Chapter 4

SOLVING SYSTEMS OF LINEAR EQUATIONS

The principal objective of this chapter is to discuss the numerical aspects of solving systems of linear equations having the form

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n = b_3 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{1}$$

This is a system of n equations in the n unknowns $x_1, x_2, \ldots, x_n$. The elements a_{ij} and b_i are assumed to be prescribed real numbers.

Matrices are useful devices for representing systems of equations. Thus, System (1) can be written as

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Then we can denote these matrices by A, x, and b, so that the equation becomes simply

$$Ax = b$$

In this chapter, we shall construct a general-purpose algorithm for solving this problem. Then we shall analyze the errors that are associated with the computer solution and study methods for controlling and reducing them. Finally, we give an introduction to the important topic of iterative algorithms for this problem.

4.1 Matrix Algebra

This section serves as a review for the basic concepts of matrix theory. Further material on this topic will be introduced and discussed when needed in the subsequent sections.

Basic Concepts

A matrix is a rectangular array of numbers such as

$$\begin{bmatrix} 3.0 & 1.1 & -0.12 \\ 6.2 & 0.0 & 0.15 \\ 0.6 & -4.0 & 1.3 \\ 9.3 & 2.1 & 8.2 \end{bmatrix} \qquad \begin{bmatrix} 3 & 6 & \frac{11}{7} & -17 \end{bmatrix} \qquad \begin{bmatrix} 3.2 \\ -4.7 \\ 0.11 \end{bmatrix}$$

These are, respectively, a 4×3 matrix, a 1×4 matrix, and a 3×1 matrix. In describing the dimensions of a matrix, the number of **rows** (horizontal lines) is given first, and the number of **columns** (vertical lines) is given second. A $1 \times n$ matrix is also called a **row vector**. An $m \times 1$ matrix is called a **column vector** or just a **vector**.

If A is a matrix, the notations a_{ij}, $(A)_{ij}$, or $A(i,j)$ are used to denote the element at the intersection of the ith row and jth column. For example, if A denotes the first of the matrices displayed above, then $A(3,2) = -4.0$. The **transpose** of a matrix is denoted by A^T, and is the matrix defined by $(A^T)_{ij} = a_{ji}$. Using the same example to illustrate the transpose, we have

$$A^T = \begin{bmatrix} 3.0 & 6.2 & 0.6 & 9.3 \\ 1.1 & 0.0 & -4.0 & 2.1 \\ -0.12 & 0.15 & 1.3 & 8.2 \end{bmatrix}$$

If a matrix A has the property $A^T = A$, we say that A is **symmetric**.

If A is a matrix and λ is a scalar (that is, a real number in this context) then λA is defined by $(\lambda A)_{ij} = \lambda a_{ij}$. If $A = (a_{ij})$ and $B = (b_{ij})$ are $m \times n$ matrices then $A + B$ is defined by $(A + B)_{ij} = a_{ij} + b_{ij}$. Of course, $-A$ means $(-1)A$. If A is an $m \times p$ matrix and B is a $p \times n$ matrix, then AB is an $m \times n$ matrix defined by

$$(AB)_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} \qquad (1 \leqq i \leqq m,\ 1 \leqq j \leqq n)$$

Here are some examples of the algebraic operations:

$$\begin{bmatrix} 1 & 3 \\ 2 & -1 \\ 4 & -4 \end{bmatrix} + \begin{bmatrix} 6 & 0 \\ 3 & -7 \\ 8 & 2 \end{bmatrix} = \begin{bmatrix} 7 & 3 \\ 5 & -8 \\ 12 & -2 \end{bmatrix}$$

$$3 \begin{bmatrix} 1 & 3 \\ 2 & -1 \\ 4 & -4 \end{bmatrix} = \begin{bmatrix} 3 & 9 \\ 6 & -3 \\ 12 & -12 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & 3 \\ 1 & 5 & -6 \\ 2 & 1 & 5 \\ 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -5 & 4 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} -3 & 13 \\ -24 & 2 \\ -3 & 19 \\ -5 & -2 \end{bmatrix}$$

In dealing with systems of linear equations there is a concept of equivalence that is important. Let two systems be given, each consisting of n equations with n unknowns:

$$Ax = b \qquad Bx = d$$

If the two systems have precisely the same solutions, we call them **equivalent systems**. Thus, to solve a system of equations, we can instead solve any equivalent system; no solutions are lost and no new ones appear. This simple idea is at the heart of our numerical procedures. Given a system of equations to be solved, we transform it by certain elementary operations into a simpler *equivalent* system which we then solve instead.

The **elementary operations** alluded to in the previous paragraph are of the following three types. (Here $\mathcal{E}_i$ denotes the ith equation in the system.)

(i) Interchanging two equations in the system: $\mathcal{E}_i \leftrightarrow \mathcal{E}_j$

(ii) Multiplying an equation by a nonzero number: $\lambda\mathcal{E}_i \rightarrow \mathcal{E}_i$

(iii) Adding to an equation a multiple of some other equation: $\mathcal{E}_i + \lambda\mathcal{E}_j \rightarrow \mathcal{E}_i$

THEOREM 1 *If one system of equations is obtained from another by a finite sequence of elementary operations, then the two systems are equivalent.*

Proof It suffices to consider the effect of a single application of each elementary operation. Suppose that an elementary operation transforms the system $Ax = b$ into the system $Bx = d$. If the operation is of type (i), then the two systems consist of precisely the same equations although written in a different order. Clearly, if x solves the first system then it solves the second and vice versa. If the operation is of type (ii), then suppose that the ith equation has been multiplied by a scalar λ with $\lambda \neq 0$. The ith and jth equations in $Ax = b$ are

$$a_{i1}x_1 + \cdots + a_{in}x_n = b_i \tag{2}$$

and

$$a_{j1}x_1 + \cdots + a_{jn}x_n = b_j \tag{3}$$

and the ith equation in $Bx = d$ is

$$\lambda a_{i1}x_1 + \cdots + \lambda a_{in}x_n = \lambda b_i \tag{4}$$

Any vector x that satisfies Equation (2) satisfies Equation (4) and vice versa, because $\lambda \neq 0$. Finally, suppose that the operation is of type (iii). Assume that λ times the jth equation has been added to the ith. Then the ith equation in $Bx = d$ is

$$(a_{i1} + \lambda a_{j1})x_1 + \cdots + (a_{in} + \lambda a_{jn})x_n = b_i + \lambda b_j \tag{5}$$

Observe particularly that the jth equation in the system $Bx = d$ has *not* been changed. If $Ax = b$, then Equations (2) and (3) are true. Hence, (5) is true. Thus, $Bx = d$. On the other hand, if we suppose that x solves $Bx = d$, then Equations (5) and (3) are true. If λ times Equation (3) is *subtracted* from Equation (5) the result is Equation (2). Hence, $Ax = b$. ∎

Matrix Properties

The $n \times n$ matrix

$$I = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

is called an **identity matrix**. It has the property that $IA = A = AI$ for any matrix A of size $n \times n$.

If A and B are two matrices such that $AB = I$, then we say that B is a **right inverse** of A and that A is a **left inverse** of B. For example,

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \alpha & \beta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6}$$

We see from this example that if a matrix has a right inverse, the latter is not necessarily unique. For square matrices the situation is better, as we now show.

THEOREM 2 *A square matrix can possess at most one right inverse.*

Proof Let $AB = I$, in which A, B, and I are all $n \times n$ matrices. Denote by $A^{(j)}$ the jth column of A and by $I^{(k)}$ the kth column of I. The equation $AB = I$ means that

$$\sum_{j=1}^{n} b_{jk} A^{(j)} = I^{(k)} \qquad (1 \leqq k \leqq n) \tag{7}$$

Each column of I is therefore a linear combination of the columns of A. Since the columns of I span $\mathbb{R}^n$, the same is true of the columns of A. Hence, the columns of A form a basis for $\mathbb{R}^n$, and, consequently, the coefficients b_{jk} in Equation (7) are *uniquely* determined. ∎

THEOREM 3 *If A and B are square matrices such that $AB = I$, then $BA = I$.*

Proof Let $C = BA - I + B$. Then

$$AC = ABA - AI + AB = A - A + I = I$$

Thus, C (as well as B) is a right inverse of A. By Theorem 2, $B = C$; hence, $BA = I$. ∎

It follows from Theorems 2 and 3 that if a square matrix A has a right inverse B, then B is unique and $BA = AB = I$. We then call B the **inverse** of A and say that A is **invertible** or **nonsingular**. (Of course, B is therefore invertible and A is *its* inverse.) We write $B = A^{-1}$ and $A = B^{-1}$. Here is an example:

$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

If A is invertible, then the system of equations $Ax = b$ has the solution $x = A^{-1}b$. If A^{-1} is already available, this equation provides a good method of computing x. If A^{-1} is not available, then in general A^{-1} should *not* be computed solely for the purpose of obtaining x. More efficient procedures will be developed in later sections.

The elementary operations discussed earlier can be carried out with matrix multiplications, as we now indicate. An **elementary matrix** is defined to be an $n \times n$ matrix that arises when an elementary operation is applied to the $n \times n$ identity matrix. The elementary operations, expressed in terms of the rows of a matrix A, are:

(i) The interchange of two rows in A: $A_s \leftrightarrow A_t$

(ii) Multiplying one row by a nonzero constant: $\lambda A_s \longrightarrow A_s$

(iii) Adding to one row a multiple of another: $A_s + \lambda A_t \longrightarrow A_s$

In this description, we have denoted the rows of A by subscripts, A_s, A_t, and so on. Each elementary row operation on A can be accomplished by multiplying A on the left by an elementary matrix. Here are three examples, illustrating the three types of operations:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ \lambda a_{21} & \lambda a_{22} & \lambda a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \lambda & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \lambda a_{21} + a_{31} & \lambda a_{22} + a_{32} & \lambda a_{23} + a_{33} \end{bmatrix}$$

If we wish to apply a succession of elementary row operations to A, we introduce elementary matrices $E_1, E_2, \ldots, E_m$ and then write the transformed matrix as

$$E_m E_{m-1} \cdots E_2 E_1 A$$

If a matrix is invertible, such a sequence of elementary row operations can be applied to A, reducing it to I. Thus

$$E_m E_{m-1} \cdots E_2 E_1 A = I$$

From this it follows that $A^{-1} = E_m E_{m-1} \cdots E_2 E_1$. Consequently, A^{-1} can be obtained by subjecting I to the same sequence of elementary row operations. Here is an example showing the computation of an inverse:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 3 \\ 2 & 4 & 7 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I$$

$$E_1 A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 2 & 4 & 7 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = E_1 I$$

$$E_2 E_1 A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} = E_2 E_1 I$$

$$E_3 E_2 E_1 A = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 3 & -2 & 0 \\ -1 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} = E_3 E_2 E_1 I$$

$$E_4 E_3 E_2 E_1 A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 9 & -2 & -3 \\ -1 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} = E_4 E_3 E_2 E_1 I = A^{-1}$$

Here the elementary matrices are

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$$

$$E_3 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad E_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

THEOREM 4 *For an $n \times n$ matrix A the following properties are equivalent:*

(i) *The inverse of A exists; that is, A is nonsingular.*

(ii) *The determinant of A is nonzero.*

(iii) *The rows of A form a basis for $\mathbb{R}^n$.*

(iv) *The columns of A form a basis for $\mathbb{R}^n$.*

(v) *As a map from $\mathbb{R}^n$ to $\mathbb{R}^n$, A is injective (one-to-one).*

(vi) *As a map from $\mathbb{R}^n$ to $\mathbb{R}^n$, A is surjective (onto).*

(vii) *The equation $Ax = 0$ implies $x = 0$.*

(viii) *For each $b \in \mathbb{R}^n$, there is exactly one $x \in \mathbb{R}^n$ such that $Ax = b$.*

(ix) *A is a product of elementary matrices.*

(x) *0 is not an eigenvalue of A.*

An important fundamental concept is the *positive definiteness* of a matrix. A matrix A is **positive definite** if $x^T Ax > 0$ for every nonzero vector x. For example, the matrix

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

is positive definite since

$$x^T Ax = [x_1 \quad x_2]\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = (x_1 + x_2)^2 + x_1^2 + x_2^2 > 0$$

for all x_1 and x_2 except $x_1 = x_2 = 0$. Here $x^T Ax$ is called a **quadratic form**. It follows from Problems 18–20 that when dealing with positive definiteness we can assume symmetry. Establishing the positive definiteness of a matrix using the definition is usually not an easy task since it involves an *arbitrary* $x \neq 0$. If A is positive definite and symmetric, then its eigenvalues are real and positive.

Partitioned Matrices

It is frequently convenient to partition matrices into submatrices and compute products as if the submatrices were numbers. An example of this procedure is as follows:

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 1 & -1 & 0 & 1 \end{bmatrix} \\ \begin{bmatrix} -1 & 1 \\ 0 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 2 \end{bmatrix} & \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 2 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ -2 & 1 \\ -1 & 1 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 1 & 2 & 7 \end{bmatrix} & \begin{bmatrix} 2 & 5 \end{bmatrix} \\ \begin{bmatrix} -3 & 1 & 3 \\ -3 & 3 & 2 \\ 4 & 0 & -1 \\ 2 & 3 & 2 \end{bmatrix} & \begin{bmatrix} 0 & 2 \\ -2 & 1 \\ 0 & 1 \\ 1 & 6 \end{bmatrix} \end{bmatrix}$$

If these matrices are partitioned as shown, and if the submatrices are denoted by single letters, we have a product of the form

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

One can verify that $C_{ij} = \sum_{s=1}^{2} A_{is}B_{sj}$. Thus, for example,

$$[1 \quad 2]\begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 2 \end{bmatrix} + [1 \quad -1 \quad 0 \quad 1]\begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 2 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = [1 \quad 2 \quad 7]$$

and, as shown by the computation, $C_{11} = A_{11}B_{11} + A_{12}B_{21}$.

To establish a general result concerning this technique, let A, B, and C be matrices that have been partitioned into submatrices:

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1k} \\ B_{21} & B_{22} & \cdots & B_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n1} & B_{n2} & \cdots & B_{nk} \end{bmatrix}$$

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1k} \\ C_{21} & C_{22} & \cdots & C_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ C_{m1} & C_{m2} & \cdots & C_{mk} \end{bmatrix}$$

THEOREM 5 *If each product $A_{is}B_{sj}$ can be formed and if $C_{ij} = \sum_{s=1}^{n} A_{is}B_{sj}$, then $C = AB$.*

Proof Let the dimensions of A_{ij} be $m_i \times n_j$. Let the dimensions of B_{ij} be $\widehat{m}_i \times \widehat{n}_j$. Since $A_{is}B_{sj}$ exists, we must have $n_s = \widehat{m}_s$ for all s. Then C_{ij} will have dimensions $m_i \times \widehat{n}_j$.

Now select an arbitrary element c_{ij} in the matrix C. Suppose that c_{ij} lies in the block C_{rs} and is in the pth row and qth column of C_{rs}. Then we must have

$$i = m_1 + m_2 + \cdots + m_{r-1} + p \tag{8}$$

$$j = \widehat{n}_1 + \widehat{n}_2 + \cdots + \widehat{n}_{s-1} + q \tag{9}$$

Consequently

$$\begin{aligned} c_{ij} &= (C_{rs})_{pq} \\ &= \Big(\sum_{t=1}^{n} A_{rt}B_{ts}\Big)_{pq} \\ &= \sum_{t=1}^{n} (A_{rt}B_{ts})_{pq} \\ &= \sum_{t=1}^{n} \sum_{\alpha=1}^{n_t} (A_{rt})_{p\alpha}(B_{ts})_{\alpha q} \end{aligned}$$

The elements $(A_{rt})_{p\alpha}$ lie in row i of A by Equation (8). These elements fill out the entire row i of A, since $1 \leqq t \leqq n$ and $1 \leqq \alpha \leqq n_t$. Similar reasoning shows that the elements $(B_{ts})_{\alpha q}$ lie in column j of B because of Equation (9). Also the entire column j of B is present and appears in its natural order. Hence

$$c_{ij} = \sum_{\beta=1}^{n} (A)_{i\beta}(B)_{\beta j} = (AB)_{ij}$$ ∎

PROGRAMMING PROJECT

For readers interested in numerical experimentation, we suggest a programming project that can be carried out in several stages. The project involves writing a number of subroutines to do basic tasks in linear algebra. This set of subroutines will provide a personal software package for solving linear equations, factoring matrices in various ways, and computing eigenvalues and pseudoinverses. The first part of the project is described in Problems 12 (Section 4.1), 4 (Section 4.3), 50–55 (Section 4.3), and 56 (Section 4.4).

PROBLEM SET 4.1

1. Show that an elementary operation of the first type can be accomplished by four operations of the other types, and that we can restrict λ to be ± 1.
2. Is Theorem 1 true for systems in which the number of equations differs from the number of unknowns?
3. Prove that each of the elementary operations can be undone by an operation of the same type.
4. Consider the system of linear equations $Ax = b$, where A is an $m \times n$ matrix, x is $n \times 1$, and b is $m \times 1$. Denote the column vectors of A by $A_1, A_2, \ldots, A_n$. Prove that the system has a solution if and only if b is in the linear span of $\{A_1, A_2, \ldots, A_n\}$. Prove that if $\{A_1, A_2, \ldots, A_n\}$ is linearly independent, then the system has at most one solution.
5. Let $E(p, q, \lambda)$ be the matrix that results from the $n \times n$ identity matrix when λ times row q is added to row p. (Assume that $p \neq q$.) Prove that the relationship $E(p, q, \lambda)^{-1} = E(p, q, -\lambda)$ holds. Prove that for any $m \times n$ matrix A, the product $AE(p, q, \lambda)$ can be computed by adding λ times column p to column q in A.
6. A *monomial* matrix is a square matrix in which each row and column contains exactly one nonzero entry. Prove that a monomial matrix is nonsingular.
7. Let A have the block form

$$A = \begin{bmatrix} B & C \\ 0 & I \end{bmatrix}$$

in which the blocks are $n \times n$. Prove that if $B - I$ is nonsingular, then, for $k \geq 1$,

$$A^k = \begin{bmatrix} B^k & (B^k - I)(B - I)^{-1}C \\ 0 & I \end{bmatrix}$$

8. Carry out the multiplication of the following two matrices—first using the block product method and then using the ordinary product.

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 1 & -1 & 0 & 1 \end{bmatrix} \\ \begin{bmatrix} -1 & 1 \\ 0 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 2 \end{bmatrix} & \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 2 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ -2 & 1 \\ -1 & 1 \end{bmatrix} \end{bmatrix}$$

9. Refer to Problem 7 and find the block structure of A^k when

$$A = \begin{bmatrix} B & 0 \\ C & I \end{bmatrix}$$

Prove your result by mathematical induction.

10. Prove that the set of upper triangular $n \times n$ matrices is a subalgebra of the algebra of all $n \times n$ matrices. In other words, prove that the set is algebraically closed under the operations of addition, multiplication, and multiplication by a scalar.
11. Prove that the inverse of a nonsingular upper triangular matrix is also upper triangular.

12. Write and test subroutines or procedures for the following:

    (i) `STORE` (n, x, y), which replaces the n-vector y by the n-vector x.

    (ii) `PROD` (m, n, A, x, y), which multiplies the n-vector x by the $m \times n$ matrix A and stores the result in the m-vector y.

    (iii) `MULT` (k, m, n, A, B, C), which computes $C = AB$, where A is $k \times m$, B is $m \times n$, and C is $k \times n$.

    (iv) `DOT` (n, x, y, a), which computes (in double-precision arithmetic) the dot product $\sum_{i=1}^{n} x_i y_i$ and stores the answer as a single-precision real number, a. *Note*: x_i, y_i, a are single-precision numbers.

13. Let A be an $n \times n$ invertible matrix, and let u and v be two vectors in $\mathbb{R}^n$. Find the necessary and sufficient conditions on u and v in order that the matrix

$$\begin{bmatrix} A & u \\ v^T & 0 \end{bmatrix}$$

    be invertible, and give a formula for the inverse when it exists.

14. Let D be a matrix in partitioned form:

$$D = \begin{bmatrix} A & B \\ C & I \end{bmatrix}$$

    Prove that if $A - BC$ is nonsingular, then so is D.

15. (Continuation) Prove the stronger result that the dimension of the null space of D is no greater than the dimension of the null space of $A - BC$.

16. Are these matrices positive definite?

    (a) $\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ (b) $\begin{bmatrix} 4 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{bmatrix}$

17. For what values of a is this matrix positive definite?

$$A = \begin{bmatrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{bmatrix}$$

18. A square matrix A is said to be skew-symmetric if $A^T = -A$. Prove that if A is skew-symmetric, then $x^T A x = 0$ for all x.

19. (Continuation) Prove that the diagonal elements and the determinant of a skew-symmetric matrix are 0.

20. (Continuation) Let A be any square matrix, and define

$$A_0 = \tfrac{1}{2}(A + A^T) \qquad A_1 = \tfrac{1}{2}(A - A^T)$$

    Prove that A_0 is symmetric, that A_1 is skew-symmetric, that $A = A_0 + A_1$, and that for all x, $x^T A x = x^T A_0 x$. This explains why, in discussing quadratic forms, we can confine our attention to symmetric matrices.

21. Give an example of a symmetric matrix A containing all positive elements such that $x^T A x$ is sometimes negative.

22. Can a matrix have a right inverse and a left inverse that are not equal?

4.2 The LU and Cholesky Factorizations

Let us consider a system of n linear equations in n unknowns $x_1, x_2, \ldots, x_n$. It can be written in the form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

The matrices in this equation are denoted by A, x, and b. Thus, our system is simply

$$Ax = b \tag{1}$$

Easy-to-Solve Systems

We begin by looking for special types of systems that can be *easily* solved. For example, suppose that the $n \times n$ matrix A has a *diagonal* structure. This means that all the nonzero elements of A are on the main diagonal and System (1) is

$$\begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

In this case, our system collapses to n simple equations and the solution is

$$x = \begin{bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ b_3/a_{33} \\ \vdots \\ b_n/a_{nn} \end{bmatrix}$$

If $a_{ii} = 0$ for some index i, and if $b_i = 0$ also, then x_i can be *any* real number. If $a_{ii} = 0$ and $b_i \neq 0$, no solution of the system exists.

Continuing our search for *easy* solutions of System (1), we assume a *lower triangular* structure for A. This means that all the nonzero elements of A are situated on or below the main diagonal and System (1) is

$$\begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

To solve this, assume that $a_{ii} \neq 0$ for all i; then obtain x_1 from the first equation. With the known value of x_1 substituted in the second equation, solve the second equation for x_2. We proceed in the same way, obtaining $x_1, x_2, \ldots, x_n$, one at a time, *and in this order*. A formal algorithm for the solution in this case is called **forward substitution**:

input $n, (a_{ij}), (b_i)$
for $i = 1, 2, \ldots, n$ **do**

$$x_i \leftarrow \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j\right) \Big/ a_{ii} \tag{2}$$

end
output (x_i)

As is customary, any sum of the type $\sum_{i=\alpha}^{\beta} x_i$ in which $\beta < \alpha$ is interpreted to be 0.

The same ideas can be exploited to solve a system having an *upper triangular* structure. Such a matrix system has the form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Again, it must be assumed that $a_{ii} \neq 0$ for $1 \leqq i \leqq n$. The formal algorithm to solve for x is as follows and is called **back substitution**:

input $n, (a_{ij}), (b_i)$
for $i = n, n-1, \ldots, 1$ **do**

$$x_i \leftarrow \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j\right) \Big/ a_{ii} \tag{3}$$

end
output (x_i)

There is still another simple type of system that can be easily solved using these same ideas; namely, a system obtained by permuting the equations in a triangular system. To illustrate, consider the system

$$\begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & a_{23} \\ a_{31} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{4}$$

If we simply reorder these equations, we can get a lower triangular system:

$$\begin{bmatrix} a_{31} & 0 & 0 \\ a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_3 \\ b_1 \\ b_2 \end{bmatrix} \tag{5}$$

This can be solved in the same manner as indicated previously. Putting this in another way, we should solve the equations of System (4) *not* in the usual order $1, 2, 3$ but rather in the order $3, 1, 2$.

Let us try to describe in a formal way the matrix property just considered. We wish to say that one row of A, say row p_1, has zeros in positions $2, 3, \ldots, n$. Then another row, say row p_2, has zeros in positions $3, 4, \ldots, n$ and so on. If this is the case, we shall use row p_1 to obtain x_1, row p_2 to obtain x_2, and so on. If we were to reorder the rows in the order $p_1, p_2, \ldots, p_n$ the resulting matrix would be lower triangular.

How do we solve $Ax = b$ if A is a permuted upper or lower triangular matrix of the type just considered? Let's assume that the *permutation vector* $(p_1, p_2, \ldots, p_n)$ is known or has been determined somehow beforehand. Modifying our previous algorithms, we arrive at **forward substitution** for a permuted system:

input $n, (a_{ij}), (b_i), (p_i)$
for $i = 1, 2, \ldots, n$ **do**

$$x_i \leftarrow \left(b_{p_i} - \sum_{j=1}^{i-1} a_{p_i j} x_j \right) \Big/ a_{p_i i} \tag{6}$$

end
output (x_i)

Of course, this works only in the case that A has the property $a_{p_i j} = 0$ for $j > i$, and $a_{p_i i} \neq 0$ for all i. Similarly, **back substitution** for a permuted system is as follows:

input $n, (a_{ij}), (b_i), (p_i)$
for $i = n, n-1, \ldots, 1$ **do**

$$x_i \leftarrow \left(b_{p_i} - \sum_{j=i+1}^{n} a_{p_i j} x_j \right) \Big/ a_{p_i i} \tag{7}$$

end
output (x_i)

This works if $a_{p_i j} = 0$ for $j < i$, and $a_{p_i i} \neq 0$ for all i.

The algorithms (2), (3), (6), and (7) have not been given with sufficient detail for direct translation into most programming languages. For example, algorithm (7) might require this more elaborate set of instructions:

input $n, (a_{ij}), (b_i), (p_i)$
for $i = n, n-1, \ldots, 1$ **do**
  $s \leftarrow b_{p_i}$
  for $j = i+1, i+2, \ldots, n$ **do**
    $s \leftarrow s - a_{p_i j} x_j$
  end
  $x_i \leftarrow s / a_{p_i i}$
end
output (x_i)

LU Factorizations

Suppose that A can be factored into the product of a lower triangular matrix L and an upper triangular matrix U: $A = LU$. Then in order to solve the system of equations $Ax = b$ it is enough to solve this problem in two stages:

$$\begin{aligned} Lz &= b \qquad \text{solve for } z \\ Ux &= z \qquad \text{solve for } x \end{aligned}$$

Our previous analysis indicates that solving these two *triangular* systems is simple.

We shall show how the factorization $A = LU$ can be carried out, provided that in certain steps of the computation zero divisors are not encountered. Not every matrix has such a factorization, and this difficulty will be investigated presently.

We begin with an $n \times n$ matrix A and search for matrices

$$L = \begin{bmatrix} \ell_{11} & 0 & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & 0 & \cdots & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & \ell_{nn} \end{bmatrix} \qquad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

such that

$$A = LU \tag{8}$$

When this is possible, we say that A has an LU**-decomposition**. It turns out that L and U are *not* uniquely determined by Equation (8). In fact, for each i, we can *assign* a nonzero value to either ℓ_{ii} or u_{ii} (but not both). For example, one simple choice is to set $\ell_{ii} = 1$ for $i = 1, 2, \ldots, n$, thus making L *unit lower triangular*. Another obvious choice is to make U *unit upper triangular* ($u_{ii} = 1$ for each i). These special cases are of particular importance.

To derive an algorithm for the LU-factorization of A, we start with the formula for matrix multiplication:

$$a_{ij} = \sum_{s=1}^{n} \ell_{is} u_{sj} = \sum_{s=1}^{\min(i,j)} \ell_{is} u_{sj} \tag{9}$$

Here we have used the fact that $\ell_{is} = 0$ for $s > i$ and $u_{sj} = 0$ for $s > j$.

Each step in this process determines one new row of U and one new column in L. At step k, we can assume that rows $1, 2, \ldots, k-1$ have been computed in U and that columns $1, 2, \ldots, k-1$ have been computed in L. (If $k = 1$, this assumption is true vacuously.) Putting $i = j = k$ in Equation (9), we obtain

$$a_{kk} = \sum_{s=1}^{k-1} \ell_{ks} u_{sk} + \ell_{kk} u_{kk} \tag{10}$$

If u_{kk} or ℓ_{kk} has been specified, we use Equation (10) to determine the other. With ℓ_{kk} and u_{kk} now known, we use Equation (9) to write for the kth row ($i = k$) and

the kth column ($j = k$), respectively,

$$a_{kj} = \sum_{s=1}^{k-1} \ell_{ks} u_{sj} + \ell_{kk} u_{kj} \qquad (k+1 \leqq j \leqq n) \tag{11}$$

$$a_{ik} = \sum_{s=1}^{k-1} \ell_{is} u_{sk} + \ell_{ik} u_{kk} \qquad (k+1 \leqq i \leqq n) \tag{12}$$

If $\ell_{kk} \neq 0$, Equation (11) can be used to obtain the elements u_{kj}. Similarly, if $u_{kk} \neq 0$, Equation (12) can be used to obtain the elements ℓ_{ik}. It is interesting to note that these two computations can theoretically be carried out in parallel (that is, simultaneously). On some computers this can actually be done, with a considerable saving in execution time. For details, see Kincaid and Oppe [1988]. The computation of the kth row in U and the kth column in L as described completes the kth step in the algorithm. The calculations call for division by ℓ_{kk} and u_{kk}; hence, if these divisors are zero the computations can usually not be completed. However in some cases, they *can* be completed. (See Problem 43.)

The algorithm based on the preceding analysis is known as **Doolittle's factorization** when L is unit lower triangular ($\ell_{ii} = 1$ for $1 \leqq i \leqq n$) and as **Crout's factorization** when U is unit upper triangular ($u_{ii} = 1$ for $1 \leqq i \leqq n$). When $U = L^T$ so that $\ell_{ii} = u_{ii}$ for $1 \leqq i \leqq n$, the algorithm is called **Cholesky's factorization**. We shall discuss the Cholesky method in more detail later in this section since this factoring requires the matrix A to have several special properties—namely, A should be real, symmetric, and positive definite.

The algorithm for the general LU-factorization is as follows:

input $n, (a_{ij})$
for $k = 1, 2, \ldots, n$ **do**
  Specify a nonzero value for either
    ℓ_{kk} or u_{kk} and compute the other from
    $$\ell_{kk} u_{kk} = a_{kk} - \sum_{s=1}^{k-1} \ell_{ks} u_{sk}$$
  for $j = k+1, k+2, \ldots, n$ **do**
    $$u_{kj} \leftarrow \left(a_{kj} - \sum_{s=1}^{k-1} \ell_{ks} u_{sj} \right) \Big/ \ell_{kk}$$
  end
  for $i = k+1, k+2, \ldots, n$ **do**
    $$\ell_{ik} \leftarrow \left(a_{ik} - \sum_{s=1}^{k-1} \ell_{is} u_{sk} \right) \Big/ u_{kk}$$
  end
end
output $(\ell_{ij}), (u_{ij})$

Notice the possible parallelism (simultaneity) in computing the kth row of U and the kth column of L. (See Problem 59.)

In the factorization $A = LU$ with L lower triangular and U upper triangular, n^2 equations are obtained from Equation (9) for the $n^2 + n$ unknown elements of L and U. Obviously, n of them have to be specified. An even more general factorization would allow any n elements from L and U to be specified and solve the resulting system of equations. Unfortunately, this system may be *nonlinear* in ℓ_{ij} and u_{ij}. (See Problem 53.) In the preceding algorithm, we must designate either ℓ_{kk} or u_{kk} and then compute all elements in the kth column of L and in the kth row of U before moving on to the next column and row. Moreover, the order $k = 1, 2, \ldots, n$ is important in these calculations.

