.2}$$

where u is the unknown function, are called **Fredholm integral equations** of the **first** and **second** kind, respectively. The function $K(x, y)$ is called the **kernel**. For the mathematical theory, see Courant-Hilbert, [4].

A natural approach in solving integral equations is to use numerical integration or collocation. The trapezoidal method is accurate if the functions are periodic, with period $(b-a)$. In the nonperiodic case the trapezoidal approximation with Richardson extrapolation has been successful for equations of the second kind. Also Gauss's quadrature can be recommended. In any of these cases, for (14.4.1) we obtain the linear system

$$\sum_{j=1}^{n} K(x_i, y_j) w_j u_j = f(x_i), \quad i = 1, 2, \ldots, m, \tag{14.4.3}$$

where the w_j are the coefficients in the quadrature formula $(m \geq n)$. Similarly, for (14.4.2) we obtain

$$u_i - \sum_{j=1}^{n} K(x_i, y_j) w_j u_j = f(x_i), \quad i = 1, 2, \ldots, m, \tag{14.4.4}$$

For equations of the *first kind* with smooth kernels, the system of equations of (14.4.3) are likely to be very illconditioned, with numerical rank $r < n$. The rank can be estimated from a rank revealing QR decomposition, see Sec. 7.6.2. Alternatively one can use the singular value decomposition and take the truncated SVD solution, see Sec. 7.3.2. Equations of the second kind are more well-behaved. A simple example can illustrate this.

Example 14.4.1.

Consider the "very smooth" kernel $K(x, y) = k \neq 0$, $[a, b] = [0, 1]$. The equation of the first kind reads

$$k \int_0^1 u(y)dy = f(x).$$

This equation has no solution unless $f = c$, a constant, in which case *any* function for which $\int_0^1 u(y)dy = c/k$ is a solution. The equation of the *second* kind reads

$$u(x) - k \int_0^1 u(y)dy = f(x).$$

If we put $u(x) = f(x) + kc$, then

$$c = \int_0^1 u(y)dy = \int_0^1 f(y)dy + kc,$$

and it follows that

$$u(x) = f(x) + \frac{k}{1-k} \int_0^1 f(y)dy.$$

Hence the solution is unique, unless $k = 1$. If $k = 1$, there is no solution unless $\int_0^1 f(y)dy = 0$, in which case the general solution is $u(x) = f(x) + p$, where p is an arbitrary constant.

It is convenient to use operator notation and write the integral equations in (14.4.1) and (14.4.2) in the form

$$Ku = f, \qquad (I - K)u = f,$$

respectively. An **integral operator** K can be regarded as a continuous analogue of a matrix.

If the homogeneous equation $Ku = \lambda u$ has a nontrivial solution, ten λ is called an eigenvalue of K. If the kernel $K(x, y)$ is square integrable, then, as a rule, the operator K has an enumerable infinity of eigenvalues with a limit point in the origin. If the kernel has the form

$$K(x, y) = \sum_{p=1}^{r} \chi_p(x)\psi_p(y), \tag{14.4.5}$$

where both sets of functions $\{\chi_p\}_1^r$ and $\{\psi_p\}_1^r$ are (internally) linearly independent, K spans an r-dimensional space; the rank of K is r. (In Example 14.4.1 $r = 1$.) Zero is then an eigenvalue of infinite multiplicity, since $ku = 0$ for any function u orthogonal to $\{\psi_p\}_1^r$. The equation of the first kind $Ku = f$ is solvable only if f is a linear combination of $\{\chi_p\}_1^r$.

Kernels can be approximated to the rounding-error level by sums of the form of (14.4.5); the smoother the kernel, the smaller r. We may call that value of r the numerical rank of K. Therefore, one can expect ill-conditioned linear systems with any method for the numerical solution of equations of the first kind with smooth kernels. Equations of the second kind, however, cause no trouble, unless an eigenvalue of K happens to be close to unity.

14.4.2. The Galerkin Method The Galerkin method (see Sec. 14.2) can be applied to integral equations. For an equation of the second kind, put

$$u(x) \approx f(x) + \sum_{p=1}^{n} c_p \psi_p(x). \tag{14.4.6}$$

The Galerkin equations

$$(\psi_i, (I - K)u) = (\psi_i, f), \quad i = 1, 2, \ldots, n$$

can then be expressed in matrix form $A_n u_n = g_n$, where (see Problem 18) the elements of A_n, u_n, g_n are $(\psi_i, (I - K)\psi_j), c_j, (\phi_i, f)$, respectively. For equations of the first kind the term $f(x)$ is to be dropped from (14.4.6), and we obtain a linear system $K_n u_n = f_n$, where the elements of K_n are $(\psi_i, K\psi_j)$. The system is ill-conditioned if n exceeds the numerical rank of K, and the special techniques referred to above should be used.

We note that for an equation of the second kind the Galerkin method gives the *exact solution* when the kernel is of the form of (14.4.6) with $n = r$, $\psi_p(x) = \chi_p(x)$. The eigenvalues of K are approximated by the values of λ which make the matrix

$K_n - \lambda M_n$ singular, where the elements of M_n are (ψ_i, ψ_j). This is a generalized eigenvalue problem, see Sec. 10.9.

The operator formalism makes the techniques of this section applicable to many linear operators other than integral operators. Examples of integro-differential equations and nonlinear integral equations are given in the problems of this section. More information about the numerical treatment of integral equations and other operator equations can be found, e.g., in Collatz [3], Todd [14] and Rall [12]. Rall's book is an excellent guide on the bumpy road from a Banach space to a working computer program.

REVIEW QUESTIONS

1. Describe two numerical methods for the solution of linear integral equations. Why are, as a rule, equations of the first kind with smooth kernels more troublesome than those of second kind?

PROBLEMS

1. (a) Determine the rank and the eigenvalues of the operator K, defined by

$$(Ku)(x) = \int_0^{2\pi} \cos(x+y)u(y)dy.$$

(b) Solve the integral equation of second kind $(I-K)u = 1$.

(c) Give the general solution of the equation $(Ku)(x) = \cos x + ax$, when a solution exists. (a is constant.)

2. A kernel is symmetric when $K(x,y) = K(y,x)$ for all x, y. Are the matrices A_n and K_n symmetric in that case?

3. Approximate the integro-differential equation

$$\frac{\partial u}{\partial t} - \int_0^1 K(x,y)u(y,t)dy = f(x,t)$$

by a finite system of linear ordinary differential equations, using numerical integration and collocation.

4. Suggest a numerical method for the following *nonlinear* equation:

$$u(x) - \int_a^b K(x,y,u(x),u(y))dy = f(x).$$

5. Many integral equations can be solved in closed form; see Collatz [3, Chap. 6] or textbooks on integral equations. Suppose that equations of the form $Av = f$ can be solved in closed form and that we want the solution of the equations $(A-K)u = f$.

(a) Set up the Galerkin equations for this problem, with the approach $u \approx v + \sum_{j=1}^{n} c_j\psi_j$, where $Av = f$.

(b) Show that the Galerkin method gives the exact solution if the operator K is of rank n and if the functions $A\phi_j$ are a basis for the n-dimensional space of functions which can be expressed in the form Kg, where g is an arbitrary function.

Notes and References

Elementary introductions to numerical solution of *partial* differential equations are given in Smith [13] and in Morton and Mayers [9]. Finite difference methods are also treated in Mitchell and Griffiths [11, 1980]. A good introduction to finite element methods is given by Johnson [8, 1987]. For a primer on multigrid methods see Briggs [2, 1987]. Time dependent problems are treated in Gustafsson, Kreiss and Oliger [6, 1996]

A comprehensive treatment of integral equations is given by Baker [1, 1977] Computational methods for integral equations are also covered in Delves and Mohamed [5, 1985].