The pseudocode for carrying out Doolittle's factorization is as follows:

input $n, (a_{ij})$
for $k = 1, 2, \ldots, n$ **do**
  $\ell_{kk} \leftarrow 1$
  for $j = k, k+1, \ldots, n$
    $u_{kj} \leftarrow a_{kj} - \sum_{s=1}^{k-1} \ell_{ks} u_{sj}$
  end
  for $i = k+1, k+2, \ldots, n$ **do**
    $\ell_{ik} \leftarrow \left(a_{ik} - \sum_{s=1}^{k-1} \ell_{is} u_{sk} \right) \Big/ u_{kk}$
  end
end
output $(\ell_{ij}), (u_{ij})$

Example Find the Doolittle, Crout, and Cholesky factorizations of the matrix

$$A = \begin{bmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{bmatrix}$$

Solution The Doolittle factorization from the algorithm is

$$A = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{bmatrix} \begin{bmatrix} 60 & 30 & 20 \\ 0 & 5 & 5 \\ 0 & 0 & \frac{1}{3} \end{bmatrix} \equiv LU$$

Rather than computing the next two factorizations directly, we can obtain them from the Doolittle factorization above. By putting the diagonal elements of U into a diagonal matrix D, we can write

$$A = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{bmatrix} \begin{bmatrix} 60 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \equiv LD\hat{U}$$

By putting $\widehat{L} = LD$, we obtain the Crout factorization

$$A = \begin{bmatrix} 60 & 0 & 0 \\ 30 & 5 & 0 \\ 20 & 5 & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \equiv \widehat{L}\widehat{U}$$

The Cholesky factorization is obtained by splitting D into the form $D^{1/2}D^{1/2}$ in the $LD\widehat{U}$-factorization and associating one factor with L and the other with $\widehat{U}$. Thus

$$A = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{60} & 0 & 0 \\ 0 & \sqrt{5} & 0 \\ 0 & 0 & \frac{1}{3}\sqrt{3} \end{bmatrix} \begin{bmatrix} \sqrt{60} & 0 & 0 \\ 0 & \sqrt{5} & 0 \\ 0 & 0 & \frac{1}{3}\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{60} & 0 & 0 \\ \frac{1}{2}\sqrt{60} & \sqrt{5} & 0 \\ \frac{1}{3}\sqrt{60} & \sqrt{5} & \frac{1}{3}\sqrt{3} \end{bmatrix} \begin{bmatrix} \sqrt{60} & \frac{1}{2}\sqrt{60} & \frac{1}{3}\sqrt{60} \\ 0 & \sqrt{5} & \sqrt{5} \\ 0 & 0 & \frac{1}{3}\sqrt{3} \end{bmatrix} \equiv \widetilde{L}\widetilde{L}^T$$

■

The factorization of primary interest is $A = LU$ where L is *unit* lower triangular and U is upper triangular. Henceforth, when we refer to an LU-factorization, we shall mean one in which L is *unit* lower triangular. Here is a sufficient condition for a square matrix A to have an LU-decomposition.

THEOREM 1 *If all n leading principal minors of the $n \times n$ matrix A are nonsingular, then A has an LU-decomposition.*

Proof Recall that the kth leading **principal minor** of the matrix A is the matrix

$$A_k = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix}$$

Let A_k, L_k, and U_k denote the kth leading principal minors in the matrices A, L, and U, respectively. Our hypothesis is that $A_1, A_2, \ldots, A_n$ are nonsingular.

For the purposes of an inductive proof, suppose that L_{k-1} and U_{k-1} have been obtained. In Equation (9), if i and j are in the range $1, 2, \ldots, k-1$, then so is s. Hence, Equation (9) states that

$$A_{k-1} = L_{k-1}U_{k-1}$$

Since A_{k-1} is nonsingular by hypothesis, L_{k-1} and U_{k-1} are also nonsingular. Since L_{k-1} is nonsingular, we can solve the system

$$\sum_{s=1}^{k-1} \ell_{is}u_{sk} = a_{ik} \qquad (1 \leq i \leq k-1)$$

for the quantities u_{sk} with $1 \leqq s \leqq k-1$. These elements lie in the kth column of U. Since U_{k-1} is nonsingular, we can solve the system

$$\sum_{s=1}^{k-1} \ell_{ks} u_{sj} = a_{kj} \qquad (1 \leqq j \leqq k-1)$$

for ℓ_{ks} with $1 \leqq s \leqq k-1$. These elements lie in the kth row of L. From the requirement

$$a_{kk} = \sum_{s=1}^{k} \ell_{ks} u_{sk} = \sum_{s=1}^{k-1} \ell_{ks} u_{sk} + \ell_{kk} u_{kk}$$

we can obtain u_{kk} since ℓ_{kk} has been specified as unity. Thus, all the new elements necessary to form L_k and U_k have been defined. The induction is completed by noting that $\ell_{11} u_{11} = a_{11}$ and, therefore, $\ell_{11} = 1$ and $u_{11} = a_{11}$. ∎

Cholesky Factorization

As mentioned earlier in this section, a matrix factorization that is useful in some situations has been given the name of the mathematician André Louis Cholesky, who proved the following result:

THEOREM 2 *If A is a real, symmetric, and positive definite matrix, then it has a unique factorization, $A = LL^T$, in which L is lower triangular with a positive diagonal.*

Proof Recall that a matrix A is *symmetric and positive definite* if $A = A^T$ and $x^T A x > 0$ for every nonzero vector x. It follows at once that A is nonsingular, for A obviously cannot map any nonzero vector into zero. Moreover, by considering special vectors of the form $x = (x_1, x_2, \ldots, x_k, 0, 0, \ldots, 0)^T$, we see that the leading principal minors of A are also positive definite. Theorem 1 implies that A has an LU decomposition. By the symmetry of A, we then have

$$LU = A = A^T = U^T L^T$$

This implies that

$$U(L^T)^{-1} = L^{-1} U^T$$

The left member of this equation is upper triangular, while the right member is lower triangular. (See Problem 1.) Consequently, there is a diagonal matrix D such that $U(L^T)^{-1} = D$. Hence, $U = DL^T$ and $A = LDL^T$. By Problem 27, D is positive definite, and thus its elements d_{ii} are positive. Denoting by $D^{1/2}$ the diagonal matrix whose diagonal elements are $\sqrt{d_{ii}}$, we have $A = \tilde{L}\tilde{L}^T$ where $\tilde{L} \equiv LD^{1/2}$, which is the Cholesky factorization. The proof of uniqueness is left as a problem. ∎

The algorithm for the Cholesky factorization is a special case of the general LU-factorization algorithm. If A is real, symmetric, and positive definite then by Theorem 2 it has a unique factorization of the form $A = LL^T$, in which L is lower triangular and has positive diagonal. Thus, in Equation (8), $U = L^T$. In the kth step of the general algorithm, the diagonal entry is computed by

$$\ell_{kk} = \left(a_{kk} - \sum_{s=1}^{k-1} \ell_{ks}^2 \right)^{1/2} \tag{13}$$

The algorithm for the Cholesky factorization will then be as follows

input $n, (a_{ij})$
for $k = 1, 2, \ldots, n$ **do**
$\quad \ell_{kk} \leftarrow \left(a_{kk} - \sum_{s=1}^{k-1} \ell_{ks}^2 \right)^{1/2}$
$\quad$ **for** $i = k+1, k+2, \ldots, n$ **do**
$\quad\quad \ell_{ik} \leftarrow \left(a_{ik} - \sum_{s=1}^{k-1} \ell_{is}\ell_{ks} \right) \Big/ \ell_{kk}$
$\quad$ **end**
end
output (ℓ_{ij})

Theorem 2 guarantees that $\ell_{kk} > 0$. Observe that Equation (13) gives us for $j \leqq k$

$$a_{kk} = \sum_{s=1}^{k} \ell_{ks}^2 \geqq \ell_{kj}^2$$

from which we conclude that

$$|\ell_{kj}| \leqq \sqrt{a_{kk}} \qquad (1 \leqq j \leqq k)$$

Hence, any element of L is bounded by the square root of a corresponding diagonal element in A. This implies that the elements of L do not become large relative to A even without any pivoting. (Pivoting is explained in the next section.)

In both the Cholesky and Doolittle algorithms, the dot products of vectors should be carried out in double precision in order to avoid a buildup of roundoff errors. (See Problem 6 in Section 4.1.)

PROBLEM SET 4.2

1. Prove these facts, needed in the proof of Theorem 2.
   (a) If U is upper triangular and invertible, then U^{-1} is upper triangular.
   (b) The inverse of a unit lower triangular matrix is unit lower triangular.
   (c) The product of two upper (lower) triangular matrices is upper (lower) triangular.
2. Prove that if a nonsingular matrix A has an LU-factorization in which L is a unit lower triangular matrix, then L and U are unique.
3. Prove that algorithms (2), (3), (6), and (7) always solve $Ax = b$ if A is nonsingular.
4. Prove that an upper or lower triangular matrix is nonsingular if and only if its diagonal elements are all different from zero.
5. Show that if all the principal minors of A are nonsingular and $\ell_{ii} \neq 0$ for each i, then $u_{kk} \neq 0$ for $1 \leqq k \leqq n$.

6. Prove that the matrix $A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ does not have an LU-factorization. *Caution:* This is *not* a simple consequence of the theorem proved in this section.

7. (a) Write the *row version* of the Doolittle algorithm that computes the kth row of L and the kth row of U at the kth step. (Consequently at the kth step, the order of computing is $\ell_{k1}, \ell_{k2}, \ldots, \ell_{k,k-1}, u_{kk}, \ldots, u_{kn}$.)

   (b) Write the *column version* of the Doolittle algorithm, which computes the kth column of U and the kth column of L at the kth step. (Consequently, the order of computing is $u_{1k}, u_{2k}, \ldots, u_{kk}, \ell_{k+1,k}, \ldots, \ell_{nk}$ at the kth step.)

8. By use of the equation $UU^{-1} = I$, obtain an algorithm for finding the inverse of an upper triangular matrix. Assume that U^{-1} exists; that is, the diagonal elements of U are all nonzero.

9. Count the number of arithmetic operations involved in the algorithms (2), (3), (6), and (7).

10. A matrix $A = (a_{ij})$ in which $a_{ij} = 0$ when $j > i$ or $j < i - 1$ is called a **Stieltjes matrix**. Devise an efficient algorithm for inverting such a matrix.

11. Let A be an $n \times n$ matrix. Let $(p_1, p_2, \ldots, p_n)$ be a permutation of $(1, 2, \ldots, n)$ such that (for $i = 1, 2, \ldots, n$) row i in A contains nonzero elements only in columns $p_1, p_2, \ldots, p_i$. Write an algorithm to solve $Ax = b$.

12. Show that every matrix of the form $A = \begin{bmatrix} 0 & a \\ 0 & b \end{bmatrix}$ has an LU-factorization. Show that even if L is *unit* lower triangular the factorization is not unique. (This problem, as well as Problems 13 and 15, illustrate Taussky's Maxim: If a conjecture about matrices is false, it can usually be disproved with a 2×2 example.)

13. Show that every matrix of the form $A = \begin{bmatrix} 0 & 0 \\ a & b \end{bmatrix}$ has an LU-factorization. Does it have an LU-factorization in which L is a *unit* lower triangular?

[c]14. Devise an efficient algorithm for inverting an $n \times n$ lower triangular matrix A. *Suggestion*: Utilize the fact that A^{-1} is also lower triangular. Code your algorithm and test it on the matrix whose elements are $a_{ij} = (i+j)^2$ when $i \geq j$. Use $n = 10$. Form the product AA^{-1} as a test of the computed inverse.

15. Find all the LU-factorizations of $A = \begin{bmatrix} 1 & 5 \\ 3 & 15 \end{bmatrix}$ in which L is unit lower triangular.

16. If A is invertible and has an LU decomposition then all principal minors of A are nonsingular.

17. Let the system $Ax = b$ have the following property: There are two permutations of $(1, 2, \ldots, n)$ called $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ such that, for each i, equation number p_i contains only the variables $x_{q_1}, x_{q_2}, \ldots, x_{q_i}$. Write an efficient algorithm to solve this system.

18. Count the number of multiplications and/or divisions needed to invert a unit lower triangular matrix.

19. Prove or disprove: If A has an LU-factorization in which L is *unit* lower triangular, then it has an LU-factorization in which U is *unit* upper triangular.

20. Assuming that its LU-factorization is known, give an algorithm for inverting A. (Use Problems 8 and 14.)

21. Develop an algorithm for inverting a matrix A that has the property $a_{ij} = 0$ if $i + j \leq n$.

22. Use the Cholesky Theorem to prove that these two properties of a symmetric matrix A are equivalent: (a) A is positive definite: (b) there exists a linearly independent set of vectors $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ in $\mathbb{R}^n$ such that $A_{ij} = (x^{(i)})^T(x^{(j)})$.

23. Establish the correctness of the following algorithm for solving $Ux = b$ in the case that U is upper triangular:

```
for j = n, n − 1, ..., 1 do
    x_j ← b_j / u_jj
    for i = 1, 2, ..., j − 1 do
        b_i ← b_i − u_ij x_j
    end
end
```

24. Prove that if all the leading principal minors of A are nonsingular, then A has a factorization LDU in which L is unit lower triangular, U is unit upper triangular, and D is diagonal.

25. (Continuation) If A is a symmetric matrix whose leading principal minors are nonsingular, then A has a factorization LDL^T in which L is unit lower triangular and D is diagonal.

26. (Continuation) Write an algorithm to compute the LDL^T-factorization of a symmetric matrix A. Your algorithm should do approximately half as much work as the standard Gaussian algorithm. *Note:* This algorithm can fail if some principal minors of A are singular. (This modification of the Cholesky algorithm does not involve square root calculations.)

27. Prove: A is positive definite and B is nonsingular if and only if BAB^T is positive definite.

28. If A is positive definite, does it follow that A^{-1} is also positive definite?

29. Consider

$$A = \begin{bmatrix} 2 & 6 & -4 \\ 6 & 17 & -17 \\ -4 & -17 & -20 \end{bmatrix}$$

Determine *directly* the factorization $A = LDL^T$ where D is diagonal and L is unit lower triangular—that is, do *not* use Gaussian elimination.

30. Develop an algorithm for finding directly the UL-factorization of a matrix A where L is *unit* lower triangular and U is upper triangular. Give an algorithm for solving $ULx = b$.

31. Find the LU-factorization of the matrix

$$A = \begin{bmatrix} 3 & 0 & 1 \\ 0 & -1 & 3 \\ 1 & 3 & 0 \end{bmatrix}$$

in which L is lower triangular and U is unit upper triangular.

32. Factor the matrix $A = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix}$ so that $A = LL^T$, where L is lower triangular.

33. Determine directly the LL^T-factorization, in which L is a lower triangular matrix with positive diagonal elements, for the matrix

$$A = \begin{bmatrix} 4 & \frac{1}{2} & 1 \\ \frac{1}{2} & \frac{17}{16} & \frac{1}{4} \\ 1 & \frac{1}{4} & \frac{33}{64} \end{bmatrix}$$

34. Suppose that the nonsingular matrix A has a Cholesky factorization. What can be said about the determinant of A?

35. Determine the LU-factorization of the matrix $A = \begin{bmatrix} 1 & 5 \\ 3 & 16 \end{bmatrix}$ in which *both* L and U have unit diagonal elements.

[c]36. Solve this system by the Cholesky method:

$$\begin{cases} 0.05x_1 + 0.07x_2 + 0.06x_3 + 0.05x_4 = 0.23 \\ 0.07x_1 + 0.10x_2 + 0.08x_3 + 0.07x_4 = 0.32 \\ 0.06x_1 + 0.08x_2 + 0.10x_3 + 0.09x_4 = 0.33 \\ 0.05x_1 + 0.07x_2 + 0.09x_3 + 0.10x_4 = 0.31 \end{cases}$$

37. Consider the symmetric tridiagonal positive definite matrix

$$A = \begin{bmatrix} 136.01 & 90.860 & 0.0 & 0.0 \\ 90.860 & 98.810 & -67.590 & 0.0 \\ 0.0 & -67.590 & 132.01 & 46.260 \\ 0.0 & 0.0 & 46.260 & 177.17 \end{bmatrix}$$

Using five significant figures, factor A in the following ways.

(a) $A = LU$, where L is unit lower triangular and U is upper triangular.

(b) $A = LDU$, where L is unit lower triangular, D is diagonal, and U is unit upper triangular.

(c) $A = LU$, where L is lower triangular and U is unit upper triangular.

(d) $A = LL^T$, where L is lower triangular.

38. Determine the LU-factorization of the matrix

$$A = \begin{bmatrix} 6 & 10 & 0 \\ 12 & 26 & 4 \\ 0 & 9 & 12 \end{bmatrix}$$

in which L is a lower triangular matrix with twos on its main diagonal.

39. Prove that if a singular matrix has a Doolittle factorization, then that factorization is not unique.

40. Prove the uniqueness of the factorization $A = LL^T$, where L is lower triangular and has positive diagonal.

41. A matrix A that is symmetric and positive definite (SPD) has a square root X that is SPD. Thus $X^2 = A$. Find X if $A = \begin{bmatrix} 13 & 10 \\ 10 & 17 \end{bmatrix}$.

42. Develop algorithms for solving the linear system $Ax = b$ in two special cases: (a) $a_{ij} = 0$ when $j \leq n - i$; (b) $a_{ij} = 0$ when $j > n + 1 - i$.

43. Using Equations (10), (11), and (12), find all the Doolittle factorizations of the matrix

$$A = \begin{bmatrix} 2 & 1 & -2 \\ 4 & 2 & -1 \\ 6 & 3 & 11 \end{bmatrix}$$

In this example, the algorithm works, although $u_{22} = 0$.

44. Prove that if A is symmetric, then in its LU-factorization the columns of L are multiples of the rows of U.

45. Find all UL-factorizations and all UL-factorizations in which L is *unit* lower triangular for the matrix $A = \begin{bmatrix} 1 & 5 \\ 3 & 17 \end{bmatrix}$.

46. Define a *P-matrix* to be one in which $a_{ij} = 0$ if $j \leq n - i$ and a *Q-matrix* to be a P-matrix in which $a_{i,n-i+1} = 1$ for $i = 1, 2, \ldots, n$. Find the *PQ-factorization* of the matrix $A = \begin{bmatrix} 3 & 15 \\ -1 & -1 \end{bmatrix}$.

47. (Continuation) Devise an algorithm to obtain the PQ-factorization of a given matrix. Similarly, devise an algorithm for solving a system of equations of the form $PQx = b$.

48. Assuming that the LU-factorization of A is available, write an algorithm to solve the equation $x^T A = b^T$.

[c]49. Write a subprogram or procedure that implements the general LU-factorization algorithm. The diagonal elements that are prescribed can be stored in an array D. An associated logical array can be used to indicate whether an element of D belongs to the diagonal of L or U. Test the routine on some Hilbert matrices, whose elements are $a_{ij} = (i + j - 1)^{-1}$. For each matrix, produce the Doolittle, Crout, and Cholesky factorizations plus one or more others with specified diagonal entries.

50. If A has a Doolittle factorization what is a simple formula for the determinant of A?

51. Let

$$A = \begin{bmatrix} 25 & 0 & 0 & 0 & 1 \\ 0 & 27 & 4 & 3 & 2 \\ 0 & 54 & 58 & 0 & 0 \\ 0 & 108 & 116 & 0 & 0 \\ 100 & 0 & 0 & 0 & 24 \end{bmatrix}$$

Determine the most general LU-factorization of A in which L is unit lower triangular. Show that the Doolittle algorithm produces one of these LU-factorizations.

52. Let A be a symmetric matrix whose leading principal minors are nonnegative. Does the matrix $A + \varepsilon I$ have the same properties for $\varepsilon > 0$?

53. Consider the LU-factorization of a 2×2 matrix A. Show that if ℓ_{22} and u_{22} are specified, then the equations that determine the remaining elements of L and U are nonlinear.

54. Prove: If A is symmetric and nonnegative definite, then $A = LL^T$ for some lower triangular matrix L. The terminology **nonnegative definite** means that $x^T Ax \geq 0$ for all x.

55. Find the precise conditions on a, b, c in order that the matrix $\begin{bmatrix} a & b \\ b & c \end{bmatrix}$ will be nonnegative definite.

56. Prove that if the matrix $\begin{bmatrix} a & b \\ b & c \end{bmatrix}$ is nonnegative definite, then it has a factorization LL^T in which L is lower triangular.

57. Prove or disprove: A symmetric matrix is nonnegative definite if and only if all of its leading principal minors are nonnegative.

58. Find necessary and sufficient conditions on a, b, c in order that the matrix $\begin{bmatrix} a & b \\ b & c \end{bmatrix}$ has a factorization LL^T in which L is lower triangular.

59. In this problem, use the notation $X_{i..j,k}$ to denote the part of the kth column of the matrix X consisting of entries i to j. Similarly, let $X_{k,i..j}$ be the part of row k in X consisting of entries i to j.
(a) Refer to Equation (11) and show that it can be written as

$$U_{k,k+1:n} = (A_{k,k+1:n} - L_{k,1:k-1}M)/\ell_{kk}$$

in which M is the matrix whose rows are $U_{i,\,k+1:n}$, for $1 \leqq i \leqq k-1$.
(b) Carry out the analogous transformation of Equation (12). The computations discussed in this problem have the form $y \leftarrow y - Mx$. They can be carried out very efficiently on a vector supercomputer. (See Kincaid and Oppe [1988] and Oppe and Kincaid [1988] for further details.)

4.3 Pivoting and Constructing an Algorithm

In the previous section, an abstract version of Gaussian elimination was presented in the guise of the LU-factorization of a matrix. In this section, the traditional form of Gaussian elimination will be described and related to the abstract form. Then we shall take up the modifications of the process necessary to produce a satisfactory computer realization of it. Throughout this discussion, we shall use the words *equation* and *row* of a matrix system interchangeably.

Basic Gaussian Elimination

Here is a simple system of four equations in four unknowns that will be used to illustrate the Gaussian algorithm:

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 34 \\ 27 \\ -38 \end{bmatrix} \tag{1}$$

In the first step of the process, we subtract 2 times the first equation from the second. Then we subtract $\frac{1}{2}$ times the first equation from the third. Finally, we subtract -1 times the first equation from the fourth. The numbers 2, $\frac{1}{2}$, and -1 are called the **multipliers** for the first step in the elimination process. The number 6 used as the divisor in forming each of these multipliers is called the **pivot element** for this step.

After the first step has been completed the system will look like this:

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ 21 \\ -26 \end{bmatrix} \tag{2}$$

Although the first row was used in the process, it was not changed. In this first step we refer to row 1 as the **pivot row**. In the next step of the process, row 2 is used as the pivot row and -4 is the pivot element. We subtract 3 times the second row from the third, and then $-\frac{1}{2}$ times the second row is subtracted from the fourth. The multipliers for this step are 3 and $-\frac{1}{2}$. The result is

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -21 \end{bmatrix} \tag{3}$$

The final step consists of subtracting 2 times the third row from the fourth so that 2 is both the multiplier and pivot element. The resulting system is

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -3 \end{bmatrix} \tag{4}$$

This system is upper triangular and **equivalent** to the original system in the sense that the solutions of the two systems are the same. The final system is easily solved by starting at the fourth row and working backward up the rows. The solution is

$$x = \begin{bmatrix} 1 \\ -3 \\ -2 \\ 1 \end{bmatrix}$$

The multipliers used in transforming the system can be exhibited in a unit lower triangular matrix $L = (\ell_{ij})$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{bmatrix} \tag{5}$$

Notice that each multiplier is written in the location corresponding to the zero entry in the matrix it was responsible for creating. The coefficient matrix of the final system is an upper triangular matrix $U = (u_{ij})$

$$U = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \tag{6}$$

These two matrices give the LU-factorization of A, where A is the coefficient matrix of the original system. Thus

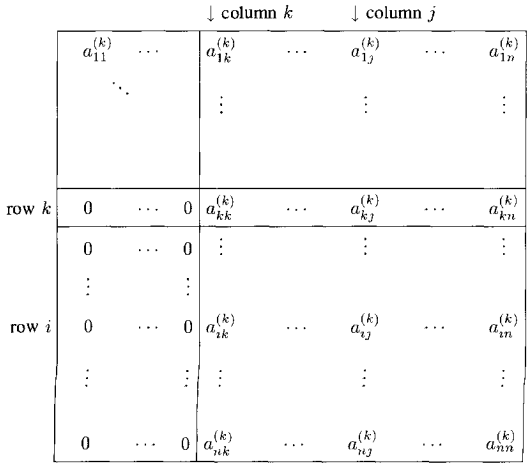
$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{bmatrix} \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \tag{7}$$

It is not hard to see why this must be true. If we know how U was obtained from A, then by reversing the process we can get A from U. If we denote the rows of A by A_1, A_2, A_3, A_4 and the rows of U by U_1, U_2, U_3, U_4, then the elimination process gives us, for example, $U_2 = A_2 - 2A_1$. Hence, $A_2 = 2A_1 + U_2 = 2U_1 + U_2$. The coefficients 2 and 1 occupy the second row of L. Similarly, the row operations leading to row 3 are $U_3 = (A_3 - \frac{1}{2}A_1) - 3U_2$, and finally we have $A_3 = \frac{1}{2}A_1 + 3U_2 + U_3 = \frac{1}{2}U_1 + 3U_2 + U_3$. The coefficients $\frac{1}{2}, 3, 1$ must therefore occupy the third row of L, and so on.

To describe formally the progress of the Gaussian algorithm, we interpret it as a succession of $n - 1$ major steps resulting in a sequence of matrices as follows:

$$A = A^{(1)} \to A^{(2)} \to \cdots \to A^{(n)}$$

At the conclusion of step $k - 1$, the matrix $A^{(k)}$ will have been constructed; its appearance is shown in the following display.



Our task is to describe how $A^{(k+1)}$ is obtained from $A^{(k)}$. To produce zeros in column k below the pivot element $a_{kk}^{(k)}$, we subtract multiples of row k from the rows beneath

it. Rows $1, 2, \ldots, k$ are *not* altered. The formula is therefore

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} & \text{if } i \leqq k \\ a_{ij}^{(k)} - \left(a_{ik}^{(k)} / a_{kk}^{(k)}\right) a_{kj}^{(k)} & \text{if } i \geqq k+1 \text{ and } j \geqq k+1 \\ 0 & \text{if } i \geqq k+1 \text{ and } j \leqq k \end{cases} \tag{8}$$

Then we define $U = A^{(n)}$ and define L by

$$\ell_{ik} = \begin{cases} a_{ik}^{(k)} / a_{kk}^{(k)} & \text{if } i \geqq k+1 \\ 1 & \text{if } i = k \\ 0 & \text{if } i \leqq k-1 \end{cases} \tag{9}$$

Here $A = LU$ is the standard Gaussian factorization of matrix A with L *unit* lower triangular and U upper triangular. It should be clear from Equations (8) and (9), as well as from the previous numerical example, that this entire elimination process will break down if any of the pivot elements are zero. Now we can prove:

THEOREM 1 *If all the pivot elements $a_{kk}^{(k)}$ are nonzero in the process just described, then $A = LU$.*

Proof Observe that $a_{ij}^{(k+1)} = a_{ij}^{(k)}$ if $i \leqq k$ or $j \leqq k-1$. Next note that $u_{kj} = a_{kj}^{(n)} = a_{kj}^{(k)}$. Finally, note that $\ell_{ik} = 0$ if $k > i$ and $u_{kj} = 0$ if $k < j$. Now let $i \leqq j$. Using the above facts, we have

$$\begin{aligned} (LU)_{ij} &= \sum_{k=1}^{n} \ell_{ik} u_{kj} = \sum_{k=1}^{i} \ell_{ik} u_{kj} = \sum_{k=1}^{i} \ell_{ik} a_{kj}^{(k)} \\ &= \sum_{k=1}^{i-1} \ell_{ik} a_{kj}^{(k)} + \ell_{ii} a_{ij}^{(i)} = \sum_{k=1}^{i-1} \left(a_{ik}^{(k)} / a_{kk}^{(k)}\right) a_{kj}^{(k)} + a_{ij}^{(i)} \\ &= \sum_{k=1}^{i-1} (a_{ij}^{(k)} - a_{ij}^{(k+1)}) + a_{ij}^{(i)} = a_{ij}^{(1)} = a_{ij} \end{aligned}$$

Similarly, if $i > j$, then

$$\begin{aligned} (LU)_{ij} &= \sum_{k=1}^{j} \ell_{ik} a_{kj}^{(k)} = \sum_{k=1}^{j} (a_{ij}^{(k)} - a_{ij}^{(k+1)}) \\ &= a_{ij}^{(1)} - a_{ij}^{(j+1)} = a_{ij}^{(1)} = a_{ij} \end{aligned}$$

since $a_{ij}^{(k)} = 0$ if $i \geqq j+1$ and $k \geqq j+1$. ∎

An algorithm to carry out the basic Gaussian elimination process just described on the matrix $A = (a_{ij})$ is as follows:

```
input n, (a_ij)
for k = 1, 2, ..., n − 1 do
    for i = k + 1, k + 2, ..., n do
        z ← a_ik / a_kk
        a_ik ← 0
        for j = k + 1, k + 2, ..., n do
            a_ij ← a_ij − z a_kj
        end
    end
end
output (a_ij)
```

Here it is assumed that all the pivot elements are nonzero. The multipliers are chosen so that entries below the main diagonal in A are computed to be zero. Rather than carrying out this computation, we simply replace these entries with zeros in the algorithm.

Pivoting

The Gaussian algorithm, in the simple form just described, is not satisfactory since it fails on systems that are in fact easy to solve. To illustrate this remark, we consider three elementary examples. The first is

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{10}$$

The simple version of the algorithm fails because there is no way of adding a multiple of the first equation to the second in order to obtain a 0-coefficient of x_1 in the second equation. (See Problem 6 in the preceding section.)

The difficulty just encountered will persist for the following system, in which ε is a small number different from zero.

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{11}$$

When applied to (11), the Gaussian algorithm will produce this upper triangular system:

$$\begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - \varepsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 - \varepsilon^{-1} \end{bmatrix} \tag{12}$$

The solution is

$$\begin{cases} x_2 = \dfrac{2 - \varepsilon^{-1}}{1 - \varepsilon^{-1}} \approx 1 \\ x_1 = (1 - x_2)\varepsilon^{-1} \approx 0 \end{cases} \tag{13}$$

In the computer, if ε is small enough, $2 - \varepsilon^{-1}$ will be computed to be the same as $-\varepsilon^{-1}$. Likewise, the denominator $1 - \varepsilon^{-1}$ will be computed to be the same as $-\varepsilon^{-1}$. Hence, under these circumstances, x_2 will be computed as 1, and x_1 will be computed as 0. Since the correct solution is

$$\begin{cases} x_1 = \dfrac{1}{1-\varepsilon} \approx 1 \\ x_2 = \dfrac{1-2\varepsilon}{1-\varepsilon} \approx 1 \end{cases}$$

the computed solution is accurate for x_2 but is extremely inaccurate for x_1!

In the computer, if ε is small enough, why does the computation of $2 - \varepsilon^{-1}$ lead to the same machine number as the computation of $-\varepsilon^{-1}$? The reason is that before the subtraction can take place, the *exponents* in the floating point form of 2 and ε^{-1} must be made to agree by shifting the radix point. If this shift is great enough, the mantissa of 2 will be zero. For example, in a seven-place decimal machine similar to the hypothetical MARC-32, with $\varepsilon = 10^{-8}$, we have $\varepsilon^{-1} = .1000000 \times 10^9$ and $2 = .2000000 \times 10^1$. If 2 is rewritten with exponent 9, we have $2 = .000000002 \times 10^9$ and $2 - \varepsilon^{-1} = -.099999998 \times 10^9$, so that $2 - \varepsilon^{-1} = -.1000000 \times 10^9 = -\varepsilon^{-1}$ in the machine.

The final example will show that it is not actually the smallness of the coefficient a_{11} that is causing the trouble. Rather, it is the smallness of a_{11} *relative* to other elements in its row. Consider the following system, which is equivalent to the preceding one:

$$\begin{bmatrix} 1 & \varepsilon^{-1} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varepsilon^{-1} \\ 2 \end{bmatrix} \tag{14}$$

The simple Gaussian algorithm produces

$$\begin{bmatrix} 1 & \varepsilon^{-1} \\ 0 & 1-\varepsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varepsilon^{-1} \\ 2-\varepsilon^{-1} \end{bmatrix} \tag{15}$$

The solution of (15) is

$$\begin{cases} x_2 = \dfrac{2-\varepsilon^{-1}}{1-\varepsilon^{-1}} \approx 1 \\ x_1 = \varepsilon^{-1} - \varepsilon^{-1} x_2 \approx 0 \end{cases}$$

Again, for small ε, x_2 will be computed as 1 and x_1 will be computed as 0, which is, as before, wrong!

The difficulties in these examples will disappear if the order of the equations is changed. Thus, interchanging of the two equations in System (11) leads to

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Gaussian elimination applied to this system produces

$$\begin{bmatrix} 1 & 1 \\ 0 & 1-\varepsilon \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1-2\varepsilon \end{bmatrix}$$

The solution is then given by

$$\begin{cases} x_2 = \dfrac{1-2\varepsilon}{1-\varepsilon} \approx 1 \\ x_1 = 2 - x_1 \approx 1 \end{cases}$$

The conclusion to be drawn from these elementary examples is that good algorithms must incorporate the interchanging of equations in a system when the circumstances require it. For reasons of economy in computing, we prefer *not* to move the rows of the matrix around in the computer's memory. Instead, we simply choose the *pivot rows* in a logical manner. Suppose that instead of using the rows in the order $1, 2, \ldots, n-1$ as pivot rows, we use rows $p_1, p_2, \ldots, p_{n-1}$. Then in the first step, multiples of row p_1 will be subtracted from the other rows. If we introduce entry p_n so that $(p_1, p_2, \ldots, p_n)$ is a permutation of $(1, 2, \ldots, n)$, then row p_n will not occur as a pivot row, but we can say that multiples of row p_1 will be subtracted from rows $p_2, p_3, \ldots, p_n$. In the next step, multiples of row p_2 will be subtracted from rows $p_3, p_4, \ldots, p_n$; and so on.

Here is an algorithm to accomplish this. (It is assumed that the permutation array p has been predetermined and consists of the natural numbers $1, 2, \ldots, n$ in some order.)

input $n, (a_{ij})$
for $k = 1, 2, \ldots, n-1$ **do**
  for $i = k+1, k+2, \ldots, n$ **do**
    $z \leftarrow a_{p_i k}/a_{p_k k}$
    $a_{p_i k} \leftarrow 0$
    for $j = k+1, k+2, \ldots, n$ **do**
      $a_{p_i j} \leftarrow a_{p_i j} - z a_{p_k j}$
    end
  end
end
output (a_{ij})

Comparing this algorithm to the one for the basic Gaussian elimination process, we see that they are identical except for two changes. First, the computation of the multipliers has been moved outside the j-loop since they do not depend on j. Second, the first subscript on the elements of the coefficient array A involves the permutation array p. Of course, when the permutation array corresponds to the natural ordering $(p_i = i)$ the basic process will be obtained.

Gaussian Elimination with Scaled Row Pivoting

We now describe an algorithm called *Gaussian elimination with scaled row pivoting* for solving an $n \times n$ system

$$Ax = b$$

The algorithm consists of two parts: a *factorization* phase (also called *forward elimination*) and a *solution* phase (involving *updating* and *back substitution*). The factorization phase is applied to A only, and is designed to produce the LU-decomposition

of PA, where P is a permutation matrix derived from the permutation array p. (PA is obtained from A by permuting its rows.) The permuted linear system is

$$PAx = Pb$$

The factorization $PA = LU$ is obtained from a modified Gaussian elimination algorithm to be explained below. In the solution phase, we consider two equations $Lz = Pb$ and $Ux = z$. First, the right-hand side b is rearranged according to P and the results are stored back in b—that is, $b \leftarrow Pb$. Next, $Lz = b$ is solved for z and the results stored back in b—that is, $b \leftarrow L^{-1}b$. Since L is unit lower triangular, this amounts to a forward substitution process. This procedure is called *updating* b. Then, back substitution is used to solve $Ux = b$ for $x_n, x_{n-1}, \ldots, x_1$

In the factorization phase, we begin by computing the *scale* of each row. We put

$$s_i = \max_{1 \leqq j \leqq n} |a_{ij}| = \max\{|a_{i1}|, |a_{i2}|, \ldots, |a_{in}|\} \qquad (1 \leqq i \leqq n)$$

These are recorded in an array s in the algorithm.

In starting the factorization phase, we do not arbitrarily subtract multiples of row one from the other rows. Rather, we select as the *pivot row* the row for which $|a_{i1}|/s_i$ is largest. The index thus chosen is denoted by p_1 and becomes the first entry in the permutation array. Thus, $|a_{p_1 1}|/s_{p_1} \geqq |a_{i1}|/s_i$, for $1 \leqq i \leqq n$. Once p_1 has been determined, we subtract appropriate multiples of row p_1 from the other rows in order to produce zeros in the first column of A. Of course, row p_1 will remain unchanged throughout the remainder of the factorization process.

To keep track of the indices p_i that arise, we begin by setting the permutation vector $(p_1, p_2, \ldots, p_n)$ to $(1, 2, \ldots, n)$. Then we select an index j for which $|a_{p_j 1}|/s_{p_j}$ is maximal, and interchange p_1 with p_j in the permutation array p. The actual elimination step involves subtracting $(a_{p_i 1}/a_{p_1 1})$ times row p_1 from row p_i for $2 \leqq i \leqq n$.

To describe the general process, suppose that we are ready to create zeros in column k. We scan the numbers $|a_{p_i k}|/s_{p_i}$ for $k \leqq i \leqq n$ looking for a maximal entry. If j is the index of the *first* of the largest of these ratios, we interchange p_k with p_j in the array p, and then subtract $(a_{p_i k}/a_{p_k k})$ times row p_k from row p_i for $k+1 \leqq i \leqq n$.

Here is how this works on the matrix

$$A = \begin{bmatrix} 2 & 3 & -6 \\ 1 & -6 & 8 \\ 3 & -2 & 1 \end{bmatrix}$$

At the beginning, $p = (1, 2, 3)$ and $s = (6, 8, 3)$. To select the first pivot row, we look at the ratios $\{2/6, 1/8, 3/3\}$. The largest corresponds to $j = 3$, and row 3 is to be the first pivot row. So we interchange p_1 with p_3, obtaining $p = (3, 2, 1)$. Now multiples of row 3 are subtracted from rows 2 and 1 to produce zeros in the first column. The result is

$$\begin{bmatrix} \frac{2}{3} & \frac{13}{3} & -\frac{20}{3} \\ \frac{1}{3} & -\frac{16}{3} & \frac{23}{3} \\ 3 & -2 & 1 \end{bmatrix}$$

The entries in locations a_{11} and a_{21} are the multipliers. In the next step, the selection of a pivot row is made on the basis of the numbers $|a_{p_2 2}|/s_{p_2}$ and $|a_{p_3 2}|/s_{p_3}$. The first of these ratios is $(16/3)/8$ and the second is $(13/3)/6$. So $j = 3$, and we interchange p_2 with p_3. Then a multiple of row p_2 is subtracted from row p_3. The result is $p = (3, 1, 2)$ and

$$\begin{bmatrix} \frac{2}{3} & \frac{13}{3} & -\frac{20}{3} \\ \frac{1}{3} & -\frac{16}{13} & -\frac{7}{13} \\ 3 & -2 & 1 \end{bmatrix}$$

The final multiplier is stored in location a_{22}.

If the rows of the original matrix A were interchanged according to the final permutation array p, then we would have an LU-factorization of A. Hence, we have

$$PA = \begin{bmatrix} 1 & 0 & 0 \\ \frac{2}{3} & 1 & 0 \\ \frac{1}{3} & -\frac{16}{13} & 1 \end{bmatrix} \begin{bmatrix} 3 & -2 & 1 \\ 0 & \frac{13}{3} & -\frac{20}{3} \\ 0 & 0 & -\frac{7}{13} \end{bmatrix}$$

$$= \begin{bmatrix} 3 & -2 & 1 \\ 2 & 3 & -6 \\ 1 & -6 & 8 \end{bmatrix}$$

where

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & 3 & -6 \\ 1 & -6 & 8 \\ 3 & -2 & 1 \end{bmatrix}$$

The permutation matrix P is obtained from the permutation array p by setting $(P)_{ij} = \delta_{p_i j}$. In other words, P is formed by permuting the rows of the identity matrix I according to the entries in p.

Here is an algorithm to do the factorization phase of Gaussian elimination with scaled row pivoting:

input $n, (a_{ij})$
for $i = 1, 2, \ldots, n$ **do**
  $p_i \leftarrow i$
  $s_i \leftarrow \max_{1 \leqq j \leqq n} |a_{ij}|$
end
for $k = 1, 2, \ldots, n-1$ **do**
  select $j \geqq k$ so that
    $|a_{p_j k}|/s_{p_j} \geqq |a_{p_i k}|/s_{p_i}$ for $i = k, k+1, \ldots, n$
  $p_k \leftrightarrow p_j$
  for $i = k+1, k+2, \ldots, n$ **do**
    $z \leftarrow a_{p_i k}/a_{p_k k}$
    $a_{p_i k} \leftarrow z$
    for $j = k+1, k+2, \ldots, n$ **do**
      $a_{p_i j} \leftarrow a_{p_i j} - z a_{p_k j}$
    end
  end
end
output $(a_{ij}), (p_i)$

Notice that the *multipliers* are being stored in A at the locations where zeros would have been created by the elimination process. Hence, all the necessary data for reconstructing the LU-factorization are stored within this array. Nevertheless, the user should be aware that the algorithm overwrites the values in the original A-array. The matrix A should be saved in a separate array if it will be needed again.

To solve $Ax = b$, once the factorization phase has been carried out on A, we apply the forward phase of the algorithm to b, using the final permutation array p and the multipliers that were determined by the factorization phase of the algorithm. Next we carry out the back substitution—that is, solve for $x_n, x_{n-1}, \ldots, x_1$ in that order. Here is the algorithm that carries out the solution phase of the algorithm.

input $n, (a_{ij}), (p_i), (b_i)$
for $k = 1, 2, \ldots, n-1$ **do**
  for $i = k+1, k+2, \ldots, n$ **do**
    $b_{p_i} \leftarrow b_{p_i} - a_{p_i k} b_{p_k}$
  end
end
for $i = n, n-1, \ldots, 1$ **do**
  $x_i \leftarrow \left(b_{p_i} - \sum_{j=i+1}^{n} a_{p_i j} x_j \right) \Big/ a_{p_i i}$
end
output (x_i)

The first part of this pseudocode *updates* the right-hand side b corresponding to what happened during the factorization phase. The second part recovers x from a lower triangular system—that is, back substitution. Of course, the permutation array p must be used to keep things from being jumbled. Notice that when $i = n$, the j-sum is vacuous, so that $x_n = b_{p_n}/a_{p_n n}$.

Factorizations $PA = LU$

As alluded to earlier, if scaled row pivoting is included in the Gaussian algorithm, then we obtain the LU-factorization of PA, where P is a certain permutation matrix. A formal proof of this result will now be given, modeled on the proof of Theorem 1. The proof is *not* dependent on the strategy used to determine the pivots.

Let $p_1, p_2, \ldots, p_n$ be the indices of the rows in the order in which they become pivot rows. Let $A^{(1)} = A$, and define $A^{(2)}, A^{(3)}, \ldots, A^{(n)}$ recursively by the formula

$$a_{p_i j}^{(k+1)} = \begin{cases} a_{p_i j}^{(k)} & \text{if } i \leqq k \text{ or } i > k > j \\ a_{p_i j}^{(k)} - \left(a_{p_i k}^{(k)}/a_{p_k k}^{(k)}\right) a_{p_k j}^{(k)} & \text{if } i > k \text{ and } j > k \\ a_{p_i k}^{(k)}/a_{p_k k}^{(k)} & \text{if } i > k \text{ and } j = k \end{cases} \tag{16}$$

THEOREM 2 *Define a permutation matrix P whose elements are $P_{ij} = \delta_{p_i j}$. Define an upper triangular matrix U whose elements are $u_{ij} = a_{p_i j}^{(n)}$ if $j \geqq i$. Define a unit lower triangular matrix L whose elements are $\ell_{ij} = a_{p_i j}^{(n)}$ if $j < i$. Then $PA = LU$.*

Proof From the recursive formula, we have

$$u_{kj} = a_{p_k j}^{(n)} = a_{p_k j}^{(k)} \qquad (j \geqq k)$$

$$\ell_{ik} = a_{p_i k}^{(n)} = a_{p_i k}^{(k+1)} = a_{p_i k}^{(k)}/a_{p_k k}^{(k)} \qquad (i \geqq k)$$

These two equations depend on the fact that row p_k in $A^{(n)}$ became fixed in step k, and column k in $A^{(n)}$ became fixed in step $k+1$. Thus

$$a_{p_k j}^{(n)} = a_{p_k j}^{(k)} \quad \text{and} \quad a_{p_i k}^{(n)} = a_{p_i k}^{(k+1)}$$

Furthermore, the formula just given for ℓ_{ik} is valid when $i = k$, since the formula produces 1 for ℓ_{kk}. Now suppose that $i \leqq j$. Then

$$(LU)_{ij} = \sum_{k=1}^{i} \ell_{ik} u_{kj} = \sum_{k=1}^{i-1} \left(a_{p_i k}^{(k)}/a_{p_k k}^{(k)}\right) a_{p_k j}^{(k)} + \ell_{ii} a_{p_i j}^{(i)}$$

$$= \sum_{k=1}^{i-1} \left(a_{p_i j}^{(k)} - a_{p_i j}^{(k+1)}\right) + a_{p_i j}^{(i)} = a_{p_i j}^{(1)} = a_{p_i j}$$

If $i > j$, then

$$(LU)_{ij} = \sum_{k=1}^{j} \ell_{ik} u_{kj} = \sum_{k=1}^{j-1} \left(a_{p_i k}^{(k)}/a_{p_k k}^{(k)}\right) a_{p_k j}^{(k)} + \left(a_{p_i j}^{(j)}/a_{p_j j}^{(j)}\right) a_{p_j j}^{(j)}$$

$$= \sum_{k=1}^{j-1} \left(a_{p_i j}^{(k)} - a_{p_i j}^{(k+1)}\right) + a_{p_i j}^{(j)} = a_{p_i j}^{(1)} = a_{p_i j}$$

On the other hand,

$$(PA)_{ij} = \sum_{k=1}^{n} P_{ik}a_{kj} = \sum_{k=1}^{n} \delta_{p_i k}a_{kj} = a_{p_i j}$$

Thus we have proved, for all pairs (i, j), that

$$(PA)_{ij} = (LU)_{ij}$$ ∎

THEOREM 3 *If the factorization $PA = LU$ is produced from the Gaussian algorithm with scaled row pivoting, then the solution of $Ax = b$ is obtained by first solving $Lz = Pb$ and then solving $Ux = z$. Similarly, the solution of $y^T A = c^T$ is obtained by solving $U^T z = c$ and then $L^T Py = z$.*

Proof This is left as an exercise. ∎

The pseudocode for solving $Ax = b$ in terms of L and U looks like this:

input $n, (\ell_{ij}), (u_{ij}), (b_i), (p_i)$
for $i = 1, 2, \ldots, n$ **do**
$\quad z_i \leftarrow b_{p_i} - \sum_{j=1}^{i-1} \ell_{ij} z_j$
end
for $i = n, n-1, \ldots, 1$ **do**
$\quad x_i \leftarrow \left(z_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \Big/ u_{ii}$
end
output (x_i)

The same algorithm can be written using entries in the final A-matrix that resulted from the Gaussian process. The result is

input $n, (a_{ij}), (b_i), (p_i)$
for $i = 1, 2, \ldots, n$ **do**
$\quad z_i \leftarrow b_{p_i} - \sum_{j=1}^{i-1} a_{p_i j} z_j$
end
for $i = n, n-1, \ldots, 1$ **do**
$\quad x_i \leftarrow \left(z_i - \sum_{j=i+1}^{n} a_{p_i j} x_j \right) \Big/ a_{p_i i}$
end
output (x_i)

A pseudocode for solving $y^T A = c^T$ follows. Recall that $P^{-1} = P^T$.

input $n, (a_{ij}), (c_i), (p_i)$
for $j = 1, 2, \ldots, n$ **do**
$$z_j \leftarrow \left(c_j - \sum_{i=1}^{j-1} a_{p_i j} z_i \right) \Big/ a_{p_j j}$$
end
for $j = n, n-1, \ldots, 1$ **do**
$$y_{p_j} \leftarrow z_j - \sum_{i=j+1}^{n} a_{p_i j} y_{p_i}$$
end
output (y_i)

Operation Counts

To estimate the computing effort that goes into the solution of a system of linear equations, we shall count the number of arithmetic operations involved in the factorization phase and solution phase. Since the operations of multiplication and division are usually comparable in execution time and are much more time consuming than addition and subtraction, it has become traditional to count only multiplications and divisions, lumping them together as *long operations*, or *ops* for short.

Consider the first major step ($k = 1$) in the factorization process, as it operates on an $n \times n$ matrix A. The computation necessary to define p_1 (the index of the pivot row) involves n divisions (n ops). For each of the $n-1$ rows numbered $p_2, p_3, \ldots, p_n$, a multiplier is computed (1 op) and then a multiple of row p_1 is subtracted from row p_i for $2 \leq i \leq n$. The zeros being created in column one are *not* computed. Thus, the multiplier and the elimination process consume n ops per row. Since $n-1$ rows are processed in this way, we have $n(n-1)$ ops. To this are added the n ops needed for determining p_1. The total is n^2 ops.

The remainder of the factorization process can be interpreted as a repetition of step 1 on smaller and smaller matrices. Thus in step 2, the row p_1 is ignored and column 1 is ignored. The entire calculation in step 2 is like step 1 applied to an $(n-1) \times (n-1)$ matrix. This observation implies that the factorization requires

$$n^2 + (n-1)^2 + \ldots + 3^2 + 2^2 = \tfrac{1}{3}n^3 + \tfrac{1}{2}n^2 + \tfrac{1}{6}n - 1 \approx \tfrac{1}{3}n^3 + \tfrac{1}{2}n^2$$

long operations. Here we use the fact that $\sum_{k=1}^{n} k^2 = \frac{1}{6}n(n+1)(2n+1)$. For large n, the term $\frac{1}{3}n^3$ is the dominant one. Thus, finding the LU-factorization with scaled row pivoting involves roughly $\frac{1}{3}n^3$ long operations for large n.

An examination of the second phase of the algorithm shows that in updating the right-hand side b there are $n-1$ steps. In the first of these, there are $n-1$ long operations. In the second, there are $n-2$, and so on. The total is therefore

$$(n-1) + (n-2) + \cdots + 1 = \tfrac{1}{2}n^2 - \tfrac{1}{2}n$$

Here we use $\sum_{k=1}^{n} k = \frac{1}{2}n(n+1)$. In the back substitution, there is one long operation in the first step (computing x_n). Then there are successively $2, 3, \ldots, n$ long operations. The total is

$$1 + 2 + 3 + \cdots + n = \tfrac{1}{2}n^2 + \tfrac{1}{2}n$$

The grand total for this phase of the algorithm is therefore n^2. To summarize these findings, we state the following slightly more general result.

THEOREM 4 *If Gaussian elimination is used with scaled row pivoting, the solution of the system $Ax = b$ with fixed A and m different vectors b involves approximately*

$$\tfrac{1}{3}n^3 + (\tfrac{1}{2} + m)n^2$$

long operations (multiplications and divisions).

This organization of the Gaussian elimination algorithm permits an efficient treatment of the problem $Ax^{(i)} = b^{(i)}$ for $i = 1, 2, \ldots, m$. Here we have m linear systems, each with the same coefficient matrix but with different right-hand sides. One application of the forward elimination phase gives the factorization $PA = LU$. Then m "back solves" using this factorization are needed to obtain the $x^{(i)}$'s. Thus, only $O(\frac{1}{3}n^3 + (\frac{1}{2} + m)n^2)$ long operations are needed rather than $O(\frac{1}{3}mn^3)$ as would be the case if the m systems were treated separately. Moreover, we can compute A^{-1} in $O(\frac{4}{3}n^3)$ ops since it is a special case of the situation previously discussed; that is, $AX = I$ can be solved for each column of A^{-1} by solving $Ax^{(i)} = e_i$ using n values for i. Be advised that A^{-1} should *not* be computed when solving $Ax = b$, but rather x should be solved for directly!

Diagonally Dominant Matrices

Sometimes a system of equations has the property that Gaussian elimination *without* pivoting can be safely used. One class of matrices for which this is true is the class of **diagonally dominant** matrices. This property is expressed by the inequality

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad (1 \leqq i \leqq n) \tag{17}$$

If the coefficient matrix has this property, then in the first step of Gaussian elimination we can use row 1 as the pivot row since the pivot element a_{11} is not zero, by Inequality (17). After step 1 has been completed, we would like to know that row 2 can be used as the next pivot row. This situation is governed by the next theorem.

THEOREM 5 *Gaussian elimination without pivoting preserves the diagonal dominance of a matrix.*

Proof It suffices to consider the first step in Gaussian elimination (the step in which zeros are created in column 1), because subsequent steps mimic the first, except for being applied to matrices of smaller size.

Thus, let A be an $n \times n$ matrix that is diagonally dominant. Taking account of the zeros created in column 1 as well as the fact that row 1 is unchanged, we have to prove that, for $i = 2, 3, \ldots, n$,

$$|a_{ii}^{(2)}| > \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij}^{(2)}|$$

In terms of A, this means

$$|a_{ii} - (a_{i1}/a_{11})a_{1i}| > \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij} - (a_{i1}/a_{11})a_{1j}|$$

It suffices to prove the stronger inequality

$$|a_{ii}| - |(a_{i1}/a_{11})a_{1i}| > \sum_{\substack{j=2 \\ j\neq i}}^{n} \{|a_{ij}| + |(a_{i1}/a_{11})a_{1j}|\}$$

An equivalent inequality is

$$|a_{ii}| - \sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij}| > \sum_{j=2}^{n} |(a_{i1}/a_{11})a_{1j}|$$

From the diagonal dominance in the ith row, we know that

$$|a_{ii}| - \sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij}| > |a_{i1}|$$

Hence, it suffices to prove that

$$|a_{i1}| \geqq \sum_{j=2}^{n} |(a_{i1}/a_{11})a_{1j}|$$

This is true because of the diagonal dominance in row 1:

$$|a_{11}| > \sum_{j=2}^{n} |a_{1j}| \Rightarrow 1 > \sum_{j=2}^{n} |a_{1j}/a_{11}|$$ ∎

COROLLARY 1 *Every diagonally dominant matrix is nonsingular and has an LU-factorization.*

Proof The preceding theorem together with Theorem 1 implies that a diagonally dominant matrix A has an LU-decomposition in which L is unit lower triangular. The matrix U, by the preceding theorem, is diagonally dominant. Hence, its diagonal elements are nonzero. Thus, L and U are nonsingular. ∎

COROLLARY 2 *If the scaled row pivoting version of Gaussian elimination is applied to a diagonally dominant matrix, then the pivots will be the natural ones:* $1, 2, \ldots, n$. *Hence, the work of choosing the pivots can be omitted in this case.*

Proof By Theorem 5, it suffices to prove that the first pivot chosen in the algorithm is 1. Thus, it is enough to prove that

$$|a_{11}|/s_1 > |a_{i1}|/s_i \qquad (2 \leqq i \leqq n)$$

By the diagonal dominance, $|a_{ii}| = \max_j |a_{ij}| = s_i$ for all i. Hence, $|a_{11}|/s_1 = 1$. For $i \geqq 2$, we have

$$|a_{i1}| \leqq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| < |a_{ii}| = s_i$$

Thus, $|a_{i1}|/s_i < 1$. ∎

In Section 4.2, it was proved (Theorem 2) that a symmetric and positive definite matrix A has a unique Cholesky factorization, $A = LL^T$. It was shown also that the elements of the lower triangle matrix L satisfy the inequality

$$|\ell_{ij}| \leqq \sqrt{a_{ii}} \tag{18}$$

The Cholesky factorization algorithm is a special case of the LU-factorization. Pivoting is not needed in this particular case, because—as Inequality (18) indicates—the elements of L remain modest in magnitude compared to the elements of A.

Tridiagonal System

In applications, systems of equations often arise in which the coefficient matrix has a special structure. It is usually better to solve these systems using tailor-made algorithms that exploit the special structure. We consider one example of this—the tridiagonal system.

A square matrix $A = (a_{ij})$ is said to be **tridiagonal** if $a_{ij} = 0$ for all pairs (i, j) satisfying $|i - j| > 1$. Thus in the ith row, only $a_{i,i-1}$ and a_{ii} and $a_{i,i+1}$ can be different from zero. Three vectors can be used to store the nonzero elements, and we arrange the notation like this:

$$\begin{bmatrix} d_1 & c_1 & & & & & \\ a_1 & d_2 & c_2 & & & & \\ & a_2 & d_3 & c_3 & & & \\ & & a_3 & d_4 & c_4 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & a_{n-2} & d_{n-1} & c_{n-1} \\ & & & & & a_{n-1} & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} \tag{19}$$

In this display, matrix elements not shown are zeros.

We shall assume that the coefficient matrix is such that pivoting is not necessary in solving the system. This is true, for example, if the matrix is symmetric and positive definite. Simple Gaussian elimination will be used, and the right-hand side (vector b) is processed simultaneously in this algorithm. In step 1, a multiple of row 1 is subtracted from row 2 to produce a zero where a_1 stood originally. Note that d_2 and b_2 are altered but c_2 is not. The appropriate multiple is a_1/d_1. Hence, step one consists in these replacements:

$$\begin{aligned} d_2 &\leftarrow d_2 - (a_1/d_1)c_1 \\ b_2 &\leftarrow b_2 - (a_1/d_1)b_1 \end{aligned}$$

All the remaining steps in the forward elimination phase are exactly like the first. In the back substitution phase, the first step is

$$x_n \leftarrow b_n/d_n$$

The next step is

$$x_{n-1} \leftarrow (b_{n-1} - c_{n-1}x_n)/d_{n-1}$$

and all the rest are similar. Here is the algorithm:

input $n, (a_i), (b_i), (c_i), (d_i)$
for $i = 2, 3, \ldots, n$ **do**
$d_i \leftarrow d_i - (a_{i-1}/d_{i-1})c_{i-1}$
$b_i \leftarrow b_i - (a_{i-1}/d_{i-1})b_{i-1}$
end
$x_n \leftarrow b_n/d_n$
for $i = n-1, n-2, \ldots, 1$ **do**
$x_i \leftarrow (b_i - c_i x_{i+1})/d_i$
end
output (x_i)

PROBLEM SET 4.3

1. Solve the following linear systems twice. First, use Gaussian elimination and give the factorization $A = LU$. Second, use Gaussian elimination with scaled row pivoting and determine the factorization of the form $PA = LU$.

(a)
$$\begin{bmatrix} -1 & 1 & -4 \\ 2 & 2 & 0 \\ 3 & 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \frac{1}{2} \end{bmatrix}$$

(b)
$$\begin{bmatrix} 1 & 6 & 0 \\ 2 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix}$$

(c)
$$\begin{bmatrix} -1 & 1 & 0 & -3 \\ 1 & 0 & 3 & 1 \\ 0 & 1 & -1 & -1 \\ 3 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 3 \\ 1 \end{bmatrix}$$

(d)
$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 4 & 10 \\ 3 & -13 & 3 & 3 \\ -6 & 4 & 2 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -10 \\ -39 \\ -16 \end{bmatrix}$$

(e)
$$\begin{bmatrix} 1 & 0 & 2 & 1 \\ 4 & -9 & 2 & 1 \\ 8 & 16 & 6 & 5 \\ 2 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 14 \\ -3 \\ 0 \end{bmatrix}$$

2. Show that Equation (8) defining the Gaussian elimination algorithm can also be written in the form

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} & \text{if } i \leqq k \text{ or } j < k \\ a_{ij}^{(k)} - \left(a_{ik}^{(k)} / a_{kk}^{(k)}\right) a_{kj}^{(k)} & \text{if } i > k \text{ and } j \geqq k \end{cases}$$

3. Let $(p_1, p_2, \ldots, p_n)$ be a permutation of $(1, 2, \ldots, n)$ and define the matrix P by $P_{ij} = \delta_{p_i, j}$. Let A be an arbitrary $n \times n$ matrix. Describe PA, AP, P^{-1}, and PAP^{-1}.

[c]4. Gaussian elimination with *full* pivoting treats both rows and columns in an order different from the natural order. Thus, in the first step, the pivot element a_{ij} is chosen so that $|a_{ij}|$ is the largest in the entire matrix. This determines that row i will be the pivot row and column j will be the pivot column. Zeros are created in column j by subtracting multiples of row i from the other rows. Write the algorithm to carry out this process. Two permutation vectors will be required.

5. Let A be an $n \times n$ matrix with scale factors $s_i = \max_{1 \leqq j \leqq n} |a_{ij}|$. Assume that all s_i are positive, and let B be the matrix whose elements are (a_{ij}/s_i). Prove that if forward elimination is applied to A and to B, then the two final L-arrays are the same. Find the formulæthat relate the final A and B matrices (after processing).

6. It is sometimes advisable to modify a system of equations $Ax = b$ by introducing new variables $y_i = d_i x_i$, where d_i are positive numbers. If the x_i correspond to physical quantities, then this change of variables corresponds to a change in the units by which x_i is measured. Thus, if we decide to change x_1 from centimeters to meters, then $y_1 = 10^{-2} x_1$. In matrix terms, we define a diagonal matrix D with d_i as diagonal entries, and put $y = Dx$. The new system of equations is $AD^{-1}y = b$. If d_j is chosen as $\max_{1 \leqq i \leqq n} |a_{ij}|$, we call this **column equilibration**. Modify the factorization and solution algorithms to incorporate column equilibration. (The two algorithms together still will solve $Ax = b$.)

7. Show that the multipliers in the Gaussian algorithm with *full* pivoting (both row and column pivoting) lie in the interval $[-1, 1]$. (See Problem 4.)

8. Let the $n \times n$ matrix A be processed by forward elimination, with the resulting matrix called B, and permutation vector $p = (p_1, p_2, \ldots, p_n)$. Let P be the matrix that results from the identity matrix by writing its rows in the order $p_1, p_2, \ldots, p_n$. Prove that the LU-decomposition of PA is obtained as follows: Put $C = PB, L_{ij} = C_{ij}$ for $j < i$, and $U_{ij} = C_{ij}$ for $i \leqq j$. (Of course, $U_{ij} = 0$ if $i > j, L_{ij} = 0$ if $j > i$, and $L_{ii} = 1$.)

9. If the factor U in the LU-decomposition of A is known, what is the algorithm for calculating L?

10. Show how Gaussian elimination with scaled row pivoting works on this example (forward phase only).

$$\begin{bmatrix} 2 & -2 & -4 \\ 1 & 1 & -1 \\ 3 & 7 & 5 \end{bmatrix}$$

Display the scale array (s_1, s_2, s_3) and the *final* permutation array (p_1, p_2, p_3). Show the final A-array, with the multipliers stored in the correct locations.

11. Carry out the instructions in Problem 10 on the matrix

$$\begin{bmatrix} 3 & 7 & 3 \\ 1 & \frac{7}{3} & 4 \\ 4 & \frac{4}{3} & 0 \end{bmatrix}$$

12. Assume that A is tridiagonal. Define $c_0 = 0$ and $a_n = 0$. Show that if A is *columnwise* diagonally dominant

$$|d_i| > |a_i| + |c_{i-1}| \qquad (1 \leqq i \leqq n)$$

then the algorithm for tridiagonal systems will, in theory, be successful since no zero pivot entries will be encountered. Refer to Equation (19) for the notation.

13. Write a special Gaussian elimination algorithm to solve linear equations when A has the property $a_{ij} = 0$ for $i > j+1$. Do not use pivoting. Include the processing of the right-hand side in the algorithm. Count the operations needed to solve $Ax = b$.

14. Count the operations in the algorithm in the text for tridiagonal systems.

15. Rewrite the algorithm for tridiagonal systems so that the order of processing the equations and variables is reversed.