References

[1] C. T. H. Baker. *The Numerical Treatment of Integral Equations.* Clarendon Press, Oxford, 1977.

[2] W. L. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, 1987.

[3] L. Collatz. *The Numerical Treatment of Differential Equations. 3rd ed.* Springer-Verlag, Berlin, 1966.

[4] R. Courant and D. Hilbert. *Methods of Mathematical Physics. Vol. I.* Interscience, New York, NY, 1953.

[5] Courant L. M. Delves and J. L. Mohamed. *Computational Methods for Integral Equations.* Cambridge University Press, Cambridge, UK, 1985.

[6] B. Gustafsson, H.-O. Kreiss, and J. Oliger. *Time Dependent Problems and Difference Methods.* John Wiley, New York, 1996.

[7] W. Hackbusch. *Multi-Grid Methods and Applications.* Springer-Verlag, Berlin-Heidelberg-New York, 1985.

[8] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Studentlitteratur, Lund, Sweden, 1987.

[9] K. W. Morton and D. F. Mayers. Numerical Solution of Partial Differential Equations", Cambridge University Press, Cambridge, U.K., 1994.

[10] S. McCormick. *Multigrid Methods.* SIAM, Philadelphia, PA, 1987.

[11] A. R. Mitchell and D. F. Griffiths. *The Finite Difference Method in Partial Differential Equations.* John Wiley, New York, 1980.

[12] L. B. Rall *Computational Solution of Nonlinear Operator Equations.* McGraw-Hill, New York

[13] G. D. Smith. *Numerical Solution of Partial Difference Equations. 3rd. ed.*, Oxford University Press, Oxford, 1985.

[14] J. Todd, editor. *A Survey of Numerical Analysis.* McGraw-Hill, New York, 1962.

[15] D. M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.

Chapter 15

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).

15.1. Guide to Literature

15.1.1. Textbooks in Numerical Analysis In scientific and technical applications, one often needs to supplement one's knowledge of mathematical analysis. The two volumes by Courant and Hilbert [9, 1953] and [10, 1962] are still a good source, although some sections are quite advanced. An excellent modern introduction to applied mathematics is given by Strang [42, 1986]. For a rigid exposition of modern analysis Dieudeonné [12, 1961] is recommended. A valuable sourcebook to the literature before 1956 is Parke [33, 1958].

Textbooks which can be read as a complement to this book include Conte and de Boor [6, 1980], Deuflhard and Hohmann [11, 1995], and the two books by Stewart [39, 40] In order to make applications easier, some books contain listings of algorithms, or even comes with a disk containing software. This is true of the introductory book by Forsythe, Malcolm, and Moler [13, 1977] and its successor Kahaner, Moler and Nash [26, 1988].

More advanced classical texts include Isaacson and Keller [25, 1966], Ralston

and Rabinowitz [35, 1978], and Schwarz [38, 1989]. The authoritative book by Stoer and Bulirsch [41, 1992], translated from German, is particularly suitable for a reader who has a good mathematical background.

References

[1] F. S Acton. *Numerical Methods That (Usually) Work,* 2nd ed.,. Math. Assoc. of America, New York, 1990.

[2] K. E. Atkinson. *An Introduction to Numerical Analysis* 2nd. ed. John Wiley, New York, NY, 1989.

[3] E. K Blum. *Numerical Analysis and Computation: Theory and Practice.* Addison-Wesley, Reading, MA, 1971.

[4] E. W. Cheney and D. Kincaid. *Numerical Mathematics and Computing.* 3rd ed. Brooks and Cole, Pacific Grove, CA, 1994.

[5] E E. W. Cheney and D. Kincaid. *The Mathematics of Scientific Computing.* Brooks and Cole, Pacific Grove, CA, 1991.

[6] S. D. Conte and C. de Boor. *Elementary Numerical Analysis. An Algorithmic Approach.* 3rd ed. McGraw-Hill, New York, 1980.

[7] R. Courant. *Differential and Integral Calculus. Vol. I.* Blackie & Son, London, 1934. Reprinted in Classics Library, John Wiley, New York, NY, 1988.

[8] R. Courant. *Differential and Integral Calculus. Vol. II.* Blackie & Son, London, 1936. Reprinted in Classics Library, John Wiley, New York, NY, 1988.

[9] R. Courant and D. Hilbert. *Methods of Mathematical Physics. Vol. I.* Interscience, New York, NY, 1953.

[10] R. Courant and D. Hilbert. *Methods of Mathematical Physics. Vol. I.* Interscience, New York, NY, 1962.

[11] P. Deuflhard and A. Hohmann. *Numerical Analysis. A First Course in Scientific Computing.* de Gruyter, Berlin, New York, 1995.

[12] J. Dieudonné. *Foundations of Modern Analysis.* Academic Press, New York, NY, 1961.

[13] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer Method for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[14] C.-E. Fröberg. *Numerical Mathematics. Theory and Computer Applications.* The Benjamin/Cummings, Menlo Park, CA, 1985.

[15] W. Gautschi. *Numerical Analysis.* Birkhäuser, Boston, NY, 1997.

[16] H. H. Goldstine. *A History of Numerical Analysis from the 16th through the 19th Century.* Springer-Verlag, 1977.

[17] G. H. Golub, editor. *Studies in Numerical Analysis.* The Math. Assoc. of America, 1984.

[18] G. H. Golub and J. M. Ortega. *Scientific Computing and Differential Equations. An Introduction to Numerical Methods.* Academic Press, 1992.

[19] G. H. Golub and J. M. Ortega. *Scientific Computing. An Introduction with Parallel Computing.* Academic Press, 1993.

[20] G. Hämmerling and K.-H. Hoffmann. *Numerical Mathematics.* Springer-Verlag, Berlin, 1991.

[21] R. W. Hamming. *Numerical Methods for Scientists and Engineers.* 2nd ed. McGraw-Hill, New York, NY, 1974.

[22] P. Henrici. *Elements of Numerical Analysis.* John Wiley, New York, NY, 1964.

[23] F. B. Hildebrand. *Introduction to Numerical Analysis.* McGraw-Hill, New York, NY, 1974.

[24] A. S. Householder. *Principles of Numerical Analysis.* McGraw-Hill, New York, NY, 1953.
[25] E. Isaacson and H.B. Keller. *Analysis of Numerical Methods.* Dover, New York Corrected reprint of 1966 original.
[26] D. Kahaner, C. B. Moler, and S. Nash. *Numerical Methods and Software.* Prentice-Hall, Englewood Cliffs, NJ, 1988.
[27] D. Kincaid and W. Cheney. *Numerical Analysis.* 2nd. ed. Brooks/Cole, Pacific Grove, CA, 1996.
[28] C. Lanczos. *Applied Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1956.
[29] G. I. Marchuk. *Methods in Numerical Mathematics.* 2nd. ed. Springer-Verlag, Berlin, 1982.
[30] J. H. Mathews. *Numerical Methods–for Computer Science, Engineering and Mathematics.* Prentice-Hall, New York, NY, 1987.
[31] J. C. Nash. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation.* 2nd ed. Adam Hilger: Bristol & American Institute of Physics, New York, NY, 1990.
[32] J. Ortega. *Numerical Analysis: A Second Course.* Academic Press, New York, NY, 1972.
[33] N. G. III. Parke. *Guide to the Literature of Mathematics and Physics.* 2nd. ed. Dover Publications, New York, NY, 1958.
[34] J. Phillips. *The NAG Library. A Beginners Guide.* Clarendon Press, Oxford, GB, 1986.
[35] A. Ralston and P. Rabinowitz. *A First Course in Numerical Analysis.* 2nd ed. McGraw-Hill, New York, 1978.
[36] J. R. Rice. *Numerical Methods, Software, and Analysis.* Academic Press, New York, NY, 1983.
[37] H. Rutishauser. *Lectures on Numerical mathematics.* Birkhäuser, Boston, MA.
[38] H. R. Schwarz. *Numerical Analysis. A Comprehensive Introduction.* John Wiley, New York, Ny, 1989.
[39] G. W. Stewart. *Afternotes on Numerical Analysis.* SIAM, Philadelphia, 1996.
[40] G. W. Stewart. *Afternotes Goes to Graduate School.* SIAM, Philadelphia, 1998.
[41] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* 2nd ed. Springer-Verlag, New York, 1992.
[42] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, USA, 1986.
[43] C. W. Ueberhuber. *Numerical Computation. 1 & 2.* Springer-Verlag, Berlin Heidelberg New York, 1997.
[44] C. F. Van Loan. *Introduction to Scientidic Computing.* Prentice Hall, New Jersey, 1997.
[45] J. S. Vandergraft. *Introduction to Numerical Computations.* Academic Press, New York–London, 1983.
[46] D. M. Young and R. T. Gregory. *A Survey of Numerical Analysis. Vol. 1.* Addison-Wesley, Reading, MA, 1972.
[47] D. M. Young and R. T. Gregory. *A Survey of Numerical Analysis. Vol. 2.* Addison-Wesley, Reading, MA, 1973.