16. Prove the theorem concerning the number of long operations in Gaussian elimination.

17. Show how Gaussian elimination with scaled row pivoting works on this example:

$$A = \begin{bmatrix} -9 & 1 & 17 \\ 3 & 2 & -1 \\ 6 & 8 & 1 \end{bmatrix}$$

Show the scale array. The final A-array should contain the multipliers stored in the correct positions. Determine P, L, and U, and verify that $PA = LU$.

18. Show how the factorization phase of Gaussian elimination with scaled row pivoting works on the matrix

$$\begin{bmatrix} 1 & -2 & 3 \\ 2 & -4 & 2 \\ 3 & -5 & -1 \end{bmatrix}$$

Show all intermediate steps—that is, multipliers, scale array s, and index array p—and the final array A as it would appear after the algorithm had finished working on it.

19. This problem shows how the solution to a system of equations can be *unstable* relative to perturbations in the data. Solve $Ax = b$ with $b = (100,\ 1)^T$, and with each of the following matrices. (See Stoer and Bulirsch [1980, p. 185].)

$$A_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0.01 \end{bmatrix}$$

20. Assume that $0 < \varepsilon < 2^{-22}$. If the Gaussian algorithm without pivoting is used to solve the system

$$\begin{cases} \varepsilon x_1 + 2x_2 = 4 \\ x_1 - x_2 = -1 \end{cases}$$

on the MARC-32, what will be the solution vector (x_1, x_2)?

21. Solve the following system by Gaussian elimination with full pivoting (as described in Problem 4):

$$\begin{bmatrix} -9 & 1 & 17 \\ 3 & 2 & -1 \\ 6 & 8 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 9 \\ -3 \end{bmatrix}$$

22. Solve the system

$$\begin{cases} 0.2641x_1 + 0.1735x_2 + 0.8642x_3 = -0.7521 \\ 0.9411x_1 + 0.0175x_2 + 0.1463x_3 = 0.6310 \\ -0.8641x_1 - 0.4243x_2 + 0.0711x_3 = 0.2501 \end{cases}$$

using Gaussian elimination with: (a) no pivoting, and (b) scaled row pivoting.

23. Write an algorithm to solve the system $Ax = b$ under the following conditions: There is a permutation $(p_1, p_2, \ldots, p_n)$ of $(1, 2, \ldots, n)$ such that for each i, equation p_i contains only the variable x_i.

24. Repeat the preceding problem assuming that for each i, equation i contains only the variable x_{p_i}.

25. Repeat the preceding problem assuming that for each i, equation p_i contains only the variable x_{p_i}. In this case, give the *simplest* algorithm.

26. (a) Show that if we apply Gaussian elimination without pivoting to a symmetric matrix A, then $\ell_{i1} = a_{1i}/a_{11}$.
(b) From this, show that if the first row and column of $A^{(2)}$ are removed, the remaining $(n-1) \times (n-1)$ matrix is symmetric. Conclude then that elements below the diagonal in this smaller matrix need not be computed. Use induction to infer that this simplification will occur in each succeeding step of the factorization phase.
(c) Show that the computation required is almost halved compared to the non-symmetric case.
(d) Use this simplification to solve the system

$$\begin{cases} 0.6428x_1 + 0.3475x_2 - 0.8468x_3 = 0.4127 \\ 0.3475x_1 + 1.8423x_2 + 0.4759x_3 = 1.7321 \\ -0.8468x_1 + 0.4759x_2 + 1.2147x_3 = -0.8621 \end{cases}$$

27. Consider the matrix

$$\begin{bmatrix} 0 & 4 & 25 & 79 \\ 9 & 7 & 39 & 89 \\ 0 & 16 & 2 & 99 \\ 0 & 6 & 6 & 49 \end{bmatrix}$$

Circle the entry that will be used as the next pivot element in Gaussian elimination with scaled row pivoting. The scale array is $s = (80, 89, 160, 30)$.

28. Show the resulting matrix after the forward phase of Gaussian elimination with scaled row pivoting is applied to the matrix

$$\begin{bmatrix} 2 & -2 & -4 \\ 1 & 1 & -1 \\ 3 & 7 & 5 \end{bmatrix}$$

In the final matrix, write the multipliers in the appropriate locations.

29. Determine $\det(A)$ where

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

without computing the determinant by expansion by minors.

30. Use Gaussian elimination with scaled row pivoting to find the determinant of

$$A = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 2 & 0 \\ 2 & 0 & 1 & 0 \end{bmatrix}$$

31. Consider the system

$$\begin{cases} x_2 + 2x_3 = 1 \\ 2x_1 - x_2 = 2 \\ 2x_2 + x_3 = 3 \end{cases}$$

Determine the factorization $PA = LU$ where P is a permutation matrix. Use this factorization to obtain $\det(A)$.

32. Consider

$$A = \begin{bmatrix} 3 & 2 & -1 \\ 6 & 6 & 2 \\ -1 & 1 & 3 \end{bmatrix}$$

Use Gaussian elimination with scaled row pivoting to obtain the factorization

$$PA = LDU$$

where L is a unit lower triangular matrix, U is a unit upper triangular matrix, D is a diagonal matrix, and P is a permutation matrix.

33. In the next few problems, we fix n and denote by J the set $\{1, 2, \ldots, n\}$. A **permutation** of J is a map $p : J \twoheadrightarrow J$, where the double arrow indicates that p is *surjective*. Thus each element of J is the image, $p(i)$, of some element i in J. The **identity** permutation is defined by $u(i) = i$ for all $i \in J$. Show that if p and q are permutations of J, then so is $p \circ q$, which is defined as usual by the equation $(p \circ q)(i) = p(q(i))$. Prove that $p \circ (q \circ r) = (p \circ q) \circ r$ and that $p \circ u = u \circ p = p$.

34. (Continuation) Prove that each permutation p has an inverse p^{-1}, which satisfies $p \circ p^{-1} = u = p^{-1} \circ p$. The set of all permutations of J is called the **symmetric group** on J.

35. (Continuation) Give an algorithm for determining the inverse of any given permutation. (A permutation of J can be represented as a vector $(p(1), p(2), \ldots, p(n))$ in a computer.)

36. Let a system of equations, $Ax = b$, be given, in which A is $n \times n$. Let p and q be permutations of $\{1, 2, \ldots, n\}$. Write an algorithm for solving the system under the assumption that for $i = 1, 2, \ldots, n$, the equation numbered p_i contains only the variable x_{q_i}.

37. (Continuation) Repeat the preceding problem under the assumption that for each i, the variables $x_{q_1}, x_{q_2}, \ldots, x_{q_{i-1}}$ do not appear in the equation numbered p_i.
38. (Continuation) Repeat Problem 36 under the assumption that for each i, the variable x_{q_i} occurs only in the equations numbered $p_1, p_2, \ldots, p_i$.
39. (Difficult) Find an algorithm to solve $Ax = b$ under the assumption that all elements a_{ij} are zero unless $|i - j| \leqq 1$ or $(i, j) = (1, n)$ or $(i, j) = (n, 1)$. Use Gaussian elimination without pivoting.
40. Count the number of long operations involved in the LU-factorization of an $n \times n$ matrix, assuming that no pivoting is employed.
41. Let A be an $n \times n$ matrix that is diagonally dominant in its columns. Thus

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| < |a_{jj}| \qquad (1 \leqq j \leqq n)$$

Determine whether Gaussian elimination without pivoting preserves this diagonal dominance.

42. In a diagonally dominant matrix A, define the *excess in row* i by the equation

$$e_i = |a_{ii}| - \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

Show that in the proof of Theorem 5 the following is true:

$$|a_{ii} - a_{i1}a_{1i}/a_{11}| \geqq \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij} - a_{i1}a_{1j}/a_{11}| + e_i$$

Thus, the excess in row i is not diminished in Gaussian elimination.

[c]43. Refer to Problem 26, and write a program to carry out the factorization phase of Gaussian elimination on a symmetric matrix. Assume that pivoting is not required.

44. Refer to Problems 33–35 if necessary. Let p be a permutation of $\{1, 2, \ldots, n\}$, and let P be the corresponding permutation matrix. (Thus $P_{ij} = \delta_{p_i j}$.) Let q be the inverse of p and Q the permutation matrix corresponding to q. Prove that $P^{-1} = Q$.
45. (Continuation) Prove that if P is a permutation matrix, then $P^{-1} = P^T$.
46. If A is $n \times n$ and B is $n \times m$, how many multiplications and divisions are required to solve $AX = B$ by Gaussian elimination with scaled row pivoting? What if $B = I$?
47. Prove or disprove: If A is tridiagonal and P is a permutation matrix, then PAP^{-1} is tridiagonal.
48. Suppose that the scale array is recomputed in each major step of Gaussian elimination with scaled row pivoting. Prove that for a symmetric and diagonally dominant matrix, Gaussian elimination without pivoting is the same as with scaled row pivoting.
49. Prove Theorem 3.

[c]50. Write and test programs to carry out Gaussian elimination with scaled row pivoting. Suitable test cases are in Problems 19, 21, 22, and 26.

51. In the scaled row pivoting algorithm for Gaussian elimination, suppose that the scale numbers s_i are redefined by

$$s_i = |a_{i1}| + |a_{i2}| + \cdots + |a_{in}|$$

Prove that if the resulting algorithm is applied to a diagonally dominant matrix then the natural pivot order $(1, 2, \ldots, n)$ will be chosen. Write and test programs to carry out the ideas in Problem 26. This is Gaussian elimination without pivoting on a symmetric system.

[c]52. Write and test programs that include column equilibration (Problem 6) in the Gaussian algorithms.

[c]53. Write and test programs to solve $Ax = b$ and $y^T A = c^T$ using only one factorization of A (with scaled row pivoting) and two other subprograms to solve for x and y.

[c]54. Write and test programs to solve $Ax = b$ using column equilibration, row equilibration, and full pivoting. (Refer to Problems 4 and 6 for the terminology.)

[c]55. Write a subroutine `GAUSSJ` (N, A, B, X, P, S, D) that solves an $n \times n$ system $Ax = b$ by the **Gauss-Jordan** method, with column equilibration and scaled row pivoting. In the Gauss-Jordan algorithm (without pivoting) at the kth major step, multiples of row k are subtracted from *all* the other rows so that the coefficient of x_k is 0 in all rows except the kth row. At the end, the matrix will be a diagonal matrix (rather than an upper triangular matrix, as in the Gaussian elimination method). With scaled row pivoting, row p_k is used as pivot row to produce 0 coefficients of x_k in all rows except row p_k. Column equilibration, as discussed in Problem 6, should be carried out at the beginning. The divisors needed in this process should be stored in array D, since they will be needed at the end in obtaining x.

56. Prove or disprove the natural conjecture that if a matrix has the property

$$0 \neq |a_{ii}| \geqq \sum_{\substack{j=1 \\ j \neq i}} |a_{ij}| \qquad (1 \leqq i \leqq n)$$

then the Gaussian elimination without pivoting will preserve this property.

57. (a) Prove that computing the determinant of a matrix by expansion by minors involves $(n - 1)(n!)$ ops.

(b) Prove that Cramer's rule requires $(n^2 - 1)(n!)$ ops.

(c) Prove that the Gauss-Jordan method involves $\frac{1}{2}n(n + 1)^2 \approx \frac{1}{2}n^3$ ops and therefore is 50 percent more expensive than Gaussian elimination.

4.4 Norms and the Analysis of Errors

To discuss the errors in numerical problems involving vectors, it is useful to employ *norms*. Our vectors are usually in one of the spaces $\mathbb{R}^n$, but a norm can be defined on any vector space.

Vector Norms

On a vector space V, a **norm** is a function $\|\cdot\|$ from V to the set of nonnegative reals that obeys these three postulates:

$$\|x\| > 0 \qquad \text{if } x \neq 0,\ x \in V \tag{1}$$

$$\|\lambda x\| = |\lambda|\ \|x\| \qquad \text{if } \lambda \in \mathbb{R},\ x \in V \tag{2}$$

$$\|x + y\| \leqq \|x\| + \|y\| \qquad \text{if } x, y \in V \quad \text{(triangle inequality)} \tag{3}$$

We can think of $\|x\|$ as the *length* or *magnitude* of the vector x. A norm on a vector space generalizes the notion of absolute value, $|r|$, for a real or complex number. The most familiar norm on $\mathbb{R}^n$ is the **Euclidean norm** defined by

$$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2} \qquad \text{where} \quad x = (x_1, x_2, \ldots, x_n)^T$$

This is the norm that corresponds to our intuitive concept of length. We use the subscript 2 as an identifier only. In numerical analysis, other norms are also used; the simplest and easiest to compute is this one, called the ℓ_∞*-norm*:

$$\|x\|_\infty = \max_{1 \leqq i \leqq n} |x_i| \tag{4}$$

Again, a subscript is used to distinguish this norm from others. A third important example of a norm on $\mathbb{R}^n$ is called the ℓ_1*-norm*:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \tag{5}$$

Example 1 Using the norm $\|\cdot\|_1$, compare the lengths of the following three vectors in $\mathbb{R}^4$. Then repeat the calculation for the norms $\|\cdot\|_2$ and $\|\cdot\|_\infty$.

$$x = (4, 4, -4, 4)^T \qquad v = (0, 5, 5, 5)^T \qquad w = (6, 0, 0, 0)^T$$

Solution The results are displayed in the following table.

	$\|\cdot\|_1$	$\|\cdot\|_2$	$\|\cdot\|_\infty$
x	16.	8.	4.
v	15.	8.66	5.
w	6.	6.	6.

■

To understand these norms better, it is instructive to consider $\mathbb{R}^2$. For the three norms given above, we give sketches in Figure 4.1 of the set

$$\{x : x \in \mathbb{R}^2, \|x\| \leqq 1\}$$

This set is called the **unit cell** or the **unit ball** in two-dimensional vector space.

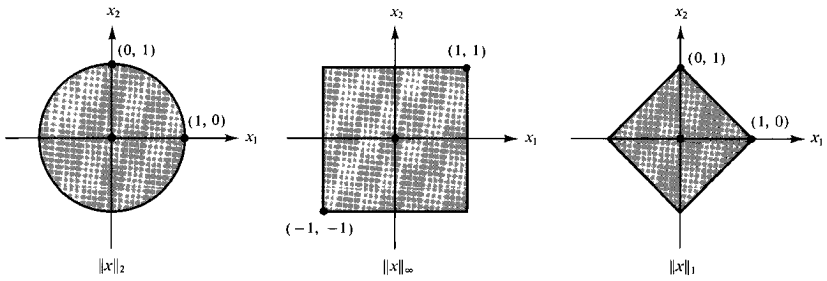


Figure 4.1 Unit cells in $\mathbb{R}^2$ for three norms

Matrix Norms

Now we turn to the question of defining norms for matrices. Although we could deal with general matrix norms, subjecting them only to the same requirements (1)–(3), we usually prefer a matrix norm to be intimately related to a vector norm. If a vector norm $\|\cdot\|$ has been specified, the matrix norm **subordinate** to it is defined by

$$\|A\| = \sup\{\|Au\| : u \in \mathbb{R}^n, \|u\| = 1\} \tag{6}$$

This is also called the matrix norm **associated** with the given vector norm. Here A is understood to be an $n \times n$ matrix.

THEOREM 1 *If $\|\cdot\|$ is any norm on $\mathbb{R}^n$, then Equation (6) defines a norm on the linear space of all $n \times n$ matrices.*

Proof We shall verify the three axioms for a norm. First, if $A \neq 0$, then A has at least one nonzero column; say, $A^{(j)} \neq 0$. Consider the vector in which 1 is the jth component—that is, $x = (0, \ldots, 0, 1, 0, \ldots, 0)^T$. Obviously, $x \neq 0$ and the vector $v = x/\|x\|$ is of norm 1. Hence by the definition of $\|A\|$,

$$\|A\| \geq \|Av\| = \frac{\|Ax\|}{\|x\|} = \frac{\|A^{(j)}\|}{\|x\|} > 0$$

Next, from Property (2) of the vector norm, we have

$$\|\lambda A\| = \sup\{\|\lambda Au\| : \|u\| = 1\} = |\lambda| \sup\{\|Au\| : \|u\| = 1\} = |\lambda| \, \|A\|$$

For the triangle inequality, we use the analogous property of the vector norm and Problem 4 to write

$$\|A + B\| = \sup\{\|(A + B)u\| : \|u\| = 1\}$$

$$\begin{aligned} &\leqq \sup\{\|Au\| + \|Bu\| : \|u\| = 1\} \\ &\leqq \sup\{\|Au\| : \|u\| = 1\} + \sup\{\|Bu\| : \|u\| = 1\} \\ &= \|A\| + \|B\| \end{aligned}$$

∎

An important consequence of Definition (6)—and indeed the motivation for the definition—is that

$$\|Ax\| \leqq \|A\|\,\|x\| \qquad (x \in \mathbb{R}^n) \tag{7}$$

To prove this, observe that it is true for $x = 0$. If $x \neq 0$, then the vector $v = x/\|x\|$ is of norm 1. Hence, from (6)

$$\|A\| \geqq \|Av\| = \frac{\|Ax\|}{\|x\|}$$

As an illustration of this important concept, let us fix as our vector norm the norm $\|x\|_\infty$, defined in Equation (4). What is its subordinate matrix norm? Here is the calculation:

$$\begin{aligned} \|A\|_\infty &= \sup_{\|u\|_\infty = 1} \|Au\|_\infty = \sup_{\|u\|_\infty = 1} \left\{ \max_{1 \leqq i \leqq n} |(Au)_i| \right\} = \max_{1 \leqq i \leqq n} \left\{ \sup_{\|u\|_\infty = 1} |(Au)_i| \right\} \\ &= \max_{1 \leqq i \leqq n} \left\{ \sup_{\|u\|_\infty = 1} \left| \sum_{j=1}^{n} a_{ij} u_j \right| \right\} = \max_{1 \leqq i \leqq n} \sum_{j=1}^{n} |a_{ij}| \end{aligned} \tag{8}$$

Here we use the fact (Problem 9) that two maximization processes can be interchanged. Also used is the fact that the supremum of $|\sum_{j=1}^{n} a_{ij} u_j|$ for fixed i and $\|u\|_\infty = 1$ is obtained by putting $u_j = +1$ if $a_{ij} \geqq 0$ and $u_j = -1$ if $a_{ij} < 0$.

Thus, we have proved the following theorem

THEOREM 2 *If the vector norm $\|\cdot\|_\infty$ is defined by*

$$\|x\|_\infty = \max_{1 \leqq i \leqq n} |x_i|$$

then its subordinate matrix norm is given by

$$\|A\|_\infty = \max_{1 \leqq i \leqq n} \sum_{j=1}^{n} |a_{ij}|$$

A matrix norm subordinate to a vector norm has additional properties besides the basic ones (1)–(3). For example:

$$\|I\| = 1 \tag{9}$$

$$\|AB\| \leqq \|A\|\,\|B\| \tag{10}$$

The proof of (9) is immediate from the definition (6), and Equation (10) follows from Equations (6) and (7).

Condition Number

Now let us put these concepts to work. We consider an equation

$$Ax = b$$

where A is an $n \times n$ matrix. Suppose that A is invertible.

Example 2 If A^{-1} is perturbed to obtain a new matrix B, the solution $x = A^{-1}b$ is perturbed to become a new vector $\tilde{x} = Bb$. How large is this latter perturbation in absolute and relative terms?

Solution We have, using any vector norm and its subordinate matrix norm:

$$\|x - \tilde{x}\| = \|x - Bb\| = \|x - BAx\| = \|(I - BA)x\| \leqq \|I - BA\| \, \|x\|$$

This gives the magnitude of the perturbation in x. If the relative perturbation is being measured, we can write

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leqq \|I - BA\| \tag{11}$$

Inequality (11) gives an upper bound on $\|x - \tilde{x}\|/\|x\|$, and this ratio is taken as a measure of the *relative* error between x and $\tilde{x}$. ∎

Example 3 Suppose that the vector b is perturbed to obtain a vector $\tilde{b}$. If x and $\tilde{x}$ satisfy $Ax = b$ and $A\tilde{x} = \tilde{b}$, by how much do x and $\tilde{x}$ differ, in absolute and relative terms?

Solution Assuming that A is invertible, we have

$$\begin{aligned} \|x - \tilde{x}\| &= \|A^{-1}b - A^{-1}\tilde{b}\| = \|A^{-1}(b - \tilde{b})\| \\ &\leqq \|A^{-1}\| \, \|b - \tilde{b}\| \end{aligned}$$

This gives a measure of the perturbation in x. To estimate the relative perturbation, we write

$$\begin{aligned} \|x - \tilde{x}\| &\leqq \|A^{-1}\| \, \|b - \tilde{b}\| = \|A^{-1}\| \, \|Ax\| \, \frac{\|b - \tilde{b}\|}{\|b\|} \\ &\leqq \|A^{-1}\| \, \|A\| \, \|x\| \, \frac{\|b - \tilde{b}\|}{\|b\|} \end{aligned}$$

Hence

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leqq \|A\| \, \|A^{-1}\| \, \frac{\|b - \tilde{b}\|}{\|b\|} \tag{12}$$

∎

Inequality (12) tells us that the relative error in x is no greater than $\kappa(A)$ times the relative error in b, where

$$\kappa(A) \equiv \|A\| \, \|A^{-1}\| \tag{13}$$

This number $\kappa(A)$ is called a **condition number** of the matrix A. It depends on the vector norm chosen at the beginning of the analysis. From Inequality (12), we see that if the condition number is small, then small perturbations in b lead to small perturbations in x. The inequality $\kappa(A) \geqq 1$ is always true. (See Problem 13.)

For an example to illustrate the condition number, let $\varepsilon > 0$ and

$$A = \begin{bmatrix} 1 & 1+\varepsilon \\ 1-\varepsilon & 1 \end{bmatrix} \qquad A^{-1} = \varepsilon^{-2} \begin{bmatrix} 1 & -1-\varepsilon \\ -1+\varepsilon & 1 \end{bmatrix}$$

If the ∞-norm is employed, we have $\|A\|_\infty = 2+\varepsilon$ and $\|A^{-1}\|_\infty = \varepsilon^{-2}(2+\varepsilon)$, by Equation (8). Hence, $\kappa(A) = [(2+\varepsilon)/\varepsilon]^2 > 4/\varepsilon^2$. If $\varepsilon \leqq 0.01$, then $\kappa(A) \geqq 40,000$. In such a case, a small relative perturbation in b may induce a relative perturbation *40,000 times greater* in the solution of the system $Ax = b$.

If we solve a system of equations

$$Ax = b$$

numerically, we obtain not the exact solution x but an approximate solution $\tilde{x}$. One can test $\tilde{x}$ by forming $A\tilde{x}$ to see whether it is close to b. Thus, we have the **residual vector**

$$r = b - A\tilde{x}$$

The difference between the exact solution x and the approximate solution $\tilde{x}$ is called the **error vector**

$$e = x - \tilde{x}$$

The following relationship

$$Ae = r \tag{14}$$

between the error vector and the residual vector is of fundamental importance.

Notice that $\tilde{x}$ is the exact solution of the linear system $A\tilde{x} = \tilde{b}$, which has a perturbed right-hand side $\tilde{b} = b - r$. We now establish relationships between the relative errors in x and b. In other words, we want to relate $\|x - \tilde{x}\|/\|x\|$ to $\|b - \tilde{b}\|/\|b\| = \|r\|/\|b\|$. The following theorem shows that the condition number $\kappa(A)$ plays an important role.

THEOREM 3

$$\frac{1}{\kappa(A)} \frac{\|r\|}{\|b\|} \leqq \frac{\|e\|}{\|x\|} \leqq \kappa(A) \frac{\|r\|}{\|b\|}$$

Proof The inequality on the right can be written as

$$\|e\| \, \|b\| \leqq \|A\| \, \|A^{-1}\| \, \|r\| \, \|x\|$$

and this is true since

$$\|e\| \, \|b\| = \|A^{-1}r\| \, \|Ax\| \leqq \|A^{-1}\| \, \|r\| \, \|A\| \, \|x\|$$

(In fact, the inequality on the right in the theorem is Inequality (12).) The inequality on the left can be written as

$$\|r\| \, \|x\| \leqq \|A\| \, \|A^{-1}\| \, \|b\| \, \|e\|$$

and this follows at once from

$$\|r\| \, \|x\| = \|Ae\| \, \|A^{-1}b\| \leqq \|A\| \, \|e\| \, \|A^{-1}\| \, \|b\| \qquad \blacksquare$$

A matrix with a large condition number is said to be **ill conditioned**. For an ill-conditioned matrix A, there will be cases in which the solution of a system $Ax = b$ will be very sensitive to small changes in the vector b. In other words, to attain a certain precision in the determination of x, we shall require significantly higher precision in b. If the condition number of A is of moderate size, the matrix is said to be **well conditioned**.

PROBLEM SET 4.4

1. Show that the norms $\|x\|_\infty, \|x\|_2, \|x\|_1$ satisfy the postulates (1), (2), (3) for norms.
2. Show that $\|x\|_\infty \leqq \|x\|_2 \leqq \|x\|_1$ for all $x \in \mathbb{R}^n$, and that equalities can occur, even for nonzero vectors.
3. Show that $\|x\|_1 \leqq n\|x\|_\infty$ and $\|x\|_2 \leqq \sqrt{n}\,\|x\|_\infty$ for $x \in \mathbb{R}^n$.
4. Show that for any two functions into $\mathbb{R}$,

$$\sup[f(x) + g(x)] \leqq \sup f(x) + \sup g(x)$$

5. For the matrix norm in Theorem 2, prove or disprove: $\|AB\|_\infty = \|A\|_\infty \|B\|_\infty$. What about the special case $\|A^2\|_\infty = \|A\|_\infty^2$?
6. Show that a norm defined on $\mathbb{R}^n$ must involve all components of a vector in some way. Show that the norm $\|x\|_\infty$ is indeed the simplest on $\mathbb{R}^n$. (You will have to define a suitable concept of *simplicity*.)
7. Determine whether these expressions define norms on $\mathbb{R}^n$:
   (a) $\max\{|x_2|, |x_3|, \ldots, |x_n|\}$
   (b) $\sum_{i=1}^{n} |x_i|^3$
   (c) $\left\{ \sum_{i=1}^{n} |x_i|^{1/2} \right\}^2$
   (d) $\max\{|x_1 - x_2|, |x_1 + x_2|, |x_3|, |x_4|, \ldots, |x_n|\}$
   (e) $\sum_{i=1}^{n} 2^{-i}|x_i|$
8. Define $\|A\| = \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|$. Show that this is a matrix norm (that is, a norm on the linear space of all $n \times n$ matrices). Show that it is not subordinate to any vector norm. Does it conform to Equations (9) and (10)?
9. (a) Prove that if A and B are arbitrary sets and f is a bounded real function on $A \times B$, then

$$\sup_{a \in A} \sup_{b \in B} f(a, b) = \sup_{b \in B} \sup_{a \in A} f(a, b)$$

(b) Show by example that a supremum and an infimum cannot be interchanged in general.

(c) Show that

$$\sup_{a\in A}\inf_{b\in B} f(a,b) \leqq \inf_{b\in B}\sup_{a\in A} f(a,b)$$

10. Prove that for any vector norm and its subordinate matrix norm, and for any $n \times n$ matrix A, there corresponds a vector $x \neq 0$ such that $\|Ax\| = \|A\|\|x\|$.
11. Show that for the vector norm $\|x\|_1$ defined in Equation (5), the subordinate matrix norm is

$$\|A\|_1 = \max_{1\leqq j\leqq n}\sum_{i=1}^{n}|a_{ij}|$$

12. Do the subordinate matrix norms satisfy $\|AB\| = \|BA\|$? Explain.
13. Prove that the condition number of an invertible matrix must be at least 1.
14. What matrices have condition number equal to 1?
15. Using the one matrix norm (Problem 11), compute the condition number of the matrix $\begin{bmatrix} 1 & 1+\varepsilon \\ 1-\varepsilon & 1 \end{bmatrix}$.
16. Using the infinity matrix norm, Equation (8), compute the condition number of the matrix $\begin{bmatrix} 7 & 8 \\ 9 & 10 \end{bmatrix}$.
17. Let A be an $n \times n$ matrix with inverse $C = (c_{ij})$. Show that in solving $Ax = b$, a perturbation of amount δ in b_j will cause a perturbation of $c_{ij}\delta$ in x_i.
18. (Continuation) Prove that a perturbation of amount δ in a_{jk} will produce a perturbation of approximately $-c_{ij}x_k\delta$ in x_i.
19. (Continuation) The following quantity is sometimes used for a condition number of an $n \times n$ matrix A:

$$M(A) = n \max_{1\leqq i,j\leqq n}|a_{ij}| \max_{1\leqq i,j\leqq n}|c_{ij}|$$

where C is the inverse of A. Prove that if $Ax = b$ and if only one component of b is perturbed (say by an amount ε), then the perturbed x (denoted by $\widetilde{x}$) satisfies

$$\frac{\|x-\widetilde{x}\|_\infty}{\|x\|_\infty} \leqq M(A)\frac{|\varepsilon|}{\|b\|_\infty}$$

20. It is proved in the text that if $Ax = b$ and $A\widetilde{x} = \widetilde{b}$, then

$$\frac{\|x-\widetilde{x}\|}{\|x\|} \leqq \kappa(A)\frac{\|b-\widetilde{b}\|}{\|b\|}$$

Prove that for every nonsingular matrix A, this inequality will become an equality for some vectors b and $\widetilde{b}$. (Of course, we want $b \neq 0$ and $b \neq \widetilde{b}$.)

21. Let $n = 3$ and let

$$A = \begin{bmatrix} 4 & -3 & 2 \\ -1 & 0 & 5 \\ 2 & 6 & -2 \end{bmatrix}$$

Among all the vectors x satisfying $\|x\|_\infty \leqq 1$, find one for which $\|Ax\|_\infty$ is as large as possible. Also give the numerical value of $\|A\|_\infty$.

22. A *weighted* ℓ_∞-norm is a norm on $\mathbb{R}^n$ of the form

$$\|x\| = \max_{1 \leqq i \leqq n} w_i |x_i|$$

where $w_1, w_2, \ldots, w_n$ are fixed positive numbers called *weights*. Prove the norm postulates for this norm. What is the subordinate matrix norm?

23. Prove that if $\|\cdot\|$ is a norm on a vector space, and if we define $\|x\|' = \alpha\|x\|$ with a fixed positive α, then $\|\cdot\|'$ is also a norm.

24. If the construction in the preceding problem is applied to a subordinate matrix norm, is the resulting norm also a subordinate matrix norm?

25. Let $\|\cdot\|$ be a norm on a vector space V. For x and y in V, let $d(x, y) = \|x - y\|$. Show that d has these properties

(a) $d(x, x) = 0$

(b) $d(x, y) = d(y, x)$

(c) $d(x, y) > 0$ if $x \neq y$

(d) $d(x, y) \leqq d(x, z) + d(z, y)$

(A function having these four properties is called a *metric*.)

C 26. Write subroutines for computing the norm of a vector and a square matrix. Use the maximum norm, $\|x\|_\infty$, and its subordinate matrix norm.

27. Give an example of a well-conditioned matrix whose determinant is very small.

28. Compute the condition number of the $n \times n$ lower triangular matrix that has positive ones on the diagonal and negative ones below the diagonal. Use the matrix norm $\|\cdot\|_\infty$.

29. Prove that if A has a nontrivial fixed point (that is, $Ax = x \neq 0$), then $\|A\| \geqq 1$, for any subordinate matrix norm.

30. Give an example of a norm on $\mathbb{R}^2$ such that the norm of $(1, 0)$ is 2 and the norm of $(1,1)$ is 1.

31. Is there a norm on $\mathbb{R}^2$ such that $\|(1, 0)\| = \|(0, 1)\| = \|(\frac{1}{3}, \frac{1}{3})\|$?

32. (See Problems 2 and 3.) Find the precise conditions on x in order that

(a) $\|x\|_\infty = \|x\|_1$ (b) $\|x\|_\infty = \|x\|_2$ (c) $\|x\|_1 = \|x\|_2$

33. Show that $\|A\|$ is the smallest number M such that $\|Ax\| \leqq M\|x\|$ for all x.

34. For any $n \times n$ matrix A, define $\|A\|_F = (\sum_{i=1}^n \sum_{j=1}^n a_{ij}^2)^{1/2}$. (This is called the **Frobenius norm**.) Is this a subordinate matrix norm? Answer the same question for $\|A\| = \max_{1 \leqq i,j \leqq n} |a_{ij}|$. Prove that these equations define norms on the vector space of all $n \times n$ matrices.