15.1.2. Handbooks, Computer Arithmetic, Tables and Formulas

Press et al. [19, 1992] gives an unsurpassed survey of contemporary numerical methods for the applied scientist. The book by Gander and Hřebiček [8, 1997]

contains worked through examples on how to use modern software tools in problem solving.

The three monographs edited by Jacobs [23, 1977], Iserles and Powell [24, 1987], and Duff and Watson [25, 1997] give good surveys of the development of "state of the art" methods in many different areas of numerical analysis during the last decades. The major handbook series [4] edited by P. G. Ciarlet and J. L. Lions is another valuable source.

Some principal questions in the production of software for mathematical computation are discussed in Rice [21, 1971]. A good presentation of different aspects of number systems is given in Knuth [14, 1981]. The IEEE standard for binary floating point arithmetic is defined in [3, 1985] An excellent tutorial on floating-point arithmetic can be found in Goldberg [9, 1991].

The classical treatise on rounding error analysis for floating point arithmetic is developed by Wilkinson [27, 1963], see also Wilkinson [28, 1986]. Alefeld and Herzberger [2, 1983] and Moore [17, 1988] give surveys of the use of interval arithmetic.

Mathematical tables are no longer as important for numerical calculations as they were in the pre-computer days. However, tables can often be an important aid in checking a calculation or planning calculations on a computer. Detailed advice about the use and choice of tables is given in Todd [26, 1962, pp.93–106]. The classical six-figure tables of Comrie [6, 1955] contains a collection of trigonometric and hyperbolic tables and formulas, and is often useful to have available. A most comprehensive source of information on mathematical functions and formulas is Abramowitz and Segun [1, 1965]. The most detailed treatment of computer approximation of elementary functions is given by Cody and Waite [5, 1980].

References

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions.* National Bureau of Standards, Dover Publications, New York, NY, 1965.

[2] G. Alefeld and J. Herzberger. *Introduction to Interval Computation.* Academic Press, New York, NY, 1983.

[3] Anon. IEEE standard 754-1985 for binary floating-point arithmetic. *SIGPLAN 22*, 2:9–25, 1985.

[4] P. G. Ciarlet and J. L. Lions. *Handbook of Numerical Analysis.* Vol. I-V. North-Holland, Amsterdam, 1990–1997.

[5] W. J. Cody and W. Waite. *Software Manual for the Elementary Functions.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[6] L. J. Comrie. *Chamber's Shorter Six-Figure Mathematical Tables.* W.& R. Chambers, Edinburgh, 1968.

[7] C. T. Fike. *Computer Evaluation of Mathematical Functions.* Prentice-Hall, Englewood Cliffs, NJ, 1968.

[8] W. Gander and J. Hřebiček. *Solving Problems in Scientific Computing using Maple and Matlab*, 3rd ed.. Springer-Verlag, Berlin, 1997.

[9] D. Goldberg. What every computer scientist should know about floating point arithmetic. *ACM Computing Surveys*, 23:5–48, 1991.

[10] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series and Products.* 5th ed. Academic Press, London, 1993.
[11] J. F. Hart et al. *Computer Approximations.* John Wiley, New York, NY, 1968.
[12] C. Hastings. *Approximations for Digital Computers.* Princeton University Press, Princeton, NJ, 1955.
[13] E. Jahnke, F. Emde, and F. Lösh. *Tables of Higher Functions.* 6th ed. McGraw-Hill, New York, NY, 1960.
[14] D. E. Knuth. *The Art of Computer Programming, Vol. 2. Seminumerical Algorithms.* 2nd ed. Addison-Wesley, Reading, MA, 1981.
[15] A. V. Lebedev and R. M. Federova. *A Guide to Mathematical Tables.* D. Van Nostrand, New York, NY, 1960.
[16] Y. L. Luke. *Mathematical Functions and Their Approximations.* Academic Press, New York, 1975.
[17] R. E. Moore. *Reliability in Computing. The Role of Interval Methods in Scientific Computing.* John Wiley, New York, Ny, 1988.
[18] Yu. A. Brychkov, A. P. Prudnikov, and O. I. Marichev. *Integrals and Series. Vol. 1: Elementary Functions.* Gordon and Breach Sci. Publ., New York, 1986.
[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes; The Art of Scientific Computing.* 2nd. ed., Cambridge University Press, Cambridge, GB, 1992.
[20] A. P. Prudnikov, Yu. A. Brychkov, and O. I. Marichev. *Integrals and Series. Vol. 2: Special Functions.* Gordon and Breach Sci. Publ., New York, 1986.
[21] J. R. Rice. *Mathematical Software.* Academic Press, New York, NY, 1971.
[22] J. Spanler and K. B. Oldham. *An Atlas of Functions.* Springer-Verlag, Berlin, 1987.
[23] D. A. H. Jacobs, editor. *The State of the Art in Numerical Analysis.* Clarendon Press, Oxford, GB, 1977.
[24] A. Iserles and M. J. D. Powell, editors. *The State of the Art in Numerical Analysis.* Clarendon Press, Oxford, GB, 1987.
[25] I. S. Duff and G. A. Watson, editors. *The State of the Art in Numerical Analysis.* Clarendon Press, Oxford, GB, 1997.
[26] J. Todd, editor. *A Survey of Numerical Analysis.* McGraw-Hill, New York, 1962.
[27] J. H. Wilkinson. *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, NJ, 1963.
[28] J. H. Wilkinson. Error analysis revisited. *IMA Bull*, 22:192–200, 1986.

15.1.3. Linear Algebra and Applications The literature on linear algbera is very extensive. For a theoretical treatise a classical source is Gantmacher [16, 1959]. Several nonstandard topics are covered in Lancaster and Tismenetsky [29, 1985] and in two excellent volumes by Horn and Johnson [25, 1985] and [26, 1991]. A very complete and useful book on and perturbation theory and related topics is Stewart and Sun [42, 1990]. Analytical aspects are emphasized in Lax [31, 1997].

An interesting survey of classical numerical methods in linear algebra can be found in Faddeev and Faddeeva [13, 1963], although many of the methods treated are now dated. A compact, lucid and still modern presentation is given by Householder [27, 1964]. Bellman [4, 1970] is an original and readable complementary text.

An excellent textbook on matrix computation are Stewart [41, 1973]. The

recent book [43, 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 [19, 1996], which has become a standard reference. The book by Higham [24, 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 [5, 1996].

Two classic texts on iterative methods for linear systems are Varga [46, 1962] and Young [50, 1971]. The more recent book by Axelsson [2, 1994], also covers conjugate gradient methods. Barret et al. [3, 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 [39, 1996].

A still unsurpassed text on computational methods for the eigenvalue problem is Wilkinson [48, 1965]. Wilkinson and Reinsch [49, 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 [36, 1980]. Large scale eigenvalue problems are treated by Saad [38, 1992]. For an introduction to the implementation of algorithms for vector and parallel computers, see also Dongarra et al. [11, 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. [10, 1979] and Smith et al. [40, 1976]. Direct methods for sparse symmetric positive definite systems are covered in George and Liu [18, 1981], while a more general treatise is given by Duff et al. [12, 1986].

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, editors. *LAPACK Users' Guide. Second Edition.* SIAM, Philadelphia, 1995.

[2] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, UK, 1994.

[3] R. Barret, M. W. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, editors. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1993.

[4] R. Bellman. *Introduction to Matrix Analysis.* 2nd. ed. McGraw-Hill, New York, 1970.

[5] Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[6] C. Brezinski. *Projection Methods for System of Equations.* Elsevier, Amsterdam, 1997.

[7] F. Chatelin *Eigenvalues of Matrices.* Wiley, Chichester.

[8] P. G. Ciarlet. *Introduction to Numerical Linear Algebra and Optimization.* Cambridge University Press, Cambridge, UK, 1989.

[9] J. W. Demmel. *Applied Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[10] J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. SIAM, Philadelphia, 1979.
[11] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst. *Numerical Linear Algebra for High-Performance Computers*. SIAM, Philadelphia, 1998.
[12] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, London, 1986.
[13] D. K. Faddeev and V. N. Faddeeva. *Computational Methods of Linear Algebra*. W. H. Freeman, San Francisco, CA, 1963.
[14] L. Fox. *Introduction to Numerical Linear Algebra*. Clarendon Press, Oxford, 1964.
[15] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. In A. Iserles, editor, *Acta Numerica 1992*, pages 57–100. Cambridge University Press, Cambridge, UK, 1992.
[16] F. R. Gantmacher. *The Theory of Matrices. Vols. I and II*. Chelsea Publishing Co., New York, NY, 1959.
[17] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and G. W. Stewart. *Matrix Eigensystems Routines: EISPACK Guide Extension*. Springer-Verlag, New York, 1977.
[18] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981.
[19] G. H. Golub and C. F. Van Loan, *Matrix Computations,* 3rd ed.. Johns Hopkins University Press, Baltimore, 1996.
[20] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, 1997.
[21] R. T. Gregory and L. K. Karney. *A Collection of Matrices for Testing Computational Algorithms*. John Wiley, New York, 1969.
[22] L. A. Hageman and D. M. Young. *Applied Iterative Methods*. Academic Press, New York, NY, 1981.
[23] W. W. Hager. *Applied Numerical Linear Algebra*. Prentice-Hall, Englewood Cliffs, NJ, 1989.
[24] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 1996.
[25] R. A. Horn and C. R. Johnson. *Matrix Analysis*. University Press, Cambridge, UK, 1985.
[26] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1991.
[27] A. S. Householder. *The Theory of Matrices in Numerical Analysis*. Dover, New York, 1964.
[28] A. Jennings and J. J. McKeown. *Matrix Computation,* 2nd. ed. John Wiley, New York, NY, 1992.
[29] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, New York, 1985.
[30] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Classics in Applied Mathematics. SIAM, Philadelphia, 1995.
[31] P. D. Lax. *Linear Algebra.* John Wiley, New York, NY, 1997.
[32] S. J. Leon. *Linear Algebra with Applications*. 4th ed. Macmillan, New York, 1994.
[33] M. Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities*. Allyn and Bacon, Boston, 1964.
[34] B. Noble and J. W. Daniel. *Applied Linear Algebra,* 2nd. ed. Prentice-Hall, Englewood Cliffs, NJ, 1977.

[35] J. M. Ortega. *Matrix Theory. A Second Course.* Plenum Publ. Co., New York, 1987.

[36] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Classsical in Applied Mathematics 20, SIAM, Philadelphia, PA, 1998.

[37] J. R. Rice. *Matrix Computation and Mathematical Software.* Mc Graw-Hill, New York, 1981.

[38] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, Manchester, UK, 1992.

[39] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, MA, 1996.

[40] B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystems Routines - EISPACK Guide,* 2nd. ed. Springer-Verlag, New York, NY, 1976.

[41] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, New York, 1973.

[42] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, Boston, 1990.

[43] G. W. Stewart. *Matrix Algorithms: Volume I: Basic Decompositions.* SIAM, Philadelphia, PA, 1998.

[44] G. Strang. *Linear Algebra and Its Applications,* 3rd ed. Academic Press, New York, NY, 1988.

[45] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra,* SIAM, Philadelphia, PA, 1997, pp. 361.

[46] R. S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1962.

[47] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. Assoc. Comput. Mach*, 8:281–330, 1961.

[48] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[49] J. H. Wilkinson and C. Reinsch, editors. *Handbook for Automatic Computation. Vol. II, Linear Algebra.* Springer-Verlag, New York, 1971.

[50] D. M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.

15.1.4. Nonlinear Systems and Optimization Classic books on methods for solving a single non-linear equations are Traub [18, 1964], Householder [10, 1970], and Ostrowski [16, 1973]. A thorough but mostly theoretical treatise on nonlinear systems is found in Ortega and Rheinboldt [15, 1970]. A standard textbook on numerical methods for unconstrained optimization, nonlinear systems, and nonlinear least squares is Dennis and Schnabel [5, 1983].

A classical reference on linear optimization is Dantzig [4, 1965]. Wright [19, 1996] give an excellent introduction to theory and implementation of primal-dual interior-point methods for linear programming. Good textbooks on optimization are Gill, Murray and Wright [8, 1981], and Fletcher [7, 1987]. A very useful guide to software for large scale optimization is Moré and Wright [14, 1993].

References

[1] E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction.*

Vol. 13 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1990.

[2] R. P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[3] Y. Censor and S. A. Zenios. *Parallel Optimization. Theory, Algorithms, and Applications* Oxford University Press, New York, Oxford, 1997.

[4] G. B. Dantzig. *Linear Programming and Extensions,* 2nd. ed. Princeton University Press, Princeton, NJ, 1965.

[5] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Classics in Applied Mathematics 16. SIAM, Philadelphia, PA, 1996.

[6] N. R. Draper and H. Smith. *Applied Regression Analysis,* 2nd. ed. John Wiley, New York, NY, 1981.

[7] R. Fletcher. *Practical Methods of Optimization,* 2nd ed. John Wiley, New York, NY, 1987.

[8] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London, UK, 1981.

[9] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization. Vol. 1.* Addison-Wesley, London, 1991.

[10] A. S. Householder. *The Numerical Treatment of a Single Nonlinear Equation.* Mc-Graw-Hill, New York, NY, 1970.

[11] H. P. Künzi, H. G. Tzschach, and C. A. Zehnder. *Numerical Methods of Mathematical Optimization.* Academic Press, New York, 1971.

[12] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming.* Addison-Wesley, Reading, MA, 1973.

[13] D. G. Luenberger. *Introduction to Dynamic Systems. Theory, Models & Applications.* John Wiley, New York, NY, 1979.

[14] J. J. Moré and S. J. Wright. *Optimization Software Guide.* SIAM, Philadelphia, 1993.