35. Is it necessarily true, for any subordinate matrix norm, that the norm of a permutation matrix is unity? Explain.

36. For a vector $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$, we define $|x|$, the absolute value of the vector x, to be the vector $(|x_1|, |x_2|, \ldots, |x_n|)^T$. For vectors x and y, we also define $x \leqq y$ to mean that $x_i \leqq y_i$ for $i = 1, 2, \ldots, n$. Prove that the norms $\|\cdot\|_1, \|\cdot\|_2$, and $\|\cdot\|_\infty$ have the property that if $|x| \leqq |y|$ then $\|x\| \leqq \|y\|$.

37. For any real number $p \geqq 1$, the formula

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

defines a norm. (For the proof, consult Bartle [1976, p. 61].) Prove that for each $x \in \mathbb{R}^n$

$$\lim_{p \to \infty} \|x\|_p = \|x\|_\infty$$

This explains why the notation $\| \cdot \|_\infty$ is used.

38. Prove these properties of norms:
    (a) $\|0\| = 0$
    (b) $\|x + y\| \geqq | \, \|x\| - \|y\| \, |$
    (c) $\left\| \sum_{i=1}^{m} x^{(i)} \right\| \leqq \sum_{i=1}^{m} \|x^{(i)}\|$ for vectors $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

39. Let $\| \cdot \|$ be a norm on $\mathbb{R}^n$. Define

$$\|x\|' = \sup \left\{ u^T x : u \in \mathbb{R}^n \ , \ \|u\| = 1 \right\}$$

Prove that this equation defines a norm. Prove that if the process is repeated, then the original norm is obtained; that is, $(\| \cdot \|')' = \| \cdot \|$. Prove that for all $x, y \in \mathbb{R}^n$,

$$|x^T y| \leqq \|x\| \, \|y\|'$$

40. Prove this inequality for condition numbers as defined in Equation (13):

$$\kappa(AB) \leqq \kappa(A) \, \kappa(B)$$

41. Compute condition numbers using norms $\|A\|_1$, $\|A\|_2$, and $\|A\|_\infty$:
    (a) $\begin{bmatrix} a+1 & a \\ a & a-1 \end{bmatrix}$ (b) $\begin{bmatrix} 0 & 1 \\ -2 & 0 \end{bmatrix}$ (c) $\begin{bmatrix} \alpha & 1 \\ 1 & 1 \end{bmatrix}$

42. Using Problems 2 and 3, prove that

$$n^{-1} \|A\|_2 \leqq n^{-1/2} \|A\|_\infty \leqq \|A\|_2 \leqq n^{1/2} \|A\|_1 \leqq n \|A\|_2$$

43. In solving the system of equations $Ax = b$ with matrix $A = \begin{bmatrix} 1 & 2 \\ 1 & 2.01 \end{bmatrix}$, predict how slight changes in b will affect the solution x. Test your prediction in the concrete case when $b = (4, 4)$ and $b' = (3, 5)$.

44. Let A be an $m \times n$ matrix. We interpret A as a linear map from $\mathbb{R}^n$ with the norm $\| \cdot \|_1$ to $\mathbb{R}^m$ with the norm $\| \cdot \|_\infty$. What is $\|A\|$ under these circumstances? What is wanted is a simple formula for

$$\|A\| = \max \{ \|Ax\|_\infty : \|x\|_1 = 1 \}$$

45. Try Problem 44 when the norms $\| \cdot \|_1$ and $\| \cdot \|_\infty$ are interchanged.

46. Prove that the condition number $\kappa(A)$ can be expressed by the formula

$$\kappa(A) = \sup_{\|x\|=\|y\|} \|Ax\|/\|Ay\|$$

47. Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$, and let A be an $n \times n$ matrix. Put $\|x\|' = \|Ax\|$. What are the precise conditions on A to ensure that $\|\cdot\|'$ is also a norm?
48. Show that if the Euclidean norm is used, the set

$$H = \{x \in \mathbb{R}^n : \|x - a\| = \|x - b\|\}$$

will be a hyperplane (that is, a translate of a linear subspace of dimension $n-1$), but for other norms this is not generally true. Illustrate in the case $n = 2$.

49. Prove that the condition number has the property

$$\kappa(\lambda A) = \kappa(A) \qquad (\lambda \neq 0)$$

50. Prove that if a square matrix A satisfies an inequality $\|Ax\| \geqq \theta\|x\|$ for all x, with $\theta > 0$, then A is nonsingular, and $\|A^{-1}\| \leqq \theta^{-1}$. This is valid for any vector norm and its subordinate matrix norm.
51. Prove that if A is diagonally dominant, then it will have the property in Problem 50. Give a value for θ when the norm $\|\cdot\|_\infty$ is used.
52. Prove that if A is nonsingular, then there is a $\delta > 0$ with the property that $A + E$ is nonsingular for all matrices E satisfying $\|E\| < \delta$. This can be shown for any norm on the vector space of all $n \times n$ matrices.
53. Show that in Problem 52 we can use

$$\delta = \inf\{\|Ax\| : \|x\| = 1\}$$

Here a vector norm and its subordinate matrix norm are used.

54. Let A be an $n \times n$ matrix and N its null space (*kernel*). Define

$$\delta = \inf\{\|Ax\| : \|x\| = 1 \text{ and } x \perp N\}$$

Prove that if $\|E\| < \delta$, then $\operatorname{rank}(A + E) \geqq \operatorname{rank}(A)$.

55. Prove: If A is nonsingular, then there is a singular matrix within distance $\|A^{-1}\|^{-1}$ of A. Here the norm can be any subordinate matrix norm.
56. Establish Equation (14).

4.5 Neumann Series and Iterative Refinement

An important application of norms is in making precise the concept of convergence in a vector space. If a vector space V is assigned a norm $\|\cdot\|$, then we say that the pair $(V, \|\cdot\|)$ is a **normed linear space.** The notion of a convergent sequence of vectors $v^{(1)}, v^{(2)}, \ldots$ is then defined as follows. We say that the given sequence **converges** to a vector v if

$$\lim_{k\to\infty} \|v^{(k)} - v\| = 0$$

This conforms to our intuitive idea that the distances between the vectors $v^{(k)}$ and the limit vector v are approaching zero as k increases.

For an example in $\mathbb{R}^4$, let

$$v^{(k)} = \begin{bmatrix} 3 - k^{-1} \\ -2 + k^{-\frac{1}{2}} \\ (k+1)k^{-1} \\ e^{-k} \end{bmatrix} \quad \text{and} \quad v = \begin{bmatrix} 3 \\ -2 \\ 1 \\ 0 \end{bmatrix}$$

Then

$$v^{(k)} - v = \begin{bmatrix} -k^{-1} \\ k^{-\frac{1}{2}} \\ k^{-1} \\ e^{-k} \end{bmatrix}$$

If we compute $\|v^{(k)} - v\|$, using for example the infinity norm of Section 4.4, we see that $\|v^{(k)} - v\|_\infty \to 0$ as $k \to \infty$. Hence, v is the limit of the sequence $[v^{(k)}]$ in the normed linear space $(\mathbb{R}^4, \|\cdot\|_\infty)$.

At this point it is appropriate to quote without proof an important result about normed linear spaces: Any two norms on a finite-dimensional vector space lead to the same concept of convergence. Thus, in the preceding example, having verified that $\|v^{(k)} - v\|_\infty \to 0$, we can conclude without further calculation that $\|v^{(k)} - v\| \to 0$ for *any* norm on $\mathbb{R}^4$. *Caution:* This theorem does not apply in infinite-dimensional normed linear spaces. (See Problem 20 for an example.)

Another important result about finite-dimensional normed linear spaces is that in such a space, each Cauchy sequence converges. Thus, if a sequence $[v^{(k)}]$ in a finite-dimensional normed linear space satisfies the **Cauchy criterion**

$$\lim_{k \to \infty} \sup_{i,j \geq k} \|v^{(i)} - v^{(j)}\| = 0$$

then there necessarily exists a point $v \in V$ to which the sequence converges.

We shall apply these ideas to vectors in $\mathbb{R}^n$ and to $n \times n$ matrices. In the next theorem, we take $\|\cdot\|$ to be any norm on $\mathbb{R}^n$ and use its subordinate matrix norm $\|\cdot\|$ as defined in Section 4.4.

Neumann Series

THEOREM 1 *If A is an $n \times n$ matrix such that $\|A\| < 1$, then $I - A$ is invertible, and*

$$(I - A)^{-1} = \sum_{k=0}^{\infty} A^k \tag{1}$$

Proof First, we shall show that $I - A$ is invertible. If it is not invertible then it is singular, and there exists a vector x satisfying $\|x\| = 1$ and $(I - A)x = 0$. From this we have

$$1 = \|x\| = \|Ax\| \leqq \|A\| \|x\| = \|A\|$$

which contradicts the hypothesis that $\|A\| < 1$.

We shall show that the partial sums of the Neumann series converge to $(I - A)^{-1}$:

$$\sum_{k=0}^{m} A^k \to (I - A)^{-1} \qquad (\text{as } m \to \infty)$$

It will suffice to prove that

$$(I - A)\sum_{k=0}^{m} A^k \to I \qquad (\text{as } m \to \infty) \tag{2}$$

The left-hand side can be written as

$$(I - A)\sum_{k=0}^{m} A^k = \sum_{k=0}^{m}(A^k - A^{k+1}) = A^0 - A^{m+1} = I - A^{m+1}$$

Since $\| A^{m+1} \| \leqq \| A \|^{m+1} \to 0$, as $m \to \infty$, this establishes (2). (Why?) ∎

This theorem occurs in essentially the same form in the theory of continuous linear operators on any Banach space. The theorem has both practical and theoretical consequences of great importance. Observe that from Equation (1) we obtain this estimate:

$$\| (I - A)^{-1} \| \leqq \sum_{k=0}^{\infty} \|A^k\| \leqq \sum_{k=0}^{\infty} \|A\|^k = \frac{1}{1 - \| A \|}$$

Example 1 Employ the Neumann series to compute the inverse of the matrix

$$B = \begin{bmatrix} 0.9 & -0.2 & -0.3 \\ 0.1 & 1.0 & -0.1 \\ 0.3 & 0.2 & 1.1 \end{bmatrix}$$

Solution Let $B = I - A$, where

$$A = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ -0.1 & 0.0 & 0.1 \\ -0.3 & -0.2 & -0.1 \end{bmatrix}$$

Since $\|A\|_\infty = 0.6$, the Neumann series $\sum_{k=0}^{\infty} A^k$ will converge to B^{-1}. Using the algorithm in Problem 23, we compute some of the partial sums:

$$\sum_{k=0}^{0} A^k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \sum_{k=0}^{1} A^k = \begin{bmatrix} 1.1 & 0.2 & 0.3 \\ -0.1 & 1.0 & 0.1 \\ -0.3 & -0.2 & 0.9 \end{bmatrix}$$

$$\sum_{k=0}^{2} A^k = \begin{bmatrix} 1.00 & 0.16 & 0.32 \\ -0.14 & 0.96 & 0.06 \\ -0.28 & -0.24 & 0.80 \end{bmatrix}$$

$$\vdots$$

$$\sum_{k=0}^{19} A^k = \begin{bmatrix} 1.00000000 & 0.14285714 & 0.28571429 \\ -0.12500000 & 0.96428571 & 0.05357143 \\ -0.25000000 & -0.21428571 & 0.82142857 \end{bmatrix}$$

The last partial sum shown gives B^{-1} to 8-decimal places of accuracy. ∎

Here is a variant of Theorem 1:

THEOREM 2 *If A and B are $n \times n$ matrices such that $\| I - AB \| < 1$, then A and B are invertible. Furthermore,*

$$A^{-1} = B\sum_{k=0}^{\infty}(I - AB)^k \quad \textit{and} \quad B^{-1} = \sum_{k=0}^{\infty}(I - AB)^k A \tag{3}$$

Proof By the preceding theorem, AB is invertible and its inverse is

$$(AB)^{-1} = \sum_{k=0}^{\infty}(I - AB)^k$$

Hence

$$A^{-1} = BB^{-1}A^{-1} = B(AB)^{-1} = B\sum_{k=0}^{\infty}(I - AB)^k$$

$$B^{-1} = B^{-1}A^{-1}A = (AB)^{-1}A = \sum_{k=0}^{\infty}(I - AB)^k A$$

∎

Iterative Refinement

If $x^{(0)}$ is an approximate solution of the equation

$$Ax = b$$

then the precise solution x is given by

$$x = x^{(0)} + A^{-1}(b - Ax^{(0)}) = x^{(0)} + e^{(0)} \tag{4}$$

where $e^{(0)} = A^{-1}(b - Ax^{(0)})$ and is called the **error vector**. The **residual vector** corresponding to the approximate solution $x^{(0)}$ is $r^{(0)} = b - Ax^{(0)}$. It is computable. Of course, we do not want to compute A^{-1}, but the vector $e^{(0)} = A^{-1}r^{(0)}$ can be obtained by solving the equation

$$Ae^{(0)} = r^{(0)}$$

These remarks lead to a numerical procedure called **iterative improvement** or **iterative refinement**, which we now describe in more detail.

Suppose that the equation $Ax = b$ has been solved by Gaussian elimination as in Section 4.3. Since the result is not expected to be the exact solution (because of roundoff errors), we denote it by $x^{(0)}$ and then compute $r^{(0)}, e^{(0)}$, and $x^{(1)}$ by the three equations

$$\begin{cases} r^{(0)} = b - Ax^{(0)} \\ Ae^{(0)} = r^{(0)} \\ x^{(1)} = x^{(0)} + e^{(0)} \end{cases} \tag{5}$$

To obtain better solutions $x^{(2)}, x^{(3)}, \ldots$, this process can be repeated. The success of the method depends on computing the residuals $r^{(i)}$ in *double precision* to avoid the loss of significance expected in the subtraction. Thus, the expression $b_i - \sum_{j=1}^{n} a_{ij} x_j^{(0)}$ is evaluated in double precision. (Remember that, ideally, $r^{(i)}$ should be the zero vector, and thus the subtraction involved in computing $r^{(i)}$ *must* involve nearly equal quantities.)

Example 2 Apply iterative refinement in solving the system

$$\begin{bmatrix} 420 & 210 & 140 & 105 \\ 210 & 140 & 105 & 84 \\ 140 & 105 & 84 & 70 \\ 105 & 84 & 70 & 60 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 875 \\ 539 \\ 399 \\ 319 \end{bmatrix}$$

Solution First, the system is factored by Gaussian elimination with scaled row pivoting, and the solution

$$x^{(0)} = (0.999988,\ 1.000137,\ 0.999670,\ 1.000215)^T$$

is obtained from back substitution. Several steps of iterative improvement give

$$x^{(1)} = (0.999994,\ 1.000069,\ 0.999831,\ 1.000110)^T$$
$$x^{(2)} = (0.999996,\ 1.000046,\ 0.999891,\ 1.000070)^T$$
$$x^{(3)} = (0.999993,\ 1.000080,\ 0.999812,\ 1.000121)^T$$
$$x^{(4)} = (1.000000,\ 1.000006,\ 0.999984,\ 1.000011)^T$$

The true solution is $x = (1,\ 1,\ 1,\ 1)^T$. Since a computer having approximately the precision of the MARC-32 was used for these calculations, we consider the final result as quite good. ∎

To analyze this algorithm theoretically, we adopt the point of view that our solution $x^{(0)}$ is obtained by the formula

$$x^{(0)} = Bb$$

where B is an approximate inverse of A. The iterative process then can be written

$$x^{(k+1)} = x^{(k)} + B(b - Ax^{(k)}) \qquad (k \geqq 0) \tag{6}$$

We shall now show that these vectors are the partial sums of the Neumann series and use this fact to show that the sequence $[x^{(k)}]$ converges to the solution of $Ax = b$.

We interpret the loose expression, "B is an approximate inverse of A", to mean that $\|I - AB\| < 1$. By Theorem 2, A^{-1} is given by

$$A^{-1} = B \sum_{k=0}^{\infty} (I - AB)^k \tag{7}$$

Thus, the exact solution of the equation $Ax = b$ is

$$x = B\sum_{k=0}^{\infty}(I - AB)^k b \tag{8}$$

THEOREM 3 *If* $\| I - AB \| < 1$, *then the method of iterative improvement given by Equation (6) produces the sequence of vectors*

$$x^{(m)} = B\sum_{k=0}^{m}(I - AB)^k b \qquad (m \geqq 0)$$

These are the partial sums in Equation (8) and therefore converge to x.

Proof We use induction. Since $x^{(0)} = Bb$, the case $m = 0$ is true. If the mth case is assumed true, then the $(m+1)$st case is true since

$$x^{(m+1)} = x^{(m)} + B(b - Ax^{(m)}) = B\sum_{k=0}^{m}(I - AB)^k b + Bb - BAB\sum_{k=0}^{m}(I - AB)^k b$$

$$= B\left\{b + (I - AB)\sum_{k=0}^{m}(I - AB)^k b\right\} = B\sum_{k=0}^{m+1}(I - AB)^k b \qquad \blacksquare$$

It is also possible to prove directly that the vectors $x^{(m)}$ converge to x. To do this, use Equation (6) to obtain

$$\begin{aligned} x^{(m+1)} - x &= x^{(m)} - x + B(Ax - Ax^{(m)}) \\ &= (I - BA)(x^{(m)} - x) \end{aligned}$$

It follows that

$$\begin{aligned} \|x^{(m+1)} - x\| &\leqq \|I - BA\|\ \|x^{(m)} - x\| \\ &\leqq \|I - BA\|^2\ \|x^{(m-1)} - x\| \\ &\leqq \quad \vdots \\ &\leqq \|I - BA\|^m\ \|x^{(0)} - x\| \end{aligned}$$

Since we have assumed that $\|I - BA\| < 1$, the errors $\|x^{(m)} - x\|$ converge to 0 as $m \to \infty$ for any $x^{(0)}$.

Equilibration

For solving systems of linear equations in extremely critical cases, a number of refinements can be added to the factorization and solution procedures described in Section 4.3. We shall discuss briefly five such techniques:

(i) preconditioning by row equilibration.

(ii) preconditioning by column equilibration.

(iii) full pivoting.

(iv) preconditioning or scaling within each major step of the elimination procedure.

(v) iterative improvement at the end.

Row equilibration is the process of dividing each row of the coefficient matrix by the maximum element in absolute value in that row; that is, multiplying row i by $r_i = 1/\max_{1\leqq j\leqq n}|a_{i,j}|$ for $1 \leqq i \leqq n$. After this has been done, the new elements, $\tilde{a}_{ij}$, will satisfy $\max_{1\leqq j\leqq n}|\tilde{a}_{ij}| = 1$ for $1 \leqq i \leqq n$. In numerical practice on a binary computer, the factor r_i is taken to be a number of the form 2^m as close as possible to $1/\max_{1\leqq j\leqq n}|a_{ij}|$. This is done to avoid the introduction of additional roundoff errors. Since the ith equation in our system is

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad \text{or} \quad \sum_{j=1}^{n}(r_i a_{ij})x_j = r_i b_i$$

we should multiply b_i by r_i also. Hence, the numbers r_i should be stored during the factorization procedure so that they can be used in the solution procedure. In matrix-vector notation, row equilibration is written as

$$(RA)x = (Rb)$$

where $R = \text{diag}\,(r_i)$.

Column equilibration is similar except that we are dealing with the columns: we multiply column j by $c_j = 1/\max_{1\leqq i\leqq n}|a_{ij}|$ for $1 \leqq j \leqq n$. Again, it is preferable to take c_j to be a number of the form 2^m as close as possible to $1/\max_{1\leqq i\leqq n}|a_{ij}|$. Our original equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad (1 \leqq i \leqq n)$$

now can be written as

$$\sum_{j=1}^{n}(c_j a_{ij})\left(\frac{x_j}{c_j}\right) = b_i \qquad (1 \leqq i \leqq n)$$

After the solution phase, we shall have computed approximate solutions for the components x_j/c_j. These solutions must be multiplied by c_j to obtain x_j. Thus, $c_1, c_2, \ldots, c_n$ will have to be stored also. Using matrices, column equilibration is written as

$$(AC)(C^{-1}x) = b$$

where $C = \text{diag}\,(c_j)$.

If row and column equilibrations have been carried out on our system, the full pivoting strategy at the initial step can be safely simplified to a search for the largest element (in magnitude) in the matrix. This element determines both the first pivot row *and* the first column in which zeros will be introduced by the elimination. Thus, we intend to process the columns not in the natural order $1, 2, 3, \ldots, n$ but in an order determined by this more accurate pivoting strategy. Two permutation arrays will be needed, one to list the row numbers of the successive pivot elements and another to list the corresponding column numbers. (Refer to Problems 4 and 6 in Section 4.3.)

The fourth technique in our list of refinements is preconditioning or scaling. It contributes to a more logical organization of the factorization phase, for each step in the algorithm will be just like the first except for being applied to smaller matrices.

The fifth technique in our list is iterative improvement, which has been discussed previously.

The value of row and column equilibration as a *preconditioning* procedure is somewhat controversial. We can see that row equilibration followed by Gaussian elimination with *unscaled* row pivoting is virtually the same as Gaussian elimination with *scaled* row pivoting. Hence, one reasonable strategy is to begin with column equilibration followed by scaled row pivoting in Gaussian elimination.

This example shows the difference between row-column equilibration

$$\begin{bmatrix} 1 & 10^8 \\ 2 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 10^{-8} & 1 \\ 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 10^{-8} & 1 \\ 1 & 0 \end{bmatrix}$$

and column-row equilibration

$$\begin{bmatrix} 1 & 10^8 \\ 2 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{1}{2} & 1 \\ 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{1}{2} & 1 \\ 1 & 0 \end{bmatrix}$$

Note that row-column equilibration results in a pre- and post-scaling of the system by diagonal matrices, say R and B, such as

$$((RA)B)(B^{-1}x) = (Rb)$$

whereas column-row equilibration corresponds to

$$(S(AC))(C^{-1}x) = (Sb)$$

for some diagonal matrices C and S.

PROBLEM SET 4.5

1. Prove that the set of invertible $n \times n$ matrices is an open set in the set of all $n \times n$ matrices. Thus, if A is invertible, then there is a positive ε such that every matrix B satisfying $\| A - B \| < \varepsilon$ is also invertible.
2. Prove that if A is invertible and $\| B - A \| < \| A^{-1} \|^{-1}$, then B is invertible.
3. Prove that if $\| A \| < 1$, then

$$\| (I - A)^{-1} \| \geqq \frac{1}{1 + \| A \|}$$

4. Prove that if A is invertible and $\| A - B \| < \| A^{-1} \|^{-1}$, then

$$\| A^{-1} - B^{-1} \| \leqq \| A^{-1} \| \frac{\| I - A^{-1}B \|}{1 - \| I - A^{-1}B \|}$$

5. Prove that if $\| AB - I \| < 1$, then (with E denoting $AB - I$) we have

$$A^{-1} = B - BE + BE^2 - BE^3 + \cdots$$

6. Prove that if A is invertible, then for any B,

$$\| B - A^{-1} \| \geqq \frac{\| I - AB \|}{\| A \|}$$

7. Prove or disprove: If $1 = \| A \| > \| B \|$, then $A - B$ is invertible.
8. Prove that if $\| A \| < 1$, then

$$(I + A)^{-1} = I - A + A^2 - A^3 + \cdots$$

9. Prove or disprove: If $\| AB - I \| < 1$, then $\| BA - I \| < 1$.
10. Prove that if $\|A\| < 1$, then $\|(I + A)^{-1}\| \leqq (1 - \|A\|)^{-1}$.
11. Prove that if A is invertible and $\| B - A \| < \| A^{-1} \|^{-1}$, then

$$B^{-1} = A^{-1} \sum_{k=0}^{\infty} (I - BA^{-1})^k$$

12. For any $n \times n$ matrix A, prove that

$$A^m = I - (I - A) \sum_{k=0}^{m-1} A^k$$

13. Criticize this "proof" that every $n \times n$ matrix is invertible: Given A, select a vector norm so that $\| I - A \| < 1$ when the associated matrix norm is used. Then apply the theorem concerning the Neumann series.
14. Prove that if $\inf_{\lambda \in \mathbb{R}} \| I - \lambda A \| < 1$, then A is invertible.
15. Prove that the invertible $n \times n$ matrices form a dense set in the set of all $n \times n$ matrices. This means that if A is an $n \times n$ matrix and if $\varepsilon > 0$, then there is an invertible matrix B such that $\| A - B \| < \varepsilon$. (See Problem 1.)
16. Show that if $\| AB - I \| = \varepsilon < 1$, then

$$\| A^{-1} - B \| < \| B \| \left(\frac{\varepsilon}{1 - \varepsilon} \right)$$

17. Prove that the operation $x \mapsto Ax$ is continuous. That is, for fixed A, show that if a sequence $[x^{(k)}]$ converges to x then $[Ax^{(k)}]$ converges to Ax.
18. Prove that if E is an $n \times n$ matrix for which $\| E \|$ is sufficiently small, then

$$\| (I - E)^{-1} - (I + E) \| \leqq 3 \| E \|^2$$

How small must $\| E \|$ be?

19. Prove that if A is invertible, then

$$\| Ax \| \geqq \|x\| \, \| A^{-1} \|^{-1}$$

20. Consider the vector space V of all continuous functions defined on the interval $[0, 1]$. Two important norms on V are

$$\|x\|_\infty = \max_{0 \leqq t \leqq 1} |x(t)| \qquad \|x\|_1 = \int_0^1 |x(t)| dt$$

Show that the sequence of functions $x_n(t) = t^n$ has the properties $\|x_n\|_\infty = 1$ and $\|x_n\|_1 \to 0$ as $n \to \infty$. Thus these norms lead to different concepts of convergence.

21. Prove that if $\| AB - I \| < 1$, then $2B - BAB$ is a better approximate inverse for A than B, in the sense that $A(2B - BAB)$ is closer to I.

[c]22. The purpose of this project is to write and test procedures that are refinements of the algorithms given in the text. The refinements to be incorporated are these:

(i) Column equilibration.

(ii) Row equilibration.

(iii) Full pivoting.

(iv) Include two steps of iterative refinement at the end. Alternatively, write a separate subprogram to take A, x, b and improve x twice (or m times). Here, each residual $b_i - \sum_{j=1}^{n} a_{ij} x_j$ should be computed in double precision and then rounded to single precision. Notice that the *original* matrix A and vector b must be saved in order to compute the residuals. Test your codes by solving $Ax = b$, where $n = 10$ and

$$\begin{cases} a_{ij} = (i/11)^j & (1 \leqq i, j \leqq n) \\ b_i = i[1 - (i/11)^{10}]/(33 - 3i) & (1 \leqq i \leqq n) \end{cases}$$

As a second test of your codes, let $n = 4, a_{ij} = 1/(2n - i - j + 1)$, and $b_i = \sum_{j=1}^{n} a_{ij}$. The solution should be $x = (1, 1, 1, 1)^T$. For the third test, repeat the previous test with $n = 10$. Include enough print statements to see lots of detail, in particular, the initial solution and the solutions obtained with iterative improvement.

23. Let $B_k = \sum_{j=0}^{k} A^j$. Show that the sequence $[B_k]$ can be computed recursively by the formulæ $B_0 = I$ and $B_{k+1} = I + AB_k$.

[c]24. Using a test matrix A of dimensions 3×3, and the method of the preceding problem, compute $B = \sum_{j=0}^{20} A^j$ and see whether $(I - A)B \approx I$. Be sure that $\| A \| < 1$ for some subordinate matrix norm.

25. Give a series that represents A^{-1} under the assumption that $\| I - \alpha A \| < 1$ for some known scalar α.

26. In a normed linear space, prove that if a sequence of vectors converges, then it must satisfy the Cauchy criterion.

27. Prove that if A is ill conditioned, then there is a singular matrix near A. In fact, there is a singular matrix within distance $\|A\|/\kappa(A)$ of A.

28. Consider the linear system $\begin{bmatrix} 1 & 2 \\ 1+\delta & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3+\delta \end{bmatrix}$, for small $\delta > 0$.

    (a) Using the approximate solution $\widetilde{x} = (3, 0)^T$, compare the infinity norm of the residual vector and the infinity norm of the error vector. What conclusion can you draw?

    (b) Determine the condition number $\kappa_\infty(A)$. What happens as δ gets small?

    (c) Carry out one step of iterative improvement based on the approximate solution $\widetilde{x}$.

29. Prove that if $\|I - AB\| < 1$, then BA is invertible. (Here A and B are square matrices. What happens if they are not square?)

30. Prove that if $\|I - cA^n\| < 1$ for some c and some integer $n \geqq 1$, then A is invertible.

31. Prove that if there is a polynomial p without constant term such that

$$\|I - p(A)\| < 1$$

    then A is invertible.

32. Prove that if p is a polynomial with constant term c_0 and if $|c_0| + \|I - p(A)\| < 1$, then A is invertible.

°33. Solve the following linear system and apply three sweeps of iterative improvement. Print r, e, and x after iteration

$$\begin{bmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 110 \\ 65 \\ 47 \end{bmatrix}$$

*4.6 Solution of Equations by Iterative Methods

The Gaussian algorithm and its variants are termed **direct** methods for the matrix problem $Ax = b$. They proceed through a finite number of steps and produce a solution x that would be completely accurate were it not for roundoff errors.

An **indirect** method, by contrast, produces a sequence of vectors that ideally *converges* to the solution. The computation is halted when an approximate solution is obtained having some specified accuracy or after a certain number of iterations. Indirect methods are almost always **iterative** in nature: a simple process is applied repeatedly to generate the sequence referred to previously.

For large linear systems containing thousands of equations, iterative methods often have decisive advantages over direct methods in terms of speed and demands on computer memory. Sometimes, if the accuracy requirements are not stringent, a modest number of iterations will suffice to produce an acceptable solution. For **sparse** systems (in which a large proportion of the elements in A are zero), iterative methods are often very efficient. In sparse problems, the nonzero elements of A are sometimes stored in a sparse-storage format, while in other cases it is not necessary to store A at all! The latter situation is common in the numerical solution of partial differential equations. In this case, each row of A might be generated as needed but not retained after use. Another advantage of iterative methods is that they are usually stable, and

they will actually dampen errors (due to roundoff or minor blunders) as the process continues.

To convey the general idea, we describe two fundamental iterative methods.

Example 1 Consider the linear system

$$\begin{bmatrix} 7 & -6 \\ -8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \end{bmatrix}$$

How can it be solved by an iterative process?