[15] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[16] A. M. Ostrowski. *Solution of Equations in Euclidian and Banach Spaces.* Academic Press, New York, 1973.

[17] P. Rabinowitz. *Numerical Methods for Non-Linear Algebraic Equations.* Gordon and Breach, London, UK, 1970.

[18] J. F. Traub. *Iterative Methods for the Solution of Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1964.

[19] S. J. Wright. *Primal-Dual Interior Point Methods.* SIAM, Philadelphia, PA, 1997.

15.1.5. Interpolation, Approximation, and Quadrature The literature on approximation theorey is extensive. An excellent introduction to the mathematical theory can be found in Cheney [5, 1966]. Watson [22, 1980] also give practical aspects. de Boor [2, 1978] gives a beautiful theoretical and practical treatise on splines and their applications, complete with software.

For a comprehensive treatment on numerical quadrature see Davis and Rabinowitz [7, 1984].

References

[1] J. H. Ahlberg, E. N. Nilsson, and J. L. Walsh. *The Theory of Splines and Their Applications.* Academic Press, New York, NY, 1967.
[2] C. de Boor. *A Practical Guide to Splines.* Springer-Verlag, New York, NY, 1978.
[3] W. L. Briggs and V. E. Henson. *The DFT. An Owner's Manual for the Discrete Fourier Transform.* SIAM, Philadelphia, PA, 1995.
[4] O. Brigham. *The Fast Fourier Transform.* Prentice-Hall, Englewood Cliffs, NJ, 1974.
[5] E. W. Cheney. *Introduction to Approximation Theory.* McGraw-Hill, New York, NY, 1966.
[6] P. J. Davis. *Interpolation and Approximation.* Blaisdell, New York, NY, 1963.
[7] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration. 2nd. ed.* Academic Press, 1984.
[8] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide* Academic Press, New York, 1988.
[9] P. Henrici. Fast Fourier methods in computational complex analysis. *SIAM Review*, 21:481–527, 1979.
[10] P. Henrici. *Applied Computational Complex Analysis. Vol. 1–3.* Wiley Classic Series, New York, NY, 1988–1993.
[11] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting. An Introduction.* Academic Press, 1986.
[12] J. Lyness. Aug2—integration over a triangle. Tech. Report ANL/MCS–TM–13, Argonne National Laboratory, Argonne, IL, 1983.
[13] J. C. Mason and M. G. Cox, editors. *Algorithms for Approximations: 2.* Chapman & Hall, 1989.
[14] R. Piessens, E. de Doncker-Kapenga, C. W. Ueberhuber, and D. K. Kahaner. *QUADPACK, A Subroutine Package for Automatic Integration.* Springer-Verlag, Berlin, 1983.
[15] M. J. D. Powell. *Approximation Theory and Methods.* Cambridge University Press, Cambridge, UK, 1981.
[16] J. R. Rice. *The Approximation of Functions. Vol. 1: Linear Theory.* Addison-Wesley, Reading, MA, 1964.
[17] J. R. Rice. *The Approximation of Functions. Vol. 2: Nonlinear and Multivariate Theory.* Addison-Wesley, Reading, MA, 1969.
[18] T. J. Rivlin. *An Introduction to the Approximation of Functions.* Dover, New York, NY, 1981.
[19] L. L. Schumaker. *Spline Functions: Basic Theory.* John Wiley, New York, NY, 1981.
[20] A. Stroud. *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Englewood Cliffs, NJ, 1972.
[21] C. Van Loan. *Computational Framework of the Fast Fourier Transform.* SIAM, Philadelphia, PA, 1992.
[22] G. A. Watson. *Approximation Theory and Numerical Methods.* John Wiley, Chichester, 1980.
[23] H. J. Weaver. *Applications of Discrete and Continuous Fourier Analysis.* Wiley-Interscience, New York, NY, 1983.

15.1.6. Ordinary Differential Equations An outstanding and up-to-date presentation on the numerical methods for *ordinary* differential equations is given by Hairer, Nørsett, and Wanner [8, 1993]. Butcher [3, 1986] gives a

treatment specialized on Runge-Kutta methods, while methods for stiff and differential-algebraic problems are treated in Hairer and Wanner [9, 1991]. A more elementary but well written modern textbook is Lambert [14, 1991]. Ascher and Petzold [2, 1998] give a unified treatment of intial-value and boundary-value problems in ODE as well as differential-algebraic equations.

Numerical methods for solving boundary value problems for ordinary differential equations can be found in Keller [13, 1976], and in Ascher, Mattheij and Russel [1, 1988].

References

[1] U. M. Ascher, R. M. M. Mattheij, and R. D. Russel. *Solution of Boundary Value Problems for Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[2] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations.* SIAM, Philadelphia, PA, 1998.

[3] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods.* John Wiley, New York, NY, 1986.

[4] E. A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations.* McGraw-Hill, New York, 1955.

[5] G. Dahlquist. 33 years of numerical instability, Part I. *BIT*, 25:188–204, 1985.

[6] L. Fox and D. F. Mayers. *Numerical Solution of Ordinary Differential Equations.* Chapman & Hall, 1987.

[7] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1971.

[8] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems. 2nd. ed.* Springer-Verlag, Berlin, 1993.

[9] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential Algebraic Problems.* Springer-Verlag, Berlin, 1991.

[10] G. Hall and J. M. Watt, editors. *Modern Numerical Methods for Ordinary Differential Equations.* Clarendon and Oxford University Press, London, 1976.

[11] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1962.

[12] A. Iserles and S. P. Nørsett. *Order Stars. Theory and Applications.* Chapman & Hall, 1991.

[13] H. B. Keller. *Numerical Methods for Two-Point Boundary Value Problems.* SIAM, Philadelphia, PA, 1976.

[14] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems. The Initial Value Problem.* John Wiley, New York, NY, 1991.

[15] L. Lapidus and J. Seinfeld. *Numerical Solution of Ordinary Differential Equations.* Academic Press, New York, 1971.

[16] H. J. Stetter. *Analysis of Discretization Methods for Ordinary Differential Equations.* Springer-Verlag, Berlin, 1973.

15.1.7. Partial Differential and Integral Equations An elementary and popular introduction to numerical solution of *partial* differential equations is given in Morton and Mayers [21, 1994]. Finite difference methods are also treated in Mitchell and Griffiths [19, 1980]. [24, 1967] and Forsythe and Wasow [10, 1960]. A good introduction to finite element methods is given by Johnson [16, 1987].

Iterative methods for the solution of the resulting linear systems are treated in Axelsson and Barker [2, 1984].

For a good introduction to multigrid methods see Briggs [6, 1987]. A more complete treatment is given in Hackbusch [13, 1985] and McCormick [18, 1987]. Good surveys of the "state of the art" of domain decomposition methods are to be found in proceedings edited by Glowinski et al. [11, 1988] and Chan et al. [7, 1989]. Time dependent problems are treated in Gustafsson, Kreiss and Oliger [12, 1996]

Comprehensive treatments of computational methods for solving integral equations are given in Baker [3, 1977] and in Delves and Mohamed [9, 1985].

References

[1] W. F Ames. *Numerical Methods for Partial Differential Equations. 2nd. ed.* Academic Press, New York, 1977.

[2] O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problems. Theory and Computation.* Academic Press, 1984.

[3] C. T. H. Baker. *The Numerical Treatment of Integral Equations.* Clarendon Press, Oxford, 1977.

[4] K. J. Bathe and E. L. Wilson. *Numerical Methods in Finite Element Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

[5] G. Birkhoff and R. E. Lynch. *Numerical Solution of Elliptic Problems.* SIAM, Philadelphia, 1984.

[6] W. L. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, 1987.