Solution A straightforward procedure would be to solve the ith equation for the ith unknown as follows:

$$x_1^{(k)} = \tfrac{6}{7} x_2^{(k-1)} + \tfrac{3}{7}$$
$$x_2^{(k)} = \tfrac{8}{9} x_1^{(k-1)} - \tfrac{4}{9}$$

This is known as the **Jacobi** method or iteration. Initially, we select for $x_1^{(0)}$ and $x_2^{(0)}$ the best available guess for the solution, or simply set them to be zero. The equations above then generate what we hope are *improved* values, $x_1^{(1)}$ and $x_2^{(1)}$. The process is then repeated a prescribed number of times or until a certain precision appears to have been achieved in the vector $(x_1^{(k)}, x_2^{(k)})^T$. Here are some selected values of the iterates of the Jacobi method for this example:

k	$x_1^{(k)}$	$x_2^{(k)}$
0	0.000000	0.000000
10	0.148651	−0.198201
20	0.186816	−0.249088
30	0.196615	−0.262154
40	0.199131	−0.265508
50	0.199777	−0.266369

It is apparent that this iterative process could be modified so the *newest* value for $x_1^{(k)}$ is used immediately in the second equation. The resulting method is called the **Gauss-Seidel** method or iteration. Its equations are

$$x_1^{(k)} = \tfrac{6}{7} x_2^{(k-1)} + \tfrac{3}{7}$$
$$x_2^{(k)} = \tfrac{8}{9} x_1^{(k)} - \tfrac{4}{9}$$

Some of the output from the Gauss-Seidel method follows:

k	$x_1^{(k)}$	$x_2^{(k)}$
0	0.000000	0.000000
10	0.219776	−0.249088
20	0.201304	−0.265308
30	0.200086	−0.266590
40	0.200006	−0.266662
50	0.200000	−0.266666

Both the Jacobi and the Gauss-Seidel iterates seem to be converging to the same limit, and the latter is converging faster. Also, notice that, in contrast to a direct method, the precision we obtain in the solution depends on when the iterative process is halted. ■

Basic Concepts

We now consider iterative methods in a more general mathematical setting. A general type of iterative process for solving the system

$$Ax = b \tag{1}$$

can be described as follows. A certain matrix Q—called the **splitting** matrix—is prescribed, and the original problem is rewritten in the equivalent form

$$Qx = (Q - A)x + b \tag{2}$$

Equation (2) suggests an iterative process, defined by writing

$$Qx^{(k)} = (Q - A)x^{(k-1)} + b \qquad (k \geqq 1) \tag{3}$$

The initial vector $x^{(0)}$ can be arbitrary; if a good guess of the solution is available, it should be used for $x^{(0)}$. We shall say that the iterative method in Equation (3) converges if it converges for *any* initial vector $x^{(0)}$. A sequence of vectors $x^{(1)}, x^{(2)}, \ldots$ can be computed from Equation (3), and our objective is to choose Q so that

(i) the sequence $[x^{(k)}]$ is easily computed, and

(ii) the sequence $[x^{(k)}]$ converges rapidly to a solution.

In this section, we shall see that both of these conditions follow if it is easy to solve $Qx^{(k)} = y$ and if Q^{-1} approximates A^{-1}.

Observe, to begin with, that if the sequence $[x^{(k)}]$ converges, say to a vector x, then x is *automatically* a solution. Indeed, if we simply take the limit in Equation (3) and use continuity of the algebraic operations, the result is

$$Qx = (Q - A)x + b \tag{4}$$

which means that $Ax = b$.

To assure that System (1) have a solution for any vector b, we shall assume that A is nonsingular. We shall assume that Q is nonsingular as well, so that Equation (3) can be solved for the unknown vector $x^{(k)}$. Having made these assumptions, we can use the following equation for the *theoretical* analysis:

$$x^{(k)} = (I - Q^{-1}A)x^{(k-1)} + Q^{-1}b \tag{5}$$

It is to be emphasized that Equation (5) is convenient for the analysis, but in numerical work $x^{(k)}$ is almost always obtained by solving Equation (3) without the use of Q^{-1}.

Observe that the actual solution x satisfies the equation

$$x = (I - Q^{-1}A)x + Q^{-1}b \tag{6}$$

Thus, x is a fixed point of the mapping

$$x \longmapsto (I - Q^{-1}A)x + Q^{-1}b \tag{7}$$

By subtracting the terms in Equation (6) from those in Equation (5), we obtain

$$x^{(k)} - x = (I - Q^{-1}A)(x^{(k-1)} - x) \tag{8}$$

Now select any convenient vector norm and its subordinate matrix norm. We obtain from Equation (8)

$$\|x^{(k)} - x\| \leqq \| I - Q^{-1}A \| \, \|x^{(k-1)} - x\| \tag{9}$$

By repeating this step, we arrive eventually at the inequality

$$\|x^{(k)} - x\| \leqq \| I - Q^{-1}A \|^k \, \|x^{(0)} - x\| \tag{10}$$

Thus, if $\| I - Q^{-1}A \| < 1$, we can conclude at once that

$$\lim_{k \to \infty} \|x^{(k)} - x\| = 0 \tag{11}$$

for any $x^{(0)}$. Observe that the hypothesis $\|I - Q^{-1}A\| < 1$ implies (by Theorem 1 in Section 4.5) the invertibility of $Q^{-1}A$ and of A. Hence, we have

THEOREM 1 *If $\| I - Q^{-1}A \| < 1$ for some subordinate matrix norm, then the sequence produced by Equation (3) converges to the solution of $Ax = b$ for any initial vector $x^{(0)}$.*

If the norm $\delta \equiv \|I - Q^{-1}A\|$ is less than 1, then it is safe to halt the iterative process when $\|x^{(k)} - x^{(k-1)}\|$ is small. Indeed, we can prove (Problem 35) that

$$\|x^{(k)} - x\| \leqq \frac{\delta}{1 - \delta} \|x^{(k)} - x^{(k-1)}\|$$

Richardson Method

As an illustration of these concepts, we consider the **Richardson method**, in which Q is chosen to be the identity matrix. Equation (3) in this case reads as follows:

$$x^{(k)} = (I - A)x^{(k-1)} + b = x^{(k-1)} + r^{(k-1)} \tag{12}$$

where $r^{(k-1)}$ is the residual vector, defined by $r^{(k-1)} = b - Ax^{(k-1)}$. According to Theorem 1, the Richardson iteration will produce a solution to $Ax = b$ (in the limit) if $\| I - A \| < 1$ for some subordinate matrix norm. (See Problems 2 and 3 for two classes of matrices having the required property.)

An algorithm to carry out the Richardson iteration is as follows:

input $n, (a_{ij}), (b_i), (x_i), M$
for $k = 1, 2, \ldots, M$ **do**
  for $i = 1, 2, \ldots, n$ **do**
    $r_i \leftarrow b_i - \sum_{j=1}^{n} a_{ij} x_j$
  end
  for $i = 1, 2, \ldots, n$ **do**
    $x_i \leftarrow x_i + r_i$
  end
end
output $k, (x_i), (r_i)$

Example 2 Compute 100 iterates on the following problem, using the Richardson method starting with $x = (0, 0, 0)^T$:

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{3} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{11}{18} \\ \frac{11}{18} \\ \frac{11}{18} \end{bmatrix}$$

Solution A computer program based on the above algorithm was written. A few of the iterates produced by this routine are shown here:

$$\begin{aligned} x^{(0)} &= (\ 0.0,\ \ 0.0,\ \ 0.0\ )^T \\ x^{(1)} &= (\ 0.611,\ \ 0.611,\ \ 0.611\ )^T \\ &\vdots \\ x^{(10)} &= (\ 0.279,\ \ 0.279,\ \ 0.279\ )^T \\ &\vdots \\ x^{(40)} &= (\ 0.333,\ \ 0.333,\ \ 0.333\ )^T \\ &\vdots \\ x^{(80)} &= (\ 0.333333,\ \ 0.333333,\ \ 0.333333\ )^T \end{aligned}$$

■

Jacobi Method

Another illustration of our basic theory is provided by the **Jacobi iteration**, in which Q is the diagonal matrix whose diagonal entries are the same as those in the matrix $A = (a_{ij})$. In this case, the generic element of $Q^{-1}A$ is a_{ij}/a_{ii}. The diagonal elements of this matrix are all 1, and hence

$$\| I - Q^{-1}A \|_\infty = \max_{1 \leqq i \leqq n} \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}/a_{ii}| \tag{13}$$

THEOREM 2 *If A is diagonally dominant, then the sequence produced by the Jacobi iteration converges to the solution of $Ax = b$ for any starting vector.*

Proof Diagonal dominance means that

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad (1 \leqq i \leqq n)$$

From Equation (13), we then conclude that

$$\| I - Q^{-1}A \|_\infty < 1$$

By Theorem 1, the Jacobi iteration converges. ∎

An algorithm for the Jacobi method follows.

input $n, (a_{ij}), (b_i), (x_i), M$
for $k = 1, 2, \ldots, M$ **do**
  for $i = 1, 2, \ldots, n$ **do**
    $u_i \leftarrow \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j\right) \Big/ a_{ii}$
  end
  for $i = 1, 2, \ldots n$ **do**
    $x_i \leftarrow u_i$
  end
  output $k, (x_i)$
end

This algorithm and others in this section can be made more efficient by performing all divisions before the iteration begins. Thus, we could start the computation with these operations:

for $i = 1, 2, \ldots, n$ **do**
  $d = 1/a_{ii}$
  $b_i \leftarrow db_i$
  for $j = 1, 2, \ldots, n$ **do**
    $a_{ij} = da_{ij}$
  end
end

Then the replacement statement for u_i becomes simply

$$u_i \leftarrow b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j$$

Another way to interpret this is that the original system $Ax = b$ has been replaced by $D^{-1}Ax = D^{-1}b$, where $D = \text{diag}(a_{ii})$. Alternatively, divisions can be avoided by preprocessing the system with some other scaling such as the two-sided scaling $(D^{-1/2}AD^{-1/2})(D^{1/2}x) = (D^{-1/2}b)$, where $D^{1/2} = \text{diag}\,(\sqrt{a_{ii}})$, provided A has positive diagonal entries. Notice that if A is symmetric, then this scaling preserves symmetry. In many iterative methods, some simple preparatory work such as this can pay large dividends in efficiency or speed of convergence.

Analysis

Our next task is to develop some of the theory for *arbitrary* linear iterative processes. We consider that such a process has been defined by an equation of the form

$$x^{(k)} = Gx^{(k-1)} + c \tag{14}$$

in which G is a prescribed $n \times n$ matrix and c is a prescribed vector in $\mathbb{R}^n$. Notice that the iteration defined in Equation (3) will be included in any general theory that we may develop for Equation (14); namely, we can set $G = I - Q^{-1}A$ and $c = Q^{-1}b$. We want to find a necessary and sufficient condition on G so that the iteration of Equation (14) will converge for any starting vector. Some preliminary matters must be dealt with first.

The **eigenvalues** of a matrix A are the complex numbers λ for which the matrix $A - \lambda I$ is not invertible. These numbers are then the roots of the **characteristic equation** of A:

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

(The reader may wish to look ahead at Section 5.1, where these concepts are discussed further.) The **spectral radius** of A is defined by the equation

$$\rho(A) = \max\{|\lambda| : \det(A - \lambda I) = 0\}$$

Thus, $\rho(A)$ is the smallest number such that a circle with that radius centered at 0 in the complex plane will contain all the eigenvalues of A.

A matrix A is said to be **similar** to a matrix B if there is a nonsingular matrix S such that $S^{-1}AS = B$. It follows that similar matrices have the same eigenvalues. Moreover, it is easy to see that the eigenvalues of a triangular matrix are the elements on its diagonal.

THEOREM 3 *Every square matrix is similar to a (possibly complex) upper triangular matrix whose off-diagonal elements are arbitrarily small.*

Proof Let A be an $n \times n$ matrix. We borrow a result know as Schur's Theorem from Section 5.1. This theorem states that A is similar to an upper triangular matrix $T = (t_{ij})$, which may be complex. Now let $0 < \varepsilon < 1$, and let $D = \text{diag}(\varepsilon, \varepsilon^2, \ldots, \varepsilon^n)$. The generic element of $D^{-1}TD$ is $t_{ij}\varepsilon^{j-i}$, by an elementary calculation. Since T is upper triangular, the elements below the diagonal in $D^{-1}TD$ are 0, while the elements above the diagonal satisfy

$$|t_{ij}\varepsilon^{j-i}| \leq \varepsilon |t_{ij}|$$

because $j > i$ and $\varepsilon < 1$. This upper bound can be made as small as we wish by decreasing ε. ∎

THEOREM 4 *The spectral radius function satisfies the equation*

$$\rho(A) = \inf_{\|\cdot\|} \| A \|$$

in which the infimum is taken over all subordinate matrix norms.

Proof It is easy to prove that $\rho(A) \leq \inf_{\|\cdot\|} \| A \|$. To do so, let λ be any eigenvalue of A. Select a nonzero eigenvector x corresponding to λ. Then for any vector norm and its subordinate matrix norm, we have

$$|\lambda| \, \|x\| = \|\lambda x\| = \|Ax\| \leq \| A \| \, \|x\|$$

Hence, $|\lambda| \leq \| A \|$. It follows that $\rho(A) \leq \| A \|$. By taking an infimum, we have $\rho(A) \leq \inf_{\|\cdot\|} \| A \|$.

For the reverse inequality, we use the preceding theorem. It asserts that for any $\varepsilon > 0$ there exists a nonsingular matrix S such that $S^{-1}AS = D + T$, where D is diagonal and T is strictly upper triangular, with $\| T \|_\infty \leq \varepsilon$. Then we have

$$\| S^{-1}AS \|_\infty = \| D + T \|_\infty \leq \| D \|_\infty + \| T \|_\infty$$

Since D has the eigenvalues of A on its diagonal, it follows that

$$\| D \|_\infty = \max_{1 \leq i \leq n} |\lambda_i| = \rho(A)$$

Hence, we have

$$\| S^{-1}AS \|_\infty \leq \rho(A) + \varepsilon$$

By appealing to Problem 6, we know that the function $\| \cdot \|'_\infty$ defined by

$$\|A\|'_\infty \equiv \| S^{-1}AS \|_\infty$$

is a subordinate matrix norm. Now $\|A\|'_\infty \leq \rho(A) + \varepsilon$. An infimum over all subordinate matrix norms gives us

$$\inf_{\|\cdot\|} \| A \| \leq \rho(A) + \varepsilon$$

Since ε was arbitrary, $\inf_{\|\cdot\|} \| A \| \leq \rho(A)$. ∎

Theorem 4 tells us that for any matrix A its spectral radius lies below its norm value for any subordinate matrix norm and, moreover, there exists a subordinate matrix norm with a value arbitrarily close to the spectral radius.

We now give a necessary and sufficient condition on the iteration matrix G for convergence of the associated iterative method.

THEOREM 5 *For the iteration formula*

$$x^{(k)} = Gx^{(k-1)} + c$$

to produce a sequence converging to $(I - G)^{-1}c$, *for any starting vector* $x^{(0)}$, *it is necessary and sufficient that the spectral radius of* G *be less than* 1.

Proof Suppose that $\rho(G) < 1$. By Theorem 4, there is a subordinate matrix norm such that $\| G \| < 1$. We write

$$\begin{aligned} x^{(1)} &= Gx^{(0)} + c \\ x^{(2)} &= G^2x^{(0)} + Gc + c \\ x^{(3)} &= G^3x^{(0)} + G^2c + Gc + c \end{aligned}$$

The general formula is

$$x^{(k)} = G^k x^{(0)} + \sum_{j=0}^{k-1} G^j c \tag{15}$$

Using the vector norm that engendered our matrix norm, we have

$$\|G^k x^{(0)}\| \leqq \|G^k\| \|x^{(0)}\| \leqq \|G\|^k \|x^{(0)}\| \to 0 \quad \text{as} \quad k \to \infty$$

By the theorem on the Neumann series (Theorem 1 in Section 4.5), we have

$$\sum_{j=0}^{\infty} G^j c = (I - G)^{-1} c$$

Thus by letting $k \to \infty$ in Equation (15), we obtain

$$\lim_{k \to \infty} x^{(k)} = (I - G)^{-1} c$$

For the converse, suppose that $\rho(G) \geqq 1$. Select u and λ so that

$$Gu = \lambda u \qquad |\lambda| \geqq 1 \qquad u \neq 0$$

If $|\lambda| = 1$, let $c = u$ and $x^{(0)} = 0$. By Equation (15), $x^{(k)} = \sum_{j=0}^{k-1} G^j u = \pm k u$, which diverges as $k \to \infty$. If $\lambda > 1$, let $c = 0$ and $x^{(0)} = u$. Similarly by Equation (15), $x^{(k)} = G^k u = \lambda^k u$, which diverges as $k \to \infty$. ∎

COROLLARY *The iteration formula* (3) *— that is,* $Qx^{(k)} = (Q - A)x^{(k-1)} + b$ *— will produce a sequence converging to the solution of* $Ax = b$*, for any* $x^{(0)}$*, if* $\rho(I - Q^{-1}A) < 1$.

Gauss-Seidel Method

Let us examine the **Gauss-Seidel iteration** in more detail. It is defined by letting Q be the lower triangular part of A, including the diagonal.

THEOREM 6 *If* A *is diagonally dominant, then the Gauss-Seidel method converges for any starting vector.*

Proof By the corollary, it suffices to prove that

$$\rho(I - Q^{-1}A) < 1$$

To this end, let λ be any eigenvalue of $I - Q^{-1}A$. Let x be a corresponding eigenvector. We assume, with no loss of generality, that $\|x\|_\infty = 1$. We have now

$$(I - Q^{-1}A)x = \lambda x \qquad \text{or} \qquad Qx - Ax = \lambda Qx$$

Since Q is the lower triangular part of A, including its diagonal,

$$-\sum_{j=i+1}^{n} a_{ij} x_j = \lambda \sum_{j=1}^{i} a_{ij} x_j \qquad (1 \leqq i \leqq n)$$

By transposing terms in this equation, we obtain

$$\lambda a_{ii} x_i = -\lambda \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x_j \qquad (1 \leqq i \leqq n)$$

Select an index i such that $|x_i| = 1 \geqq |x_j|$ for all j. Then

$$|\lambda|\ |a_{ii}| \leqq |\lambda| \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}|$$

Solving for $|\lambda|$ and using the diagonal dominance of A, we get

$$|\lambda| \leqq \left\{ \sum_{j=i+1}^{n} |a_{ij}| \right\} \left\{ |a_{ii}| - \sum_{j=1}^{i-1} |a_{ij}| \right\}^{-1} < 1$$ ∎

An algorithm for the Gauss-Seidel iteration follows:

input $n, (a_{ij}), (b_i), (x_i), M$
for $k = 1, 2, \ldots, M$ **do**
  for $i = 1, 2, \ldots, n$ **do**

$$x_i \leftarrow \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j \right) \Big/ a_{ii}$$

  end
output $k, (x_i)$
end

Notice that in the Gauss-Seidel algorithm, the updated values of x_i replace the old values immediately, while in the Jacobi method, all new components of the x-vector are computed before the replacement takes place. In the Jacobi algorithm, the new components of x (denoted by u_i in the pseudocode) can be computed simultaneously, whereas in the Gauss-Seidel method they must be computed serially, since the computation of the new x_i requires all the new values of $x_1, x_2, \ldots, x_{i-1}$. Because of these differences, the Jacobi iteration may be preferable on computers that allow vector or parallel processing. Notice that improved efficiency in the Gauss-Seidel algorithm can be obtained by some preparatory work on the system. In fact, the remarks made about the Jacobi iteration apply here unchanged.

Example 3 Consider the system

$$\begin{bmatrix} 2 & -1 & 0 \\ 1 & 6 & -2 \\ 4 & -3 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -4 \\ 5 \end{bmatrix}$$

Apply the Gauss-Seidel iteration starting with $x^{(0)} = (0, 0, 0)^T$.

Solution After the scaling referred to previously, $D^{-1}Ax = D^{-1}b$ where $D = \text{diag}\,(A)$, the system becomes

$$\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ \frac{1}{6} & 1 & -\frac{1}{3} \\ \frac{1}{2} & -\frac{3}{8} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{2}{3} \\ \frac{5}{8} \end{bmatrix}$$

We refer to this system as $Ax = b$. In the Gauss-Seidel method, Q is taken to be the lower triangular part of A, including the diagonal. The formula defining the iteration is

$$Qx^{(k)} = (Q - A)x^{(k-1)} + b$$

or

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{6} & 1 & 0 \\ \frac{1}{2} & -\frac{3}{8} & 1 \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ x_3^{(k-1)} \end{bmatrix} + \begin{bmatrix} 1 \\ -\frac{2}{3} \\ \frac{5}{8} \end{bmatrix}$$

From this we obtain $x^{(k)}$ by solving a lower triangular system. The pertinent formulas in this example are

$$\begin{aligned} x_1^{(k)} &= \tfrac{1}{2}x_2^{(k-1)} + 1 \\ x_2^{(k)} &= -\tfrac{1}{6}x_1^{(k)} + \tfrac{1}{3}x_3^{(k-1)} - \tfrac{2}{3} \\ x_3^{(k)} &= -\tfrac{1}{2}x_1^{(k)} + \tfrac{3}{8}x_2^{(k)} + \tfrac{5}{8} \end{aligned}$$

The computations yield the following iterates.

$$\begin{aligned} x^{(1)} &= (1.000000,\ -0.833333,\ -0.187500)^T \\ &\vdots \\ x^{(5)} &= (0.622836,\ -0.760042,\ 0.0285661)^T \\ &\vdots \\ x^{(10)} &= (0.620001,\ -0.760003,\ 0.0299981)^T \\ &\vdots \\ x^{(13)} &= (0.620000,\ -0.760000,\ 0.03000000)^T \end{aligned}$$

■

SOR Method

The next example of an important iterative method is known as **successive over-relaxation**, commonly abbreviated as *SOR*. Since we wish to present a general theory for SOR that will apply to matrices and vectors over the field of complex numbers, we first review some of the concepts germane to this setting.

If γ is a complex number, then γ can be written in the form $\gamma = \alpha + i\beta$, where α and β are real and $i^2 = -1$. The **conjugate** of γ is defined to be

$$\overline{\gamma} = \alpha - i\beta$$

The **magnitude** of γ is $|\gamma| = \sqrt{\alpha^2 + \beta^2} = \sqrt{\gamma\overline{\gamma}}$. The space of complex n-vectors is denoted by $\mathbb{C}^n$. In this space, the inner product is defined by the equation

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

Here y^* is the row vector defined by $y^* = (\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n)$ and is called the **conjugate transpose** of y. It is easy to see that

$$\langle x, x\rangle > 0$$
$$\langle x, \lambda y\rangle = \overline{\lambda}\langle x, y\rangle$$
$$\langle x, y\rangle = \overline{\langle y, x\rangle}$$

Then it follows that $\langle \alpha x + \beta y, z\rangle = \alpha\langle x, z\rangle + \beta\langle y, z\rangle$, for scalars α, β and vectors x, y, z, and that $\langle x, Ay\rangle = \langle A^*x, y\rangle$. The **Euclidean norm** of x is

$$\|x\|_2 = \sqrt{\langle x, x\rangle} = \sqrt{x^*x} = \left\{\sum_{i=1}^{n} |x_i|^2\right\}^{1/2}$$

A matrix $A = (a_{ij})$ is said to be **Hermitian** if $A^* = A$. Here $A^* = (\overline{a}_{ji})$ is the **conjugate transpose** of A. Finally, a matrix A is said to be **positive definite** if $\langle Ax, x\rangle > 0$ for all $x \neq 0$. Notice that if A is Hermitian, $\langle Ax, y\rangle = \langle x, Ay\rangle$.

Next, we present a theorem containing conditions for the convergence of the SOR method when A is Hermitian and positive definite.

THEOREM 7 *In the SOR method, suppose that the splitting matrix Q is chosen to be $\alpha D - C$, with α a real parameter, where D is any positive definite Hermitian matrix and C is any matrix satisfying $C + C^* = D - A$. If A is positive definite Hermitian, if Q is nonsingular, and if $\alpha > \frac{1}{2}$, then the SOR iteration converges for any starting vector.*

Proof As in the previous proof, we let $G = I - Q^{-1}A$ and attempt to establish that the spectral radius of G satisfies $\rho(G) < 1$. Let λ be an eigenvalue of G, and let x be an eigenvector corresponding to λ. Put $y = (I - G)x$. The following equations are readily verified.

$$y = x - Gx = x - \lambda x = Q^{-1}Ax \tag{16}$$

$$Q - A = (\alpha D - C) - (D - C - C^*) = \alpha D - D + C^* \tag{17}$$

Using (16), we have

$$(\alpha D - C)y = Qy = Ax \tag{18}$$

Using (17), (18), and (16), we obtain

$$\begin{aligned}(\alpha D - D + C^*)y = (Q - A)y &= Ax - Ay = A(x - y)\\ &= A(x - Q^{-1}Ax) = AGx\end{aligned} \tag{19}$$

From (18) and (19), we have

$$\alpha\langle Dy, y\rangle - \langle Cy, y\rangle = \langle Ax, y\rangle \tag{20}$$

$$\alpha\langle y, Dy\rangle - \langle y, Dy\rangle + \langle y, C^*y\rangle = \langle y, AGx\rangle \tag{21}$$

On adding Equations (20) and (21), we obtain

$$2\alpha\langle Dy, y\rangle - \langle y, Dy\rangle = \langle Ax, y\rangle + \langle y, AGx\rangle \tag{22}$$

$$(2\alpha - 1)\langle Dy, y\rangle = \langle Ax, y\rangle + \langle y, AGx\rangle \tag{23}$$

Here we have used $\langle Dy, y\rangle = \langle y, Dy\rangle$ since D is Hermitian. Since $y = (1-\lambda)x$ and $Gx = \lambda x$, Equation (23) yields

$$\begin{aligned}(2\alpha - 1)|1-\lambda|^2\langle Dx, x\rangle &= (1-\overline{\lambda})\langle Ax, x\rangle + \overline{\lambda}(1-\lambda)\langle x, Ax\rangle \\ &= (1-|\lambda|^2)\langle Ax, x\rangle \end{aligned} \tag{24}$$

because A is Hermitian. If $\lambda \neq 1$, the left side of Equation (24) is positive. Hence, the right side must also be positive, and $|\lambda| < 1$. On the other hand, if $\lambda = 1$ then $y = 0$ from $y = (1-\lambda)x$ and $Ax = 0$ from Equation (18). This contradicts the condition $\langle Ax, x\rangle > 0$ for any $x \neq 0$. Consequently, $\rho(G) < 1$ and the SOR method converges. ∎

A common choice for D and C in the SOR method is to let D be the diagonal of A and to let C be the lower triangular part of A, excluding the diagonal. However, Theorem 7 does not presuppose this choice. Also, the reader should be alert to the fact that in the literature the parameter α is usually denoted by $1/\omega$. Then $0 < \omega < 2$. The question of choosing ω for the most rapid convergence of the SOR iteration is addressed in Young [1971], Varga [1962], Hageman and Young [1981], Wachspress [1966], Issacson and Keller [1966], and many other books.

Iteration Matrices

Suppose that A is partitioned into

$$A = D - C_L - C_U$$

where $D = \text{diag}\,(A)$, C_L is the negative of the strictly lower triangular part of A, and C_U is the negative of the strictly upper triangular part of A. Another partitioning is similar but with block components. In the discretization of partial differential equations, the first partitioning corresponds to single grid points while the latter corresponds to groupings such as by lines or blocks of grid points.

For the basic iterative methods presented in this section, we summarize the key matrices and the methods as follows.

Richardson:

$$\begin{cases} Q = I \\ G = I - A \end{cases}$$

$$x^{(k)} = (I - A)x^{(k-1)} + b$$

Jacobi:

$$\begin{cases} Q = D \\ G = D^{-1}(C_L + C_U) \end{cases}$$

$$Dx^{(k)} = (C_L + C_U)x^{(k-1)} + b$$

Gauss-Seidel:

$$\begin{cases} Q = D - C_L \\ G = (D - C_L)^{-1} C_U \end{cases}$$

$$(D - C_L)x^{(k)} = C_U x^{(k-1)} + b$$

SOR:

$$\begin{cases} Q = \omega^{-1}(D - \omega C_L) \\ G = (D - \omega C_L)^{-1}(\omega C_U + (1 - \omega)D) \end{cases}$$

$$(D - \omega C_L)x^{(k)} = \omega(C_U x^{(k-1)} + b) + (1 - \omega)Dx^{(k-1)}$$

SSOR:

$$\begin{cases} Q = (\omega(2 - \omega))^{-1}(D - \omega C_L)D^{-1}(D - \omega C_U) \\ G = (D - \omega C_U)^{-1}(\omega C_L + (1 - \omega)D)(D - \omega C_L)^{-1}(\omega C_U + (1 - \omega)D) \end{cases}$$

$$\begin{aligned} (D - \omega C_L)x^{(k-1/2)} &= \omega(C_U x^{(k-1)} + b) + (1 - \omega)Dx^{(k-1)} \\ (D - \omega C_U)x^{(k)} &= \omega(C_L x^{(k-1/2)} + b) + (1 - \omega)Dx^{(k-1/2)} \end{aligned}$$

Here we have included another basic iterative method—the **symmetric successive overrelaxation** (SSOR) method. Each iteration of the SSOR method consists of first a *forward* SOR iteration that computes the unknowns in a certain order and then a *backward* SOR *sweep* that solves for them in the opposite order. The *best* choice for the relaxation parameters for the SOR and SSOR methods are intriguing questions with rather complicated answers that we shall not discuss here.

Extrapolation

Next, we present a general technique called **extrapolation** that can be used to improve the convergence properties of a linear iterative process. Consider the iteration formula

$$x^{(k)} = Gx^{(k-1)} + c \tag{25}$$

We introduce a parameter $\gamma \neq 0$ and consider the method (25) to be embedded in a one-parameter family of iteration methods given by

$$\begin{aligned} x^{(k)} &= \gamma(Gx^{(k-1)} + c) + (1 - \gamma)x^{(k-1)} \\ &= G_\gamma x^{(k-1)} + \gamma c \end{aligned} \tag{26}$$

where

$$G_\gamma = \gamma G + (1 - \gamma)I$$

Notice that when $\gamma = 1$ we recover the original iteration in Equation (25).

If the iteration in (26) converges, say to x, then by taking a limit, we get

$$x = \gamma(Gx + c) + (1 - \gamma)x$$

or

$$x = Gx + c$$

since $\gamma \neq 0$. Recall that the objective of the iteration (25) is to produce a solution of the equation $x = Gx + c$. If $G = I - Q^{-1}A$ and $c = Q^{-1}b$, then this corresponds to solving $Ax = b$.

Before attempting to determine an optimum value for the parameter γ, we require a result about eigenvalues.

THEOREM 8 *If λ is an eigenvalue of a matrix A and if p is a polynomial, then $p(\lambda)$ is an eigenvalue of $p(A)$.*

Proof Let $Ax = \lambda x$ with $x \neq 0$. Then $A^2x = \lambda Ax = \lambda^2 x$. By induction and a separate verification for $k = 0$, we have

$$A^k x = \lambda^k x \qquad (k = 0, 1, 2, \ldots)$$

Thus, λ^k is an eigenvalue of A^k. For a polynomial p, write $p(z) = \sum_{k=0}^{m} c_k z^k$. Then

$$p(A)x = \sum_{k=0}^{m} c_k A^k x = \sum_{k=0}^{m} c_k \lambda^k x = p(\lambda)x$$ ∎

By Theorem 5, a criterion that is necessary and sufficient for the convergence of the extrapolated method in Equation (26) is that $\rho(G_\gamma) < 1$. Suppose that we do not know the eigenvalues of G precisely, but know only an interval $[a, b]$ on the real line that contains all of them. By Theorem 8, the eigenvalues of the matrix $G_\gamma \equiv \gamma G + (1-\gamma)I$ lie in the interval whose endpoints are $\gamma a + 1 - \gamma$ and $\gamma b + 1 - \gamma$. Is it possible to select γ so that $\rho(G_\gamma) < 1$?

Denote by $\Lambda(A)$ the set of eigenvalues of any matrix A. Then

$$\rho(G_\gamma) = \max_{\lambda \in \Lambda(G_\gamma)} |\lambda| = \max_{\lambda \in \Lambda(G)} |\gamma\lambda + 1 - \gamma| \leqq \max_{a \leqq \lambda \leqq b} |\gamma\lambda + 1 - \gamma| \qquad (27)$$

We shall prove that if $1 \notin [a, b]$, then γ can be chosen so that $\rho(G_\gamma) < 1$.

THEOREM 9 *If the only information available about the eigenvalues of G is that they lie in the interval $[a, b]$, and if $1 \notin [a, b]$, then the best choice for γ is $2/(2 - a - b)$. With this value of γ, $\rho(G_\gamma) \leqq 1 - |\gamma| d$, where d is the distance from 1 to $[a, b]$.*

Proof Assume the hypotheses, and let γ be as given. Since $1 \notin [a. b]$, either $a > 1$ or $b < 1$. We give the proof in the second case only, leaving the first to the problems. Since $a \leqq b < 1$, it follows that $\gamma > 0$ and $d = 1 - b$. Hence any eigenvalue λ of G_γ satisfies the inequality

$$\gamma a + 1 - \gamma \leqq \lambda \leqq \gamma b + 1 - \gamma \qquad (28)$$

as noted in the preceding paragraphs. Thus

$$\lambda \leqq \gamma b + 1 - \gamma = 1 + \gamma(b - 1) = 1 - \gamma d$$

Furthermore

$$\lambda \geqq \gamma a + 1 - \gamma = \gamma(a + b - 2) + 1 + \gamma(1 - b) = -1 + \gamma d$$

Thus, we have proved that

$$-1 + \gamma d \leqq \lambda \leqq 1 - \gamma d$$

whence $\rho(G_\gamma) \leqq 1 - \gamma d$. To see that our choice of γ is optimal, note that if γ is increased, then the left-hand endpoint of the interval in Equation (28) moves to the left. If that point happens to be an eigenvalue of G_γ, then $\rho(G_\gamma)$ will *increase*. A similar argument applies if γ is decreased. ∎

It should be observed that the extrapolation process just discussed can be applied to methods that are not convergent themselves. All that is required is that the eigenvalues of G be real and lie in an interval that does not contain 1.

If A is a matrix whose eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are all real, we define

$$m(A) = \min_i \lambda_i \qquad M(A) = \max_i \lambda_i \tag{29}$$

Thus, in Theorem 9, we can let $a = m(G)$ and $b = M(G)$.

Example 4 In the case of the Richardson iteration, $Q = I$ and $G = I - A$. If A has only real eigenvalues, then the same is true of G. By Theorem 8, we have

$$M(G) = 1 - m(A), \qquad m(G) = 1 - M(A) \tag{30}$$

If $m(A) > 0$ or $M(A) < 0$, then acceleration is possible in the Richardson iteration. The optimal γ, calculated from Theorem 9, is

$$\gamma = 2/\big[m(A) + M(A)\big]$$

The resulting spectral radius, from $d = m(A)$, is

$$\rho(G_\gamma) = \big[M(A) - m(A)\big]/\big[M(A) + m(A)\big]$$

∎

Example 5 Jacobi iteration can be treated as indicated in Problem 9. We let

$$D = \text{diag}\,(a_{11}, a_{22}, \ldots, a_{nn})$$

and apply Richardson iteration to the problem $D^{-1}Ax = D^{-1}b$. From the result of the preceding example, we conclude that if $m(D^{-1}A) > 0$ or $M(D^{-1}A) < 0$, then acceleration is possible in the Richardson iteration. Furthermore, the spectral radius of the iteration matrix will be

$$\rho(G_\gamma) = \big[M(D^{-1}A) - m(D^{-1}A)\big]/\big[M(D^{-1}A) + m(D^{-1}A)\big]$$

if the optimal $\gamma = 2/[m(D^{-1}A) + M(D^{-1}A)]$ is used. ∎

Chebyshev Acceleration

A more general type of acceleration procedure called **Chebyshev acceleration** can also be applied to a linear iterative algorithm. As before, let us consider a basic iterative method

$$x^{(k)} = Gx^{(k-1)} + c$$

It is assumed that the solution to the problem is a vector x such that $x = Gx + c$. At step k in the process, we shall have computed the vectors $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$, and we ask whether some linear combination of these vectors is perhaps a better approximation to the solution than $x^{(k)}$. We assume that $a_0^{(k)} + a_1^{(k)} + \cdots + a_k^{(k)} = 1$, and set

$$u^{(k)} = \sum_{i=0}^{k} a_i^{(k)} x^{(i)}$$

By familiar techniques, we obtain

$$u^{(k)} - x = \sum_{i=0}^{k} a_i^{(k)}(x^{(i)} - x) = \sum_{i=0}^{k} a_i^{(k)} G^i (x^{(0)} - x)$$

$$= P(G)(x^{(0)} - x)$$

where P is the polynomial defined by $P(z) = \sum_{i=0}^{k} a_i^{(k)} z^i$. Taking norms, we get

$$\|u^{(k)} - x\| \leqq \|P(G)\| \, \|x^{(0)} - x\|$$

Here any vector norm and its subordinate matrix norm may be used. Recall from Theorem 4 that the infimum of $\|P(G)\|$ over all subordinate matrix norms is $\rho(P(G))$; this is the quantity that should be made a minimum. If the eigenvalues μ_i of G lie within some bounded set S in the complex plane, then by Theorem 8,

$$\rho\big(P(G)\big) = \max_{1 \leqq i \leqq n} |P(\mu_i)| \leqq \max_{z \in S} |P(z)|$$

The polynomial P should be chosen to minimize this last expression, subject to the constraint $\sum_{i=0}^{k} a_i = 1$, which is $P(1) = 1$. This is a standard problem in approximation theory for which explicit solutions are known in some cases.

For example, if S is an interval $[a, b]$ on the real line, not containing 1, then a scaled and shifted Chebyshev polynomial solves the problem. The classic Chebyshev polynomial T_k $(k \geqq 1)$ is the unique polynomial of degree k that minimizes the expression

$$\max_{-1 \leqq z \leqq 1} |T_k(z)|$$

subject to the constraint that the leading coefficient is 2^{k-1}. These polynomials can be generated recursively by the formulæ

$$\begin{cases} T_0(z) = 1, \qquad T_1(z) = z \\ T_k(z) = 2zT_{k-1}(z) - T_{k-2}(z) \qquad (k \geqq 2) \end{cases}$$

Now suppose that the eigenvalues of G are contained in an interval $[a, b]$ that does not contain 1, say $b < 1$. We are interested in the following min-max problem

$$\min_{P_k(1)=1} \left\{ \max_{a \leqq z \leqq b} |P_k(z)| \right\}$$

The solution of this extremum problem is contained in the next four lemmas. These results involve the Chebyshev polynomials, T_k, which are discussed above and in Section 6.1. The set of polynomials having degree at most k is denoted by Π_k.

LEMMA 1 *Let $\beta \in \mathbb{R} \setminus (-1, 1)$. If $p \in \Pi_k$ and $p(\beta) = 1$ then $\|p\| \geqq |\alpha|$, where $\alpha = 1/T_k(\beta)$ and $\|p\| = \max_{-1 \leqq t \leqq 1} |p(t)|$.*

Proof Suppose that p satisfies the hypotheses but not the conclusion. Let $t_i = \cos(i\pi/k)$ for $0 \leqq i \leqq k$. These points are the extrema of T_k. Indeed,

$$T_k(t_i) = \cos(k \cos^{-1} t_i) = \cos i\pi = (-1)^i$$

Let $\sigma = \operatorname{sgn} \alpha$. Then

$$\sigma(-1)^i \left[\alpha T_k(t_i) - p(t_i)\right] \geqq |\alpha| - \|p\| > 0$$

This shows that the polynomial $\alpha T_k - p$ takes alternately positive and negative values at the points $t_0, t_1, \ldots, t_k$. Hence, this polynomial has at least k zeros in the interval $(-1, 1)$. It also vanishes at the point β, giving a total of at least $k + 1$ zeros. Since $\alpha T_k - p$ is of degree at most k, it must be 0, a contradiction. ∎

LEMMA 2 *Let $a < b < 1$. If $p \in \Pi_k$ and $p(1) = 1$, then $\|p\| \geqq 1/T_k(w(1))$. Here*

$$\|p\| = \max_{a \leqq t \leqq b} |p(t)| \qquad \text{and} \qquad w(t) = (2t - b - a)/(b - a)$$

The inequality becomes an equality if $p = T_k \circ w / T_k(w(1))$.

Proof Put $\beta = w(1)$ and $p = q \circ w$, with $q \in \Pi_k$. Note that $1 = p(1) = q(w(1)) = q(\beta)$ and $\beta > 1$. By the preceding lemma, it follows that $\|q\|_{[-1,1]} \geqq 1/T_k(\beta)$. Equivalently, we have $\|p\|_{[a,b]} \geqq 1/T_k(w(1))$. If $p = T_k \circ w / T_k(w(1))$, then obviously $p(1) = 1$ and $\|p\|_{[a,b]} = \|T_k\|_{[-1,1]} / T_k(w(1)) = 1/T_k(w(1))$. ∎

LEMMA 3 *The polynomials $P_k = T_k \circ w / T_k(w(1))$ can be generated by a recurrence relation:*

$$\begin{cases} P_0(t) = 1 \qquad P_1(t) = (2t - b - a)/(2 - b - a) \\ P_k(t) = \rho_k P_1(t) P_{k-1}(t) + (1 - \rho_k) P_{k-2}(t) \qquad (k \geqq 2) \end{cases}$$

in which the coefficients ρ_k are obtained from the equations

$$\rho_1 = 2 \quad \rho_k = (1 - \alpha \rho_{k-1})^{-1} \qquad \alpha = \left[2w(1)\right]^{-2} \qquad (k \geqq 2)$$

Proof Define $\beta_k = T_k(w(1))$. With the aid of the relation $T_k(t) = 2tT_{k-1}(t) - T_{k-2}(t)$ and $\beta_k P_k(t) = T_k(w(t))$, we obtain

$$\begin{aligned} P_k(t) &= \beta_k^{-1} T_k\big(w(t)\big) = \beta_k^{-1}\Big[2w(t)T_{k-1}\big(w(t)\big) - T_{k-2}\big(w(t)\big)\Big] \\ &= 2\beta_k^{-1}\beta_{k-1}w(t)P_{k-1}(t) - \beta_k^{-1}\beta_{k-2}P_{k-2}(t) \\ &= 2\beta_k^{-1}\beta_{k-1}w(1)P_1(t)P_{k-1}(t) - \beta_k^{-1}\beta_{k-2}P_{k-2}(t) \end{aligned}$$

Here we employed the easily verified equation $w(1)P_1(t) = w(t)$. It is convenient to define $\rho_k = 2\beta_k^{-1}\beta_{k-1}w(1) = \alpha^{-1/2}\beta_k^{-1}\beta_{k-1}$. Using again the recurrence relation for Chebyshev polynomials, we obtain

$$\begin{aligned} \beta_k = T_k\big(w(1)\big) &= 2w(1)T_{k-1}\big(w(1)\big) - T_{k-2}\big(w(1)\big) = \alpha^{-\frac{1}{2}}\beta_{k-1} - \beta_{k-2} \\ 1 &= 2w(1)\beta_k^{-1}\beta_{k-1} - \beta_k^{-1}\beta_{k-2} = \rho_k - \beta_k^{-1}\beta_{k-2} \end{aligned}$$

Thus our recurrence relation for P_k can be written as

$$P_k = \rho_k P_1 P_{k-1} + (1 - \rho_k)P_{k-2}$$

The coefficients ρ_k satisfy the equation

$$\begin{aligned} \rho_k &= \alpha^{-\frac{1}{2}}\beta_k^{-1}\beta_{k-1} = \alpha^{\frac{1}{2}}\beta_{k-1}\big[\alpha^{-\frac{1}{2}}\beta_{k-1} - \beta_{k-2}\big]^{-1} \\ &= \alpha^{-\frac{1}{2}}\big[\alpha^{-\frac{1}{2}} - \beta_{k-1}^{-1}\beta_{k-2}\big]^{-1} \\ &= \alpha^{-1}\Big\{\alpha^{-1} - \alpha^{-\frac{1}{2}}\beta_{k-1}^{-1}\beta_{k-2}\Big\}^{-1} \\ &= \alpha^{-1}\Big\{\alpha^{-1} - \rho_{k-1}\Big\}^{-1} \\ &= (1 - \alpha\rho_{k-1})^{-1} \end{aligned}$$

∎

LEMMA 4 *The vectors $u^{(k)}$ in the Chebyshev acceleration method can be computed recursively from an arbitrary starting vector $u^{(0)}$ by the formulæ*

$$u^{(1)} = \gamma[Gu^{(0)} + c] + (1 - \gamma)u^{(0)} \qquad \left(\gamma = \frac{2}{2 - b - a}\right)$$

$$u^{(k)} = \rho_k\big[\gamma(Gu^{(k-1)} + c) + (1 - \gamma)u^{(k-1)}\big] + (1 - \rho_k)u^{(k-2)}$$

Proof We retain all the notation established previously. In particular,

$$u^{(k)} = \sum_{i=0}^{k} a_i^{(k)} x^{(i)} \qquad P_k(t) = \sum_{i=0}^{k} a_i^{(k)} t^i$$

It follows that $u^{(0)} = x^{(0)}$ and that $u^{(1)}$ is given by the formula in the lemma. This is left to the reader to verify. From equations proved earlier, with x satisfying $x = Gx + c$,

$$\begin{aligned} u^{(k)} - x &= P_k(G)(u^{(0)} - x) \\ &= \big[\rho_k P_1(G)P_{k-1}(G) + (1-\rho_k)P_{k-2}(G)\big](u^{(0)} - x) \\ &= \rho_k P_1(G)(u^{(k-1)} - x) + (1-\rho_k)(u^{(k-2)} - x) \end{aligned}$$

This can be written in the form

$$u^{(k)} = \rho_k P_1(G)u^{(k-1)} + (1-\rho_k)u^{(k-2)} + \rho_k\big[I - P_1(G)\big]x$$

An easy calculation reveals that the last term in this equation equals $\rho_k \gamma c$. Finally, we note that

$$P_1(G) = \gamma G + (1-\gamma)I$$ ∎

As in our analysis of the extrapolation method, we can obtain an upper bound on the spectral radius of $P_k(G)$:

$$\begin{aligned} \rho\big(P_k(G)\big) &= \max_{\lambda \in \Lambda(P_k(G))} |\lambda| = \max_{\lambda \in \Lambda(G)} |P_k(\lambda)| \\ &\leqq \max_{a \leqq \lambda \leqq b} |P_k(\lambda)| = 1/T_k\big(w(1)\big) \end{aligned}$$

Here an appeal to Lemma 2 has been made. For computing this bound we can use the result of Problem 36 and arrive at these formulæ:

$$\frac{1}{T_k(w(1))} = \frac{2}{b^n + b^{-n}} \qquad b = t + \sqrt{t^2 - 1} \qquad t = w(1)$$

It can be shown that Chebyshev acceleration is an order of magnitude faster than extrapolation. See Hageman and Young [1981] or Kincaid and Young [1979] for additional details.

Example 6 Solve the problem

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -4 \\ 0 \\ 4 \\ -4 \end{bmatrix}$$

using the Chebyshev acceleration of the Jacobi method.

Solution First we scale the system by $D^{-1}Ax = D^{-1}b$, where $D = \operatorname{diag}(A)$, obtaining a Jacobi iteration matrix whose eigenvalues lie in the interval $[-\frac{1}{2}, \frac{1}{2}]$. The convergence of this basic method can be speeded up by applying Chebyshev acceleration. Beginning with the initial vector $u = (0, 0, 0, 0)^T$, we obtain convergence to the approximate solution $u = (-0.999996, -0.500002, 0.500002, -0.999996)^T$ in ten iterations using a computer similar to the MARC-32. ∎

An algorithm for the Chebyshev acceleration method can be written as follows.

input u, a, b, M, δ
$\gamma \leftarrow 2/(2 - b - a)$
$\alpha \leftarrow [\frac{1}{2}(b - a)/(2 - b - a)]^2$
output $0, u$
call `Extrap` (γ, n, G, c, u, v)
output $1, v$
$\rho \leftarrow 1/(1 - 2\alpha)$
call `Cheb` $(\rho, \gamma, n, G, c, u, v)$
output $2, u$
for $k = 3, 5, \ldots, M$ **step** 2 **do**
    $\rho \leftarrow (1 - \rho\alpha)$
    call `Cheb` $(\rho, \gamma, n, G, c, v, u)$
    output k, v
    $\rho \leftarrow 1/(1 - \rho\alpha)$
    call `Cheb` $(\rho, \gamma, n, G, c, u, v)$
    output $k + 1, u$
    if $\|u - v\|_\infty < \delta$ **stop**
end

Here the first step of this method is just the extrapolation procedure, and each successive iteration involves two acceleration steps. Two subroutines or procedures are needed to carry out the basic iteration steps, namely, `Extrap` and `Cheb`; they are given below. Notice that by appropriate calls to the subroutine or procedure `Cheb`, we obtain an automatic interchange of the desired vectors.

procedure `Extrap` (γ, n, G, c, u, v)
$v \leftarrow \gamma c + (1 - \gamma)u$
$v \leftarrow \gamma G u + v$
return

procedure `Cheb` $(\rho, \gamma, n, G, c, u, v)$
$u \leftarrow \rho\gamma c + \rho(1 - \gamma)v + (1 - \rho)u$
$u \leftarrow \rho\gamma G v + u$
return

PROBLEM SET *4.6

1. Prove that if A is diagonally dominant and if Q is chosen as in the Jacobi method, then
$$\rho(I - Q^{-1}A) < 1$$

2. Prove that if A has this property (*unit row diagonally dominant*)

$$a_{ii} = 1 > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad (1 \leqq i \leqq n)$$

then the Richardson iteration is successful.

3. Repeat Problem 2 with this assumption (*unit column diagonally dominant*)

$$a_{jj} = 1 > \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| \qquad (1 \leqq j \leqq n)$$

4. Prove that if A has the property in Problem 2, then the following iteration will solve $Ax = b$ (in the limit):

for $k = 1, 2, 3, \ldots$
  for $i = 1, 2, \ldots, n$ **do**
    $x_i \leftarrow x_i + b_i - \sum_{j=1}^{n} a_{ij} x_j$
  end
end

5. Let $\| \cdot \|$ be a norm on $\mathbb{R}^n$, and let S be an $n \times n$ nonsingular matrix. Define $\|x\|' = \|Sx\|$, and prove that $\| \cdot \|'$ is a norm.
6. (Continuation) Let $\| \cdot \|$ be a subordinate matrix norm, and let S be a nonsingular matrix. Define $\|A\|' = \| SAS^{-1} \|$, and show that $\| \cdot \|'$ is a subordinate matrix norm.
7. Using Q as in the Gauss-Seidel method, prove that if A is diagonally dominant, then $\| I - Q^{-1}A \|_\infty < 1$.
8. Prove that $\rho(A) < 1$ if and only if $\lim_{k \to \infty} A^k x = 0$ for every x.
9. Prove that if the ith equation in the system $Ax = b$ is divided by a_{ii} and if the Richardson iteration is then applied, the result is the same as applying the Jacobi iteration in the first place.
10. Which of the norm axioms are satisfied by the spectral radius function ρ and which are not? Give proofs and examples, as appropriate.
11. Consider (for fixed n) the set of upper triangular $n \times n$ matrices. Show that this set is a vector space. Prove that the spectral radius function ρ is a *pseudonorm* on the vector space since it satisfies all the norm axioms, except that $\rho(A)$ can be zero if $A \neq 0$.
12. Explain why, in the proof of Theorem 3, we cannot let $\varepsilon \to 0$ and conclude that A is similar to a diagonal matrix.
13. Let A be invertible, and let f be a function of the form $f(z) = \sum_{j=-m}^{m} c_j z^j$. Show that if λ is an eigenvalue of A then $f(\lambda)$ is an eigenvalue of $f(A)$.
14. Prove that the eigenvalues of a Hermitian matrix are real.
15. Let A be diagonally dominant, and let Q be the lower triangular part of A, as in the Gauss-Seidel method. Prove that $\rho(I - Q^{-1}A)$ is no greater than the largest of the ratios

$$r_i = \left\{ \sum_{j=i+1}^{n} |a_{ij}| \right\} \Big/ \left\{ |a_{ii}| - \sum_{j=1}^{i-1} |a_{ij}| \right\}$$

16. Prove that if A is nonsingular, then AA^* is positive definite.
17. Prove that if A is positive definite, then its eigenvalues are positive.
18. Prove that if A is positive definite, then so are $A^2, A^3, \ldots$ as well as $A^{-1}, A^{-2}, \ldots$.
19. Is there a matrix A such that $\rho(A) < \| A \|$ for all subordinate matrix norms?
20. Prove that if $\rho(A) < 1$, then $I - A$ is invertible and $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$.
21. Is the inequality $\rho(AB) \leq \rho(A)\rho(B)$ true for all pairs of $n \times n$ matrices? Is your answer the same when A and B are upper triangular?
22. Show that the basic iteration process given by Equation (3) is equivalent to the following. Given $x^{(k)}$, compute $r^{(k)} = b - Ax^{(k)}$, solve for $z^{(k)}$ in the equation $Qz^{(k)} = r^{(k)}$, and define $x^{(k+1)} = x^{(k)} + z^{(k)}$.
23. Do the Hermitian matrices of order n form a vector space over the complex field?
24. Using the notation of Problem 22, show that

$$r^{(k+1)} = (I - AQ^{-1})r^{(k)}$$
$$z^{(k+1)} = (I - Q^{-1}A)z^{(k)}$$

25. Show that for nonsingular matrices A and B, $\rho(AB) = \rho(BA)$. (Prove a stronger result, if possible.) What is the relevance of this fact to Problem 24?
26. What are the necessary and sufficient conditions on a diagonal matrix D in order that for any A, the positive definiteness of A implies that of DA?
27. [c] Program the Gauss-Seidel method and test it on these examples:

$$\text{(a)} \begin{cases} 3x + y + z = 5 \\ x + 3y - z = 3 \\ 3x + y - 5z = -1 \end{cases} \qquad \text{(b)} \begin{cases} 3x + y + z = 5 \\ 3x + y - 5z = -1 \\ x + 3y - z = 3 \end{cases}$$

Analyze what happens when these systems are solved by simple Gaussian elimination without pivoting.

28. [c] Apply the Gauss-Seidel iteration to the system in which

$$A = \begin{bmatrix} 0.96326 & 0.81321 \\ 0.81321 & 0.68654 \end{bmatrix} \qquad b = \begin{bmatrix} 0.88824 \\ 0.74988 \end{bmatrix}$$

Use $(0.33116,\ 0.70000)^T$ as the starting point and explain what happens.

29. Prove that the Gauss-Seidel method is a special case of the SOR method.
30. Show that these matrices

$$\mathcal{R} = I - A$$
$$\mathcal{J} = I - D^{-1}A$$
$$\mathcal{G} = I - (D - C_L)^{-1}A$$
$$\mathcal{L}_\omega = I - \omega(D - \omega C_L)^{-1}A$$
$$\mathcal{U}_\omega = I - \omega(D - \omega C_U)^{-1}A$$
$$\mathcal{S}_\omega = I - \omega(2 - \omega)(D - \omega C_U)^{-1}D(D - \omega C_L)^{-1}A$$

are the iteration matrices for the Richardson, Jacobi, Gauss-Seidel, forward SOR, backward SOR, and SSOR methods, respectively. Then show that the splitting matrices Q and iteration matrices G given in this section are correct.

31. Find the explicit form for the iteration matrix $I - Q^{-1}A$ in the Gauss-Seidel method when

$$A = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

32. Characterize the family of all $n \times n$ nonsingular matrices A for which one step of the Gauss-Seidel algorithm solves $Ax = b$, starting at the vector $x = 0$.
33. Give an example of a matrix A that is not diagonally dominant, yet the Gauss-Seidel method applied to $Ax = b$ converges.
34. How does the Chebyshev acceleration method simplify if the basic method is the Jacobi method?
35. Prove that if the number $\delta = \|I - Q^{-1}A\|$ is less than 1, then

$$\|x^{(k)} - x\| \leqq \frac{\delta}{1-\delta}\|x^{(k)} - x^{(k-1)}\|$$

36. Prove that

$$T_n(t) = \tfrac{1}{2}(b^n + b^{-n}) \qquad b = t + \sqrt{t^2 - 1}$$

37. Prove Theorem 9 in the case $a > 1$.
38. Prove that a positive definite matrix A is Hermitian using the definition; namely, A is positive definite if $x^*Ax > 0$ for all nonzero $x \in \mathbb{C}^n$. Thus, in particular, a real positive definite matrix A must be symmetric if this definition is used. However, if the alternative definition $x^TAx > 0$ for all nonzero $x \in \mathbb{R}^n$ is used, then a real positive definite matrix need not be symmetric.

*4.7 Steepest Descent and Conjugate Gradient Methods

In this section, some special methods will be developed for solving the system

$$Ax = b$$

for the case when A is a real $n \times n$ *symmetric* and *positive definite* matrix. These hypotheses mean that

$$A^T = A$$

and

$$x^TAx > 0 \quad \text{for} \quad x \neq 0$$

Throughout we use the inner-product notation for real vectors x and y

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

Some immediate properties are

(i) $\langle x, y\rangle = \langle y, x\rangle$
(ii) $\langle \alpha x, y\rangle = \alpha\langle x, y\rangle$, for any constant α
(iii) $\langle x + y, z\rangle = \langle x, z\rangle + \langle y, z\rangle$
(iv) $\langle x, Ay\rangle = \langle A^T x, y\rangle$

By property (i), the order of the arguments in properties (ii)–(iii) can be reversed.

We begin by establishing the equivalence of two numerical problems involving A and b.

LEMMA *If A is symmetric and positive definite, then the problem of solving $Ax = b$ is equivalent to the problem of minimizing the quadratic form*

$$q(x) = \langle x, Ax\rangle - 2\langle x, b\rangle$$

Proof First, let us find out how the function q behaves along a *one-dimensional ray*. Here we consider $x + tv$, where x and v are vectors and t is a scalar. Figure 4.2 shows why this can be thought of as a one-dimensional ray.

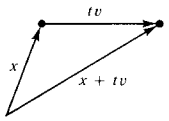


Figure 4.2 Example of one-dimensional ray

A straightforward calculation reveals that for a scalar t we have

$$\begin{aligned} q(x + tv) &= \langle x + tv, A(x + tv)\rangle - 2\langle x + tv, b\rangle \\ &= \langle x, Ax\rangle + t\langle x, Av\rangle + t\langle v, Ax\rangle + t^2\langle v, Av\rangle - 2\langle x, b\rangle - 2t\langle v, b\rangle \\ &= q(x) + 2t\langle v, Ax\rangle - 2t\langle v, b\rangle + t^2\langle v, Av\rangle \\ &= q(x) + 2t\langle v, Ax - b\rangle + t^2\langle v, Av\rangle \end{aligned} \tag{1}$$

since $A^T = A$. Note that the coefficient of t^2 in Equation (1) is positive. Thus the quadratic function on the ray has a minimum and not a maximum. From Equation (1) we compute the derivative with respect to t as

$$\frac{d}{dt}q(x + tv) = 2\langle v, Ax - b\rangle + 2t\langle v, Av\rangle \tag{2}$$

The minimum of q along the ray occurs when the derivative in (2) is zero. The value of t that yields the minimum point is therefore

$$\hat{t} = \langle v, b - Ax\rangle / \langle v, Av\rangle \tag{3}$$

Using this value, $\widehat{t}$, we compute the minimum of q on the ray:

$$\begin{aligned} q(x+\widehat{t}v) &= q(x) + \widehat{t}\,[2\langle v, Ax-b\rangle + r\langle v, Av\rangle] \\ &= q(x) + \widehat{t}\,[2\langle v, Ax-b\rangle + \langle v, b-Ax\rangle] \\ &= q(x) - \widehat{t}\,\langle v, b-Ax\rangle \\ &= q(x) - \langle v, b-Ax\rangle^2/\langle v, Av\rangle \end{aligned} \tag{4}$$

Our calculation shows that a reduction of the value of q always occurs in passing from x to $x+\widehat{t}v$ unless v is *orthogonal* to the residual—that is, $\langle v, b-Ax\rangle = 0$. If x is not a solution of the system $Ax = b$, then many vectors v exist satisfying $\langle v, b-Ax\rangle \neq 0$. Hence, if $Ax \neq b$, then x does not minimize q. On the other hand, if $Ax = b$, then there is no ray emanating from x on which q takes a lesser value than $q(x)$. Hence, such an x is a minimum point of q. ∎

The preceding proof suggests an iterative method for solving $Ax = b$. We proceed by minimizing q along a succession of rays. At the kth step in such an algorithm $x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(k)}$ would be available. Then by use of some rule, a suitable search direction $v^{(k)}$ is chosen. The next point in our sequence is

$$x^{(k+1)} = x^{(k)} + t_k v^{(k)}$$

where

$$t_k = \frac{\langle v^{(k)}, b - Ax^{(k)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle}$$

Many iterative methods are of the general form

$$x^{(k+1)} = x^{(k)} + t_k v^{(k)}$$

for specific values of the scalar t_k and the vectors $v^{(k)}$. If $\|v^{(k)}\| = 1$, then t_k measures the distance we move from $x^{(k)}$ to obtain $x^{(k+1)}$. (See Figure 4.3.)

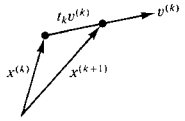


Figure 4.3 Moving along direction vector $v^{(k)}$

Steepest Descent

The method of **steepest descent** is an algorithm of the type just described. It is defined by stipulating that $v^{(k)}$ should be the negative gradient of q at $x^{(k)}$. It turns out that this negative gradient points in the direction of the residual, $r^{(k)} = b - Ax^{(k)}$. (See Problem 1.) A formal description of steepest descent then goes as follows:

input $x^{(0)}, A, b, M$
output $0, x^{(0)}$
for $k = 0, 1, 2, \ldots, M-1$ **do**
  $v^{(k)} \leftarrow b - Ax^{(k)}$
  $t_k \leftarrow \langle v^{(k)}, v^{(k)} \rangle / \langle v^{(k)}, Av^{(k)} \rangle$
  $x^{(k+1)} \leftarrow x^{(k)} + t_k v^{(k)}$
output $k+1, x^{(k+1)}$
end

In the actual programming of this algorithm, the successive vectors $x^{(0)}, x^{(1)}, \ldots$ need not be saved; the *current* x-vector can be overwritten. The same remark holds for the direction vectors $v^{(0)}, v^{(1)}, \ldots$. Thus, we could write alternatively

input x, A, b, M
output $0, x$
for $k = 1, 2, 3, \ldots, M$ **do**
  $v \leftarrow b - Ax$
  $t \leftarrow \langle v, v \rangle / \langle v, Av \rangle$
  $x \leftarrow x + tv$
  output k, x
end

The method of steepest descent is rarely used on this problem because it is too slow. How it might progress on a two-dimensional problem is depicted in Figure 4.4, where we have shown the contours (or *level-lines*) of the quadratic form q.

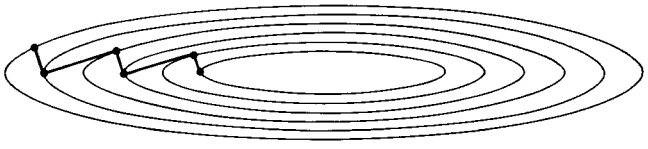


Figure 4.4 Geometric interpretation of steepest descent

Conjugate Directions

A family of methods called **conjugate direction** methods shares the basic strategy of minimizing the quadratic function along a succession of rays. Usually these search directions are determined within the solution process, one at a time. However, we shall discuss first the case when they are prescribed at the beginning.

Assuming that A is an $n \times n$ symmetric and positive definite matrix, suppose that a set of vectors $\{u^{(1)}, u^{(2)}, \ldots, u^{(n)}\}$ is provided and has the property

$$\langle u^{(i)}, Au^{(j)} \rangle = \delta_{ij} \qquad (1 \leqq i, j \leqq n)$$

This property is called **A-orthonormality** and is an obvious generalization of ordinary orthonormality. We shall see that if these vectors are used as the search directions in the step-by-step minimization of the quadratic function q, then the solution is obtained by the nth step. Before giving the formal result, we observe that the A-orthonormality condition can be expressed as a matrix equation

$$U^T AU = I$$

where U is the $n \times n$ matrix whose columns are $u^{(1)}, u^{(2)}, \ldots, u^{(n)}$. From this it is clear that A and U are nonsingular, and the columns $u^{(1)}, u^{(2)}, \ldots, u^{(n)}$ form a basis for $\mathbb{R}^n$.

THEOREM 1 *Let $\{u^{(1)}, u^{(2)}, \ldots, u^{(n)}\}$ be an A-orthonormal system. Define*

$$x^{(i)} = x^{(i-1)} + \langle b - Ax^{(i-1)}, u^{(i)} \rangle u^{(i)} \qquad (1 \leqq i \leqq n)$$

in which $x^{(0)}$ is an arbitrary point of $\mathbb{R}^n$. Then $Ax^{(n)} = b$.