[7] T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund, editors. *Proceedings of the Second International Symposium on Domain Decomposition Methods.* SIAM, Philadelphia, 1989.

[8] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems.* North Holland, 1978.

[9] L. M. Delves and J. L. Mohamed. *Computational Methods for Integral Equations.* Cambridge University Press, Cambridge, UK, 1985.

[10] G. E. Forsythe and W .R. Wasow. *Finite Difference Methods for Partial Differential Equations.* John Wiley, New York, NY, 1960.

[11] R. Glowinski, G. H. Golub, G. Meurant, and J. Periaux, editors. *Proceedings of the First International Symposium on Domain Decomposition Methods.* SIAM, Philadelphia, PA, 1988.

[12] B. Gustafsson, H.-O. Kreiss, and J. Oliger. *Time Dependent Problems and Difference Methods.* John Wiley, New York, NY, 1996.

[13] W. Hackbusch. *Multi-Grid Methods and Applications.* Springer-Verlag, Berlin-Heidelberg-New York, 1985.

[14] W. Hackbusch. *Elliptic Differential Equations. Theory and Numerical Treatment.* Springer-Verlag, Berlin, 1992.

[15] W. Hackbusch and U. Trottenberg, editors. *Multigrid Methods. Lecture Notes in Mathematics, Vol. 960.* Springer-Verlag, Berlin, 1982.

[16] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Studentlitteratur, Lund, Sweden, 1987.

[17] L. Lapidus and G. F. Pinder. *Numerical Solution of Partial Differential Equations in Science and Engineering.* John Wiley, New York, NY, 1982.

[18] S. McCormick. *Multigrid Methods.* SIAM, Philadelphia, PA, 1987.

[19] A. R. Mitchell and D. F. Griffiths. *The Finite Difference Method in Partial Differential Equations.* John Wiley, New York, NY, 1980.
[20] A. R. Mitchell and R. Wait. *The Finite Element Method in Partial Differential Equations.* John Wiley, New York, NY, 1977.
[21] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations.* Cambridge University Press, 1994.
[22] J. N. Reddy. *An Introduction to the Finite Element Method.* McGraw-Hill, New York, 1984.
[23] J. R. Rice and R. F. Boisvert. *Solving Elliptic Problems Using ELLPACK.* Springer-Verlag, Berlin, 1984.
[24] R. D. Richtmyer and K. W. Morton. *Difference Methods for Initial Value Problems. 2nd. ed.* John Wiley-Interscience, New York, NY, 1967.
[25] H. R. Schwarz. *Finite Element Methods.* Academic Press, New York, 1988.
[26] P. N. Schwarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Review*, 19:490–501, 1977.
[27] G. D. Smith. *Numerical Solution of Partial Difference Equations. 3rd. ed.* Clarendon Press, Oxford, 1985.
[28] G. Strang and G. Fix. *An Analysis of the Finite Element Method.* Prentice-Hall, Englewood Cliffs, NJ, 1973.
[29] O. C. Zienkiewicz. *The Finite Element Method. 3rd. ed.* McGraw-Hill, New York, 1977.

15.1.8. Journals

ACM Transaction on Mathematical Software (1975–), Association for Computing Machinery, New York.

Acta Numerica (1992–), Cambridge University Press.

Applied Numerical Mathematics (1983–), IMACS, North-Holland

Advances in Computational Mathematics (1993–), J. C. Balzer AG, Basel.

BIT Numerical Mathematics, (1961–), Copenhagen, Denmark.

Calcolo (1998–), Springer-Verlag, Berlin.

Journal of Computational Mathematics (1990–), Science Press, Beijing.

Computing. Archives for Informatics and Numerical Computing (1965–), Springer-Verlag, Wien and New York.

IMPACT of Computing in Science and Engineering (1988–), Academic Press

Journal of Approximation Theory (1968–), Academic Press, New York.

Journal of Computational and Applied Mathematics (1982–), North-Holland

Journal of Computational Mathematics (1982–), Science, Beijing China & Utrecht, Holland.

Journal of Computational Physics (1966–), Academic Press, New York.

Journal of Mathematical Analysis and Applications (1945–), Academic Press, New York.

IMA Journal of Numerical Analysis (1981–), Institute of Mathematics and Its

Applications, Oxford University Press.

Linear Algebra and Its Applications (1968–),Elsevier, New York.

Mathematics of Computation (1960–), previously called:

Mathematical Tables and Other Aids to Computation (1943–1959), American Mathematical Society, Providence, R.I.

Mathematical Programming (1971–), Elsevier/North-Holland, Amsterdam.

Numerical Algorithms (1991–), J. C. Balzer AG, Basel.

Numerical Linear Algebra with Applications, Wiley-Interscience

Numerical Methods in Engineering (1969–), Wiley-Interscience, London.

Numerische Mathematik (1959–), Springer-Verlag, Berlin.

Russian Journal of Numerical Analysis and Mathematical Modelling (1993–), VSP, Zeist.

SIAM Journal on Applied Mathematics (1941–), SIAM, Philadelphia.

SIAM Journal on Matrix Analysis and Applications (1980–), SIAM, Philadelphia.

SIAM Journal on Numerical Analysis (1964–), SIAM, Philadelphia.

SIAM Journal on Optimization (1991–), SIAM, Philadelphia.

SIAM Review (1958–), SIAM, Philadelphia.

SIAM Journal on Scientific Computing (1992–), SIAM, Philadelphia.

Surveys on Mathematics in Industry, Springer-Verlag, Berlin

15.2. Guide to Software

15.2.1. Linear Algebra Packages LINPACK and EISPACK software given in Dongarra et al. [10, 1979] and Smith et al. [40, 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.

15.2.2. Interactive Mathematical Software 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.

15.2.3. Commercial Subroutine Libraries A subroutine library is a unified set of supported computer programs. The two largest suppliers of scientific subroutine libraries are NAG and IMSL.

The Numerical Algorithms Group (NAG)[1] arose from a meeting in Nottingham in 1970 when representatives from several British universities agreed to collaborate in the development of a numerical algorithms library. Originally the library was only intended for the ICL 1900 machine rangef, but it was soon extended to most commercial computers. The library comprises Fortran and C subroutines, as well as symbolic system solvers.

A02 Complex Arithmetic

C02 Zeros of Polynomials

C05 Roots of one or more Transcendental Equations

C06 Summation of Series

D01 Quadrature

D02 Ordinary Differential Equations

D03 Partial Differential Equations

D04 Numerical Differentiation

D05 Integral Equations

E01 Interpolation

E02 Curve and Surface Fitting

E04 Minimizing or Maximizing a Function

F01 Matrix Operations, Including Inversion

F02 Eigenvalues and Eigenvectors

F03 Determinants

F04 Simultaneous Linear Equations

F05 Orthogonalization

F06 Linear Algebra Support Routines

The software company IMSL was founded in 1970 with the purpose of developing and maintaining an evolving set of analytical computer routines. Currently the IMSL Library is a comprehensive set of mathematical and statistical FORTRAN subroutines, available on most major computers. The subroutines are input/outfree routines designed to be called from application programs. The library subroutines are arranged in 17 chapters:

1. Analysis of Variance.

2. Basic Statistics

3. Categorized Data Analysis

4. Differential Equations; Quadrature; Differentiation

5. Eigensystem Analysis

[1] NAG Ltd, Wilkinson House, Jordan Hill Road, Oxford, UK OX2 8DR

6. Forecasting; Econometrics; Time Series; Transforms
7. Generation and Testing of Random Numbers
8. Interpolation; Approximation; Smoothing
9. Linear Algebraic Equations
10. Mathematical and Statistical Special Functions
11. Nonparametric Statistics
12. Observation Structure; Multivariate Statistics
13. Regression Analysis
14. Sampling
15. Utility Functions
16. Vector, Matrix Arithmetic
17. Zero and Extrema; Linear Programming

15.2.4. Software for Public Use 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

BLAS blas (level 1, 2 and 3) and machine constants

EISPACK A collection of Fortran subroutines that compute the eigenvalues and eigenvectors of nine classes of matrices. The package can determine the eigensystems of complex general, complex Hermitian, real general, real symmetric, real symmetric band, real symmetric tridiagonal, special real tridiagonal, generalized real, and generalized real symmetric matrices. In addition, there are two routines which use the singular value decomposition to solve certain least squares problems. Developed by the NATS Project at Argonne National Laboratory. (d.p. refer to eispack, s.p. refer to seispack)

FFTPACK A package of Fortran subprograms for the Fast Fourier Transform of periodic and other symmetric sequences This package consists of programs which perform Fast Fourier Transforms for both complex and real periodic sequences and certian other symmetric sequences. Developed by Paul Swarztrauber, at NCAR.

VFFTPK a vectorized version of FFTPACK for multiple sequences. By Sweet, Lindgren, and Boisvert.

FISHPACK A package of Fortran subprograms providing finite difference approximations for elliptic boundary value problems. Developed by Paul Swarztrauber and Roland Sweet.

FN Wayne Fullerton's special function library. (single and double)

GO Golden Oldies: routines that have been widely used, but aren't available through the standard libraries. Nominations welcome!

HARWELL Sparse matrix routine MA28 from the Harwell library. from Iain Duff

LAPACK lapack is a library of Fortran 77 subroutines for solving the most common problems in numerical linear algebra: linear equations, linear least squares problems, eigenvalue problems, and singular value problems. It has been designed to be efficient on a wide range of modern high-performance computers. by Ed Anderson, Z. Bai, Chris Bischof, Jim Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, Susan Ostrouchov, and Danny Sorensen

INPACK A collection of Fortran subroutines that analyze and solve linear equations and linear least squares problems. The package solves linear systems whose matrices are general, banded, symmetric indefinite, symmetric positive definite, triangular, and tridiagonal square. In addition, the package computes the QR and singular value decompositions of rectangular matrices and applies them to least squares problems. Developed by Jack Dongarra, Jim Bunch, Cleve Moler and Pete Stewart. (all precisions contained here)

PPPACK Subroutines from: Carl de Boor, A Practical Guide to Splines, Springer-Verlag. This is an old version, from around the time the book was published. Some calling sequences differ slightly from those in the book. We will install newer version as soon as we can.

TOMS Collected algorithms of the ACM. When requesting a specific item, please refer to the Algorithm number.

In contrast to the above libraries, the following are collections of codes from a variety of sources. Most are excellent, but you should exercise caution. We include research codes that we haven't tested and codes that may not be state-of-the-art but useful for comparisons. The following list is chronological, not by merit:

A approximation algorithms loess: multivariate smoothing of scattered data; Cleveland & Grosse
cdfcor: multivariate uniform rational fitting of scattered data; Kaufman & Taylor

AMOS Bessel functions of complex arguments and nonnegative order.

BIHAR Biharmonic solver in rectangular geometry and polar coordinates. Petter Bjorstad, University of Bergen

BMP Brent's multiple precision package

C miscellaneous codes written in C. Not all C software is in this "miscellaneous" library. If it clearly fits into domain specific library, it is assigned there.

C++ miscellaneous codes in the C++ language.

ESCACH numerical linear algebra, dense and sparse, with permutations, error handling, in C (double prec.)

FORMAL contains routines to solve the "parameter problem" associated with the Schwarz-Christoffel mapping. Includes: SCPACK (polygons with straight sides) from Nick Trefethen. CAP (circular arc polygons) from Petter Bjorstad and Eric Grosse. gearlike domains by Kent Pearce. CONFPACK (Symm's integral equation) by Hough and Gutknecht.

CONTIN methods for continuation and limit points, notably version 6.1 of PITCON, written by Werner Rheinboldt and John Burkardt, University of Pittsburgh.

CRPC Software available form the NSF Science and Technology Center for Research in Parallel Computation.

DDSV programs from: Linear Algebra Computations on Vector and Parallel Computers, by Jack Dongarra, Iain Duff, Danny Sorensen, and Henk Van der Vorst.

DIERCKX a package of spline fitting routines for various kinds of data and geometries. Written by: Professor Paul Dierckx, Dept. Computer Science, K. U. Leuven, Celestijnenlaan 200A, B-3030 Heverlee, Belgium.

ELEFUNT is a collection of transportable Fortran programs for testing the elementary function programs provided with Fortran compilers. The programs are described in detail in the book "Software Manual for the Elementary Functions" by W. J. Cody and W. Waite, Prentice Hall, 1980.

FITPACK a package for splines under tension. (an early version) For a current copy and for other routines, contact: Pleasant Valley Software, 8603 Altus Cove, Austin TX 78759, USA

FMM routines from the book Computer Methods for Mathematical Computations, by Forsythe, Malcolm, and Moler. Developed by George Forsythe, Mike Malcolm, and Cleve Moler. (d.p. refer to fmm, s.p. refer to sfmm)

FORTRAN contains tools specific to Fortran. At present, it contains a single-double precision converter, and a static debugger.

FP floating point arithmetic

GCV software for Generalized Cross Validation from: O'Sullivan, Woltring (univariate spline smoothing), Bates, Lindstrom, Wahba and Yandell (multivariate thin plate spline smoothing and ridge regression), Gu (multiple smoothing parameters), Fessler (vector smoothing spline)

HOMPACK is a suite of FORTRAN 77 subroutines for solving nonlinear systems of equations by homotopy methods. There are subroutines for fixed point, zero finding, and general homotopy curve tracking problems, utilizing both dense and sparse Jacobian matrices, and implementing three different algorithms: ODE-based, normal flow, and augmented Jacobian.

ITPACK Iterative Linear System Solver based on a number of methods: Jacobi method, SOR, SSOR with conjugate gradient acceleration or with Chebyshev (semi-iteration - SI) acceleration. Developed by Young and Kincaid and the group at U of Texas.

JAKEF is a precompiler that analyses a given Fortran77 source code for the evaluation of a scalar or vector function and then generates an expanded Fortran subroutine that simultaneously evaluates the gradient or Jacobian respectively. For scalar functions the ratio between the run-time of the resulting gradient routine and that of the original evaluation routine is never greater than a fixed bound of about five. The storage requirement may be considerable as it is also proportional to the run-time of the original routine. Since no differencing is done the partial derivative values obtained are exact up to round-off errors. A. Griewank, Argonne National Laboratory, griewank@mcs.anl.gov, 12/1/88.

LANZ large Sparse Symmetric Generalized Eigenproblem Mark T. Jones, Argonne National Laboratory Merrell L. Patrick, Duke University and NSF

ANCZOS procedures computing a few eigenvalues/eigenvectors of a large (sparse) real symmetric and Hermitian matrices. Singular values and vectors of rectangular matrices Jane Cullum and Ralph Willoughby, IBM Yorktown.

LASO A competing Lanczos package. David Scott.

LP Linear Programming - At present, this consists of one subdirectory, data: a set of test problems in MPS format, maintained by David Gay. For more information, try a request of the form send index for lp/data

ADPACK is a a compact package for solving systems of linear equations using multigrid or aggregation-disaggregation methods. Imbedded in the algorithms are implementations for sparse Gaussian elimination and symmetric Gauss-Seidel (unaccelerated or accelerated by conjugate gradients or Orthomin(1)). This package is particularly useful for solving problems which arise from discretizing partial differential equations, regardless of whether finite differences, finite elements, or finite volumes are used. It was written by Craig Douglas. installed 2/90

MATLAB software from the MATLAB Users Group. Christian Bischof bischof@mcs.anl.gov

INPACK A package of Fortran programs for the solution of systems of nonlinear equations and nonlinear least squares problems. Five algorithmic paths each include a core subroutine and an easy-to-use driver. The algorithms proceed either from an analytic specification of the Jacobian matrix or directly from the problem functions. The paths include facilities for systems of equations with a banded Jacobian matrix, for least squares problems with a large amount of data, and for checking the consistency of the Jacobian matrix with the functions. Developed by Jorge More', Burt Garbow, and Ken Hillstrom at Argonne National Laboratory. (d.p. refer to minpack, s.p. refer to sminpack)

MISC Contains various pieces of software collected over time and: the source code for the netlib processor itself; the paper describing netlib and its implementation; the abstracts list maintained by Richard Bartels.

NAPACK A collection of Fortran subroutines to solve linear systems, to estimate the condition number or the norm of a matrix, to compute determinants, to multiply a matrix by a vector, to invert a matrix, to solve least squares problems, to perform unconstrained minimization, to compute eigenvalues, eigenvectors, the singular value decomposition, or the QR decomposition. The package has special routines for general, band, symmetric, indefinite, tridiagonal, upper Hessenberg, and circulant matrices. Code author: Bill Hager, Mathematics Department, University of Florida, Gainesville, FL 32611, e-mail: hager@math.ufl.edu. Related book: Applied Numerical Linear Algebra, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

NUMERALGO Algorithms from the journal "Numerical Algorithms".

ODE various initial and boundary value ordinary differential equation solvers: colsys, dverk, rkf45, ode A subset of these in single precision is in the library sode.

ODEPACK The ODE package from Hindmarch and others. This is the double precision verison; to get sp refer to sodepack. Alan Hindmarch, Lawrence Livermore National Laboratory

ODRPACK Orthogonal Distance Regression by Boggs, Byrd, Rogers, and Schnabel. A portable collection of Fortran subprograms for fitting a model to data. It is designed primarily for instances when the independent as well as the dependent variables have significant errors, implementing a highly efficient algorithm for solving the weighted orthogonal distance regression problem, i.e., for minimizing the sum of the squares of the weighted orthogonal distances between each data point and the curve described by the model equation.

OPT miscellaneous optimization software. Contains Brent's praxis.

PARANOIA is a rather large program, devised by Prof. Kahan of Berkeley, to explore the floating point system on your computer.

PASCAL At present, codes from J. C. Nash, Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation, Second Edition Adam Hilger: Bristol & American Institute of Physics: New York, 1990

PCHIP is a fortran package for piecewise cubic hermite inter- polation of data. It features software to produce a monotone and "visually pleasing" interpolant to monotone data. Fred N. Fritsch, Lawrence Livermore National Laboratory

ADPACK multigrid by Craig Douglas: a compact package for solving systems of linear equations using multigrid or aggregation-disaggregation methods. Imbedded in the algorithms are implementations for sparse Gaussian elimination and symmetric Gauss-Seidel (unaccelerated or accelerated by conjugate gradients or Orthomin(1)). This package is particularly useful for solving problems which arise from discretizing partial differential equations, regardless of whether finite differences, finite elements, or finite volumes are used.

PLTMG elliptic partial differential equations in general regions of the plane It features adaptive local mesh refinement, multigrid iteration, and a pseudo-arclength continuation option for parameter dependencies. The package includes an initial mesh generator and several graphics packages. by Randy Bank, reference: PLTMG User's Guide, SIAM publications

YHEDRA a database of angles, vertex locations, and so on for over a hundred geometric solids, compiled by Andrew Hume.

PORT The public subset of the PORT library. Includes the latest version of Gay's NL2SOL nonlinear least squares. The rest of the PORT3 library is available by license from AT&T.

SEARCH miscellaneous software from Computing Science Research at AT&T Bell Laboratories in Murray Hill, NJ

ADPACK A package for numerical computation of definite univariate integrals. Developed by Piessens, Robert(Appl. Math. and Progr. Div.- K.U.Leuven) de Donker, Elise(Appl. Math. and Progr. Div.- K.U.Leuven Kahaner, David(National Bureau of Standards) (slatec version)

CIPORT a portable FORTRAN emulation (by M.J. McBride and S.H. Lamson) of CRAY SCILIB, a library of scientific applications subprograms developed by Cray Research, Inc.

SLAP iterative symmetric and non-symmetric linear system solution Sparse Linear Algebra Package. Included in this package are core routines to do Iterative Refinement iteration, Preconditioned Conjugate Gradient iteration, Preconditioned Conjugate Gradient iteration on the Normal Equations, Preconditioned BiConjugate Gradient iteration, Preconditioned BiConjugate Gradient Squared iteration, Orthomin iteration and Generalized Minimum Residual iteration. Core routines require the user to supply "MATVEC" (Matrix Vector Multiply) and "MSOLVE" (Preconditioning) routines. by Mark K. Seager & Anne Greenbaum

SLATEC comprehensive software library containing over 1400 general purpose mathematical and statistical routines written in Fortran 77.

SPARSPAK Subroutines from the book "Computer Solution of Large Sparse Positive Definite Systems" by George and Liu, Prentice Hall 1981.

SPARSE A library of subroutines written in C that solve large sparse systems of linear equations using LU factorization. The package is able to handle arbitrary real and complex square matrix equations. Besides being able to solve linear systems, it is solves transposed systems, find determinants, multiplies a vector by a matrix, and estimate errors due to ill-conditioning in the system of equations and instability in the computations. Sparse does not require or assume symmetry and is able to perform numerical pivoting (either diagonal or complete) to avoid unnecessary error in the solution. Sparse also has an optional interface that allow it to be called from FORTRAN programs. Ken Kundert, Alberto Sangiovanni-Vincentelli. (sparse@ic.berkeley.edu)

SEQUENT software from the Sequent Users Group. Jack Dongarra 9/88

PARSE-BLAS an extension to the set of Basic Linear Algebra Subprograms. The extension is targeted at sparse vector operations, with the goal of providing efficient, but portable, implementations of algorithms for high performance computers, by Dave Dodson, convex!dodson@a.cs.uiuc.edu

SPECFUN is an incomplete, but growing, collection of transportable Fortran programs for special functions, and of accompanying test programs similar in concept to those in ELEFUNT. W.J. Cody, Argonne National Laboratory

SVDPACK singular values and singular vectors of large sparse matrices. Mike Berry, University of Tennessee.

TOEPLITZ A package of Fortran subprograms for the solution of systems of linear equations with coefficient matrices of Toeplitz or circulant form and for orthogonal factorization of column- circulant matrices. Developed by Burt Garbow at Argonne National Laboratory, as a culmination of Soviet-American collaborative effort. (d.p. refer to toeplitz, s.p. refer to stoeplitz)

NCON/DATA test problems: unconstrained optimization, nonlinear least squares. Problems from More, Garbow, and Hillstrom; Fraley, matrix square root; Hanson, Salane; McKeown; De Villiers and Glasser; Dennis, Gay, and Vu. Collected by Chris Fraley.

VANHUFFEL The TLS problem assumes an overdetermined set of linear equations AX = B, where both the data matrix A as well as the observation matrix B are inaccurate. The subroutine PTLS solves the Total Least Squares (TLS) problem by using a Partial Singular Value Decomposition (PSVD), hereby improving considerably the computational efficiency with respect to the classical TLS algorithm. Author Sabine Van Huffel, ESAT Laboratory, KU Leuven.

ORONOI Algorithms for Voronoi regions and Delaunay triangulations. Currently contains Fortune's 2d sweepline method.