Proof Define $t_i = \langle b - Ax^{(i-1)}, u^{(i)} \rangle$ so that the recursive formula becomes simply

$$x^{(i)} = x^{(i-1)} + t_i u^{(i)}$$

From this and with the aid of the relation $\langle Au^{(j)}, u^{(i)} \rangle = \delta_{ij}$, we deduce the following equations:

$$\begin{aligned} Ax^{(i)} &= Ax^{(i-1)} + t_i Au^{(i)} \\ Ax^{(n)} &= Ax^{(0)} + t_1 Au^{(1)} + \cdots + t_n Au^{(n)} \\ \langle Ax^{(n)} - b, u^{(i)} \rangle &= \langle Ax^{(0)} - b, u^{(i)} \rangle + t_i \end{aligned}$$

We can show that the right side of this last equation is 0 as follows:

$$\begin{aligned} t_i &= \langle b - Ax^{(i-1)}, u^{(i)} \rangle \\ &= \langle b - Ax^{(0)}, u^{(i)} \rangle + \langle Ax^{(0)} - Ax^{(1)}, u^{(i)} \rangle + \cdots + \langle Ax^{(i-2)} - Ax^{(i-1)}, u^{(i)} \rangle \\ &= \langle b - Ax^{(0)}, u^{(i)} \rangle + \langle -t_1 Au^{(1)}, u^{(i)} \rangle + \cdots + \langle -t_{i-1} Au^{(i-1)}, u^{(i)} \rangle \\ &= \langle b - Ax^{(0)}, u^{(i)} \rangle \end{aligned}$$

Our analysis shows that $Ax^{(n)} - b$ is orthogonal (in the usual sense) to the vectors $u^{(1)}, u^{(2)}, \ldots, u^{(n)}$, and must therefore be zero. ∎

We shall now describe a concrete realization of this method. Let A be a symmetric and positive definite $n \times n$ matrix. The equation

$$\langle x, y \rangle_A = \langle x, Ay \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i y_j$$

defines an inner product, as is easily verified. There is a corresponding quadratic norm, $\|x\|_A^2 = \langle x, x \rangle_A$. If the Gram-Schmidt process is used with this new inner product on the set of standard unit vectors $\{e^{(1)}, e^{(2)}, \ldots, e^{(n)}\}$, the result is an A-orthonormal system $\{u^{(1)}, u^{(2)}, \ldots, u^{(n)}\}$. The Gram-Schmidt process (discussed more fully in Section 5.3) is described as follows

$$u^{(i)} = \left(\|v^{(i)}\|_A \right)^{-1} v^{(i)} \quad \text{where} \quad v^{(i)} = e^{(i)} - \sum_{j<i} \langle e^{(i)}, u^{(j)} \rangle_A \, u^{(j)}$$

Because of the special nature of the vectors $e^{(i)}$, these formulæ reduce to

$$u^{(i)} = \left(\|v^{(i)}\|_A \right)^{-1} v^{(i)} \quad \text{where} \quad v^{(i)} = e^{(i)} - \sum_{j<i} (Au^{(j)})_i \, u^{(j)}$$

For example, $u^{(1)} = (1/\sqrt{a_{11}})e^{(1)}$. The formulæ indicate that each $u^{(i)}$ is a linear combination of the unit vectors $e^{(1)}, e^{(2)}, \ldots, e^{(i)}$. The matrix U whose columns are $u^{(1)}, u^{(2)}, \ldots, u^{(n)}$ is therefore upper triangular. The equation $U^T A U = I$ mentioned previously gives us $A = (U^T)^{-1} U^{-1}$, which is an LU-factorization. The solution of $Ax = b$ is thus being computed by a variant of Gaussian elimination, without pivoting.

In numerical work, it is more convenient to employ A-orthogonal systems instead of A-orthonormal systems. A set of vectors $v^{(1)}, v^{(2)}, \ldots$ is **A-orthogonal** if $\langle v^{(i)}, Av^{(j)} \rangle = 0$ whenever $i \neq j$. From the A-orthogonal system we pass to an A-orthonormal system by the normalization process:

$$u^{(i)} = \left(\|v^{(i)}\|_A \right)^{-1} v^{(i)}$$

If each $v^{(i)}$ is nonzero vector and if A is positive definite matrix, then the $u^{(i)}$ will form an A-orthonormal system. The analogue of Theorem 1 is as follows.

THEOREM 2 *Let $\{v^{(1)}, v^{(2)}, \ldots, v^{(n)}\}$ be an A-orthogonal system of nonzero vectors for a symmetric and positive definite $n \times n$ matrix A. Define*

$$x^{(i)} = x^{(i-1)} + \frac{\langle b - Ax^{(i-1)}, v^{(i)} \rangle}{\langle v^{(i)}, Av^{(i)} \rangle} \, v^{(i)} \qquad (1 \leq i \leq n)$$

in which $x^{(0)}$ is arbitrary. Then $Ax^{(n)} = b$.

Conjugate Gradient Method

The **conjugate gradient** method of Hestenes and Stiefel [1952] is a particular type of conjugate direction method. It applies to a system $Ax = b$ in which A is symmetric and positive definite. In the conjugate gradient method, the search directions $v^{(i)}$ referred to in Theorem 2 are chosen one by one during the iterative process and form an A-orthogonal system. The distinguishing property, however, is that the residuals, $r^{(i)} = b - Ax^{(i)}$, form an orthogonal system in the ordinary sense; that is, $\langle r^{(i)}, r^{(j)} \rangle = 0$ if $i \neq j$.

The conjugate gradient method can be recommended over simple Gaussian elimination if A is very large and sparse. Theoretically, the conjugate gradient algorithm will yield the solution of the system $Ax = b$ in at most n steps. In practice, however, the algorithm is used as an iterative method to produce a sequence of vectors converging to the solution. In an ill-conditioned problem, roundoff errors often prevent the algorithm from furnishing a sufficiently precise solution at the nth step. When the conjugate gradient method was introduced by Hestenes and Stiefel [1952], there was initially much excitement over this new method. However, interest in it quickly waned when it was discovered that the finite-termination property was not obtained in practice. For a *direct* method this was an undesirable property. Some 20 years later, there was renewed interest in the method when it was viewed as an *iterative* method. Not obtaining the solution to full precision after n steps is the expected behavior of an iterative method. In fact, we hope for a satisfactory answer in many fewer than n steps for extremely large systems. In well-conditioned problems, the number of iterations necessary for satisfactory convergence of the conjugate gradient method can be much less than the order of the system. The history of the method has been written by Golub and O'Leary [1989].

The formal **conjugate gradient** algorithm follows. (This pseudocode is not designed for computer execution. A suitable algorithm for that purpose is given later.)

input $x^{(0)}, M, A, b, \varepsilon$
$r^{(0)} \leftarrow b - Ax^{(0)}$
$v^{(0)} \leftarrow r^{(0)}$
output $0, x^{(0)}, r^{(0)}$
for $k = 0, 1, 2, \ldots, M-1$ **do**
  if $v^{(k)} = 0$ **then stop**
  $t_k \leftarrow \langle r^{(k)}, r^{(k)} \rangle / \langle v^{(k)}, Av^{(k)} \rangle$
  $x^{(k+1)} \leftarrow x^{(k)} + t_k v^{(k)}$
  $r^{(k+1)} \leftarrow r^{(k)} - t_k A v^{(k)}$
  if $\|r^{(k+1)}\|_2^2 < \varepsilon$ **then stop**
  $s_k \leftarrow \langle r^{(k+1)}, r^{(k+1)} \rangle / \langle r^{(k)}, r^{(k)} \rangle$
  $v^{(k+1)} \leftarrow r^{(k+1)} + s_k v^{(k)}$
  output $k+1, x^{(k+1)}, r^{(k+1)}$
end

In the algorithm, if $v^{(k)} = 0$, then $x^{(k)}$ is (theoretically) a solution of the linear system $Ax = b$. The computer realization of the algorithm requires storage for four vectors,

namely, $x^{(k)}$, $r^{(k)}$, $v^{(k)}$, and $Av^{(k)}$. Provision must also be made for computing the products $Av^{(k)}$. (This may or may not require the storage of the full matrix A.) The work per iteration is modest, amounting to a single matrix-vector product for $Av^{(k)}$ and just two inner products (after the first iterations) for $\langle v^{(k)}, Av^{(k)}\rangle$ and $\langle r^{(k+1)}, r^{(k+1)}\rangle$. Notice that the stopping test involves $\|r^{(k+1)}\|_2^2 = \langle r^{(k+1)}, r^{(r+1)}\rangle$, which is easy to compute since $r^{(k+1)}$ is already available. A computer code for the **conjugate gradient** method should therefore be based on the following algorithm.

input $x, A, b, M, \varepsilon, \delta$
$r \leftarrow b - Ax$
$v \leftarrow r$
$c \leftarrow \langle r, r\rangle$
for $k = 1, 2, \ldots, M$ **do**
  if $\langle v, v\rangle^{1/2} < \delta$ **then stop**
  $z \leftarrow Av$
  $t \leftarrow c/\langle v, z\rangle$
  $x \leftarrow x + tv$
  $r \leftarrow r - tz$
  $d \leftarrow \langle r, r\rangle$
  if $d^2 < \varepsilon$ **then stop**
  $v \leftarrow r + (d/c)v$
  $c \leftarrow d$
  output k, x, r
end

THEOREM 3 *In the conjugate gradient algorithm, for any integer $m < n$, if $v^{(0)}, v^{(1)}, \ldots, v^{(m)}$ are all nonzero vectors, then $r^{(i)} = b - Ax^{(i)}$ for $0 \leq i \leq m$, and $r^{(0)}, r^{(1)}, \ldots, r^{(m)}$ is an orthogonal set of nonzero vectors.*

Proof We shall prove somewhat more; namely, that (under the given hypotheses) each of the following holds:

(a) $\langle r^{(m)}, v^{(i)}\rangle = 0$ $\quad (0 \leq i < m)$
(b) $\langle r^{(i)}, r^{(i)}\rangle = \langle r^{(i)}, v^{(i)}\rangle$ $\quad (0 \leq i \leq m)$
(c) $\langle v^{(m)}, Av^{(i)}\rangle = 0$ $\quad (0 \leq i < m)$
(d) $r^{(i)} = b - Ax^{(i)}$ $\quad (0 \leq i \leq m)$
(e) $\langle r^{(m)}, r^{(i)}\rangle = 0$ $\quad (0 \leq i < m)$
(f) $r^{(i)} \neq 0$ $\quad (0 \leq i \leq m)$

The proof is by induction on m. For the case $m = 0$, we assume that $v^{(0)} \neq 0$ and must prove parts (a)–(f). Notice that (a),(c), and (e) are vacuous in this case. As for (b), (d), and (f), they follow immediately from the definition of the algorithm, since

$$r^{(0)} = b - Ax^{(0)} = v^{(0)} \neq 0$$

Now assume that the theorem has been established for a certain m. On this basis, we shall prove it for $m+1$. To this end, we assume that $v^{(0)}, v^{(1)}, \ldots, v^{(m+1)}$ are all nonzero. By the induction hypothesis, parts (a)–(f) are valid. We want to prove that

(a′) $\langle r^{(m+1)}, v^{(i)} \rangle = 0$ $\quad (0 \leqq i \leqq m)$
(b′) $\langle r^{(m+1)}, r^{(m+1)} \rangle = \langle r^{(m+1)}, v^{(m+1)} \rangle$
(c′) $\langle v^{(m+1)}, Av^{(i)} \rangle = 0$ $\quad (0 \leqq i \leqq m)$
(d′) $r^{(m+1)} = b - Ax^{(m+1)}$
(e′) $\langle r^{(m+1)}, r^{(i)} \rangle = 0$ $\quad (0 \leqq i \leqq m)$
(f′) $r^{(m+1)} \neq 0$

For (a′), first let $i = m$. We have

$$\begin{aligned} \langle r^{(m+1)}, v^{(m)} \rangle &= \langle r^{(m)} - t_m A v^{(m)}, v^{(m)} \rangle = \langle r^{(m)}, v^{(m)} \rangle - t_m \langle v^{(m)}, A v^{(m)} \rangle \\ &= \langle r^{(m)}, v^{(m)} \rangle - \langle r^{(m)}, r^{(m)} \rangle = 0 \end{aligned}$$

using (b). If $0 \leqq i < m$, then by (a) and (c),

$$\langle r^{(m+1)}, v^{(i)} \rangle = \langle r^{(m)}, v^{(i)} \rangle - t_m \langle v^{(m)}, A v^{(i)} \rangle = 0$$

For the proof of (b′), we use part (a′) to conclude that

$$\langle r^{(m+1)}, v^{(m+1)} \rangle = \langle r^{(m+1)}, r^{(m+1)} + s_m v^{(m)} \rangle = \langle r^{(m+1)}, r^{(m+1)} \rangle$$

In proving part (c′) we set $s_{-1} = 0$ and $v^{(-1)} = 0$ whenever these terms appear. If $0 \leqq i \leqq m$, then

$$\begin{aligned} \langle v^{(m+1)}, Av^{(i)} \rangle &= \langle r^{(m+1)} + s_m v^{(m)}, Av^{(i)} \rangle \\ &= \langle r^{(m+1)}, Av^{(i)} \rangle + s_m \langle v^{(m)}, Av^{(i)} \rangle \rangle \\ &= t_i^{-1} \langle r^{(m+1)}, r^{(i)} - r^{(i+1)} \rangle + s_m \langle v^{(m)}, Av^{(i)} \rangle \\ &= t_i^{-1} \langle r^{(m+1)}, v^{(i)} - s_{i-1} v^{(i-1)} - v^{(i+1)} + s_i v^{(i)} \rangle \\ &\quad + s_m \langle v^{(m)}, Av^{(i)} \rangle \\ &= t_i^{-1} [\langle r^{(m+1)}, v^{(i)} \rangle - s_{i-1} \langle r^{(m+1)}, v^{(i-1)} \rangle - \langle r^{(m+1)}, v^{(i+1)} \rangle \\ &\quad + s_i \langle r^{(m+1)}, v^{(i)} \rangle] + s_m \langle v^{(m)}, Av^{(i)} \rangle \end{aligned}$$

If $i < m$, then by part (a′), $r^{(m+1)}$ is orthogonal to $v^{(i)}, v^{(i-1)}$, and $v^{(i+1)}$. Also, by part (c), $\langle v^{(m)}, Av^{(i)} \rangle = 0$. Hence in this case, $\langle v^{(m+1)}, Av^{(i)} \rangle = 0$. The case $i = m$ is special. By a previous equation, we have

$$\begin{aligned} \langle v^{(m+1)}, Av^{(m)} \rangle = & t_m^{-1} \langle r^{(m+1)}, v^{(m)} - s_{m-1} v^{(m-1)} - v^{(m+1)} + s_m v^{(m)} \rangle \\ & + s_m \langle v^{(m)}, Av^{(m)} \rangle \end{aligned}$$

By part (a'), $r^{(m+1)}$ is orthogonal to $v^{(m)}$ and to $v^{(m-1)}$. Hence

$$\begin{aligned}\langle v^{(m+1)}, Av^{(m)}\rangle &= -t_m^{-1}\langle r^{(m+1)}, v^{(m+1)}\rangle + s_m\langle r^{(m)}, Av^{(m)}\rangle \\ &= -t_m^{-1}\langle r^{(m+1)}, v^{(m+1)}\rangle + s_m\langle v^{(m)}, Av^{(m)}\rangle \\ &= -\frac{\langle v^{(m)}, Av^{(m)}\rangle}{\langle r^{(m)}, r^{(m)}\rangle}\langle r^{(m+1)}, v^{(m+1)}\rangle \\ &\quad + \frac{\langle r^{(m+1)}, r^{(m+1)}\rangle}{\langle r^{(m)}, r^{(m)}\rangle}\langle v^{(m)}, Av^{(m)}\rangle\end{aligned}$$

We see that this expression is zero by using (b').

For the proof of (d'), we write

$$\begin{aligned}b - Ax^{(m+1)} &= b - A(x^{(m)} + t_m v^{(m)}) = b - Ax^{(m)} - t_m Av^{(m)} \\ &= r^{(m)} - (r^{(m)} - r^{(m+1)}) = r^{(m+1)}\end{aligned}$$

For the proof of (e'), let $0 \leqq i \leqq m$, and put $s_{-1} = 0$ and $v^{(-1)} = 0$. Then from part (a') we have

$$\begin{aligned}\langle r^{(m+1)}, r^{(i)}\rangle &= \langle r^{(m+1)}, v^{(i)} - s_{i-1}v^{(i-1)}\rangle \\ &= \langle r^{(m+1)}, v^{(i)}\rangle - s_{i-1}\langle r^{(m+1)}, v^{(i-1)}\rangle = 0\end{aligned}$$

For part (f'), use (c') and the positive definiteness of A as follows:

$$\begin{aligned}0 < \langle v^{(m+1)}, Av^{(m+1)}\rangle &= \langle r^{(m+1)} + s_m v^{(m)}, Av^{(m+1)}\rangle \\ &= \langle r^{(m+1)}, Av^{(m+1)}\rangle + s_m\langle v^{(m)}, Av^{(m+1)}\rangle \\ &= \langle r^{(m+1)}, Av^{(m+1)}\rangle\end{aligned}$$

Hence, $r^{(m+1)} \neq 0$. ∎

This theorem and its proof are adapted from the book of Stoer and Bulirsch [1980].

Preconditioned Conjugate Gradient

We want to solve the system

$$Ax = b$$

where A is symmetric and positive definite, using a variant of the conjugate gradient method. It would be advantageous to *precondition* this system and obtain a new system that is *better conditioned* than the original system. By this we mean that for some nonsingular matrix S, the preconditioned system

$$\widehat{A}\widehat{x} = \widehat{b}$$

where

$$\begin{cases} \widehat{A} = S^T A S \\ \widehat{x} = S^{-1} x \\ \widehat{b} = S^T b \end{cases}$$

is such that $\kappa(\widehat{A}) < \kappa(A)$. As a by-product, the iterative method used to solve the preconditioned system may converge faster than it would for the original system. Rather than taking an arbitrary matrix S, we suppose that the symmetric and positive definite splitting matrix Q can be factored so that

$$Q^{-1} = SS^T$$

The reason for this will become clear shortly.

The formal conjugate gradient algorithm can be written for the preconditioned system as follows:

$\widehat{r}^{(0)} = \widehat{b} - \widehat{A}\widehat{x}^{(0)}$
$\widehat{v}^{(0)} = \widehat{r}^{(0)}$
for $k = 0, 1, \ldots, M$ **do**
  $\widehat{t}_k = \langle \widehat{r}^{(k)}, \widehat{r}^{(k)} \rangle / \langle \widehat{v}^{(k)}, \widehat{A}\widehat{v}^{(k)} \rangle$
  $\widehat{x}^{(k+1)} = \widehat{x}^{(k)} + \widehat{t}_k \widehat{v}^{(k)}$
  $\widehat{r}^{(k+1)} = \widehat{r}^{(k)} - \widehat{t}_k \widehat{A}\widehat{v}^{(k)}$
  $\widehat{s}_k = \langle \widehat{r}^{(k+1)}, \widehat{r}^{(k+1)} \rangle / \langle \widehat{r}^{(k)}, \widehat{r}^{(k)} \rangle$
  $\widehat{v}^{(k+1)} = \widehat{r}^{(k+1)} + \widehat{s}_k \widehat{v}^{(k)}$
end

In preconditioning the original system, any sparsity in the matrix A can be destroyed in forming $\widehat{A}$. So rather than explicitly forming the preconditioned system, we would like to use the original system and have the preconditioning done implicitly within the algorithm.

We write

$$\widehat{x}^{(k)} = S^{-1} x^{(k)}$$
$$\widehat{v}^{(k)} = S^{-1} v^{(k)}$$
$$\widehat{r}^{(k)} = \widehat{b} - \widehat{A}\widehat{x}^{(k)} = S^T b - \langle S^T A S \rangle \langle S^{-1} x^{(0)} \rangle = S^T r^{(k)}$$
$$\widetilde{r}^{(k)} = Q^{-1} r^{(k)}$$

Then

$$\begin{aligned} \widehat{t}_k &= \langle \widehat{r}^{(k)}, \widehat{r}^{(k)} \rangle / \langle \widehat{v}^{(k)}, \widehat{A}\widehat{v}^{(k)} \rangle \\ &= \langle S^T r^{(k)}, S^T r^{(k)} \rangle / \langle S^{-1} v^{(k)}, (S^T A S)(S^{-1} v^{(k)}) \rangle \\ &= \langle Q^{-1} r^{(k)}, r^{(k)} \rangle / \langle v^{(k)}, A v^{(k)} \rangle \\ &= \langle \widetilde{r}^{(k)}, r^{(k)} \rangle / \langle v^{(k)}, A v^{(k)} \rangle \end{aligned}$$

Moreover,

$$\widehat{x}^{(k+1)} = \widehat{x}^{(k)} + \widehat{t}_k \widehat{v}^{(k)}$$
$$S^{-1}x^{(k+1)} = S^{-1}x^{(k)} + \widehat{t}_k S^{-1}v^{(k)}$$

and multiplying by S, we have

$$x^{(k+1)} = x^{(k)} + \widehat{t}_k v^{(k)}$$

Similarly,

$$\widehat{r}^{(k+1)} = \widehat{r}^{(k)} - \widehat{t}_k \widehat{A}\widehat{v}^{(k)}$$
$$S^T r^{(k+1)} = S^T r^{(k)} - \widehat{t}_k (S^T A S)(S^{-1}v^{(k)})$$

and multiplying by S^{-T}, we have

$$r^{(k+1)} = r^{(k)} - \widehat{t}_k A v^{(k)}$$

Now

$$\begin{aligned}\widehat{s}_k &= \langle \widehat{r}^{(k+1)}, \widehat{r}^{(k+1)} \rangle / \langle \widehat{r}^{(k)}, \widehat{r}^{(k)} \rangle \\ &= \langle S^T r^{(k+1)}, S^T r^{(k+1)} \rangle / \langle S^T r^{(k)}, S^T r^{(k)} \rangle \\ &= \langle Q^{-1} r^{(k+1)}, r^{(k+1)} \rangle / \langle Q^{-1} r^{(k)}, r^{(k)} \rangle \\ &= \langle \widetilde{r}^{(k+1)}, r^{(k+1)} \rangle / \langle \widetilde{r}^{(k)}, r^{(k)} \rangle\end{aligned}$$

Also,

$$\widehat{v}^{(k+1)} = \widehat{r}^{(k+1)} + \widehat{s}_k \widehat{v}^{(k)}$$
$$S^{-1}v^{(k+1)} = S^T r^{(k+1)} + \widehat{s}_k S^{-1}v^{(k)}$$

and multiplying by S, we have

$$\begin{aligned}v^{(k+1)} &= Q^{-1}r^{(k+1)} + \widehat{s}_k v^{(k)} \\ &= \widetilde{r}^{(k+1)} + \widehat{s}_k v^{(k)}\end{aligned}$$

Now we can write the **preconditioned conjugate gradient** algorithm primarily in terms of the original system. The pseudocode given next is not suitable for a computer program. Another, more efficient, version is given later in this section.

input $x^{(0)}, A, b, M$
$r^{(0)} \leftarrow b - Ax^{(0)}$
solve $Q\tilde{r}^{(0)} = r^{(0)}$ for $\tilde{r}^{(0)}$
$v^{(0)} \leftarrow r^{(0)}$
output $0, x^{(0)}$
for $k = 0, 1, 2, \ldots, M-1$ **do**
  if $v^{(k)} = 0$ **then stop**
  $\widehat{t}_k \leftarrow \langle \tilde{r}^{(k)}, r^{(k)} \rangle / \langle v^{(k)}, Av^{(k)} \rangle$
  $x^{(k+1)} \leftarrow x^{(k)} + \widehat{t}_k v^{(k)}$
  $r^{(k+1)} \leftarrow r^{(k)} - \widehat{t}_k Av^{(k)}$
  solve $Q\tilde{r}^{(k+1)} = r^{(k+1)}$ for $\tilde{r}^{(k+1)}$
  if $\langle \tilde{r}^{(k+1)}, r^{(k+1)} \rangle < \varepsilon$ **then**
    if $\|r^{(k+1)}\|_2^2 < \varepsilon$ **then stop**
  endif
  $\widehat{s}_k \leftarrow \langle \tilde{r}^{(k+1)}, r^{(k+1)} \rangle / \langle \tilde{r}^{(k)}, r^{(k)} \rangle$
  $v^{(k+1)} \leftarrow \tilde{r}^{(k+1)} + \widehat{s}_k v^{(k)}$
  output $k+1, x^{(k+1)}, r^{(k+1)}$
end

The preconditioned conjugate gradient algorithm above reduces to the regular conjugate gradient algorithm when $Q^{-1} = I$, since $\tilde{r}^{(k)} = r^{(k)}$, $\widehat{t}_k = t_k$, and $\widehat{s}_k = s_k$.

If $Q^{-1} = A$ and $S = A^{1/2}$, then the preconditioned system would reduce to $\widehat{x} = \widehat{b}$ and would be trivially solved. Unfortunately, this *perfectly* conditioned system ($\kappa(\widehat{A}) = 1$) is of no computational value, since determining $\widehat{b} = S^T b$ would be as difficult as solving the original system.

Since we must solve a system of the form $Qx = y$ on each iteration of the preconditioned conjugate gradient algorithm, Q must be selected so that this system is *easy* to solve. If Q is a diagonal matrix, then this condition is met, but more complicated preconditioners may lead to faster convergence. As Q^{-1} becomes a better approximation to A, the preconditioned system becomes better conditioned, and the convergence of the iterative procedure occurs in fewer steps. On the other hand, solving $Qx = y$ becomes more difficult; this illustrates the typical trade-off between iterative methods requiring a few expensive steps and those requiring many inexpensive steps.

What about the convergence test? Since $\|r^{(k+1)}\|_2^2$ is not available in this algorithm, this requires an additional computation. What *is* available is $\langle \tilde{r}^{(k+1)}, r^{(k+1)} \rangle$. Using it may cause the iterative process to halt either before or after it should for the specified accuracy. Hence, an additional evaluation of $\|r^{(k+1)}\|_2^2$ should be made as a check, once the previous test indicates convergence.

A computer code can be based on the following **preconditioned conjugate gradient** algorithm:

```
input x, A, b, M, δ, ε
r ← b − Ax
solve Qz = r for z
v ← z
c ← ⟨z, r⟩
for k = 1, 2, ..., M do
    if ⟨v, v⟩^{1/2} < δ then stop
    z ← Av
    t ← c/⟨v, z⟩
    x ← x + tv
    r ← r − tz
    solve Qz = r for z
    d ← ⟨z, r⟩
    if d^2 < ε then
        e ← ⟨r, r⟩
        if e^2 < ε then stop
    endif
    v ← z + (d/c)v
    c ← d
    output k, x, r
end
```

The preconditioned conjugate gradient method is an area of active research. There are a number of choices for Q, such as the symmetric successive overrelaxation matrix and many others. Computer packages are available containing conjugate gradient routines and other iterative methods. Examples of such packages are `ITPACKV 2D` by Kincaid, Oppe, and Young [1989], `NSPCG` by Oppe, Joubert, and Kincaid [1988], or `PCGPAK` [1984]. We have patterned the presentation here for the preconditioned conjugate gradient method after that of Ortega [1988]. Additional details can be found there or in many other books, such as Golub and van Loan [1989].

PROBLEM SET *4.7

1. Prove that if A is symmetric, then the gradient of the function

$$q(x) = \langle x, Ax \rangle - 2\langle x, b \rangle$$

at x is $2(Ax - b)$. Recall that the gradient of a function $g : \mathbb{R}^n \to \mathbb{R}$ is the vector whose components are $\partial g / \partial x_i$, for $i = 1, 2, \ldots, n$.

2. Prove that the minimum value of $q(x)$ is $-\langle b, A^{-1}b \rangle$.
3. Prove that in the method of steepest descent, $v^{(k)} \perp v^{(k+1)}$.

4. Let A be positive definite, and let b be a fixed vector. For any x, the *residual vector* is $r = b - Ax$, and the *error vector* is $e = A^{-1}b - x$. Show that the inner-product of the error vector with the residual vector is positive unless $Ax = b$.
5. Let A be symmetric, let $Ax = b$, and let y be any vector. Using q as defined in the text, prove that
$$\langle (x-y), A(x-y)\rangle = \langle b, A^{-1}b\rangle + q(y)$$
This shows that the minimization of $q(y)$ is equivalent to the minimization of $\langle (x-y), A(x-y)\rangle$.
6. Prove that if $\widehat{t}$ is defined by Equation (3) and if $y = x + \widehat{t}v$, then $v \perp (b - Ay)$; that is, $(v, b - Ay) = 0$.
7. Show that in the method of steepest descent,
$$q(x^{(k+1)}) = q(x^{(k)}) - \|r^{(k)}\|^4 / \langle r^{(k)}, Ar^{(k)}\rangle$$
where $r^{(k)} = b - Ax^{(k)}$.
8. If $\{u^{(1)}, u^{(2)}, \ldots, u^{(n)}\}$ is a set of vectors that is orthonormal in the usual sense, and if these are used as the directions of search to minimize q, will the solution be obtained after n steps?
9. Let A be an $n \times n$ matrix, not assumed to be symmetric or positive definite. Assume the existence of an A-orthonormal system $\{u^{(1)}, u^{(2)}, \ldots, u^{(n)}\}$ and prove that A is symmetric and positive definite.
10. Prove that in the conjugate gradient method,
$$t_k = \langle r^{(k)}, v^{(k)}\rangle / \langle v^{(k)}, Av^{(k)}\rangle$$
$$s_k = \langle r^{(k+1)}, Av^{(k)}\rangle / \langle v^{(k)}, Av^{(k)}\rangle$$
[c]11. Program and test the conjugate gradient method. A good test case is the Hilbert matrix with a simple b-vector:
$$a_{ij} = (i + j + 1)^{-1} \qquad b_i = \frac{1}{3}\sum_{j=1}^{n} a_{ij} \qquad (1 \leqq i, j \leqq n)$$
12. In the conjugate gradient method, prove that if $v^{(k)} = 0$ then $Ax^{(k)} = b$.
13. For what values of t_k does the method of steepest descent reduce to the Richardson method? What conditions on t_k and A must be fulfilled in order that the method of steepest descent be equivalent to the Jacobi method?
14. Consider the linear system
$$\begin{bmatrix} 2 & 0 & -1 \\ -2 & -10 & 0 \\ -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -12 \\ 2 \end{bmatrix}$$
Using the starting vector $x^{(0)} = (0, 0, 0)^T$, carry out two iterations of each of the following methods: (a) Jacobi (b) Gauss-Seidel (c) conjugate gradient
[c]15. Solve the following system using the indicated method starting with $x^{(0)} = 0$ (a) Jacobi (b) Gauss-Seidel (c) conjugate gradient
$$\begin{bmatrix} 10 & 1 & 2 & 3 & 4 \\ 1 & 9 & -1 & 2 & -3 \\ 2 & -1 & 7 & 3 & -5 \\ 3 & 2 & 3 & 12 & -1 \\ 4 & -3 & -5 & -1 & 15 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 12 \\ -27 \\ 14 \\ -17 \\ 12 \end{bmatrix}$$
Remark: Here the coefficient matrix is symmetric and positive definite but not diagonally dominant.

*4.8 Analysis of Roundoff Error in the Gaussian Algorithm

In this section, we investigate the roundoff errors that inevitably arise in solving a system of linear equations:

$$Ax = b \qquad (A \in \mathbb{R}^{n\times n},\ x \in \mathbb{R}^n,\ b \in \mathbb{R}^n)$$

An analysis, due originally to Wilkinson, is given for the Gaussian algorithm using unscaled row pivoting. The results are in the form of *a posteriori* bounds on the error. Thus, at the *conclusion* of the computation, an assertion can be made about the magnitude of the error.

In the analysis, we assume that suitable row interchanges have been carried out at the beginning so that the correct pivot elements will always be situated in the desired position. The row pivoting strategy then assures us that in step k of the algorithm, $|a_{kk}^{(k)}| \geqq |a_{ik}^{(k)}|$ for $i \geqq k$. (For the notation, refer to Section 4.3.) The inequality just given implies that the multipliers needed in the elimination process will not exceed unity in magnitude.

The formulæ that define the LU-decomposition are as follows:

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} & i \leqq k \\ a_{ij}^{(k)} - \ell_{ik}a_{kj}^{(k)} & i > k \text{ and } j > k \\ 0 & j \leqq k < i \end{cases}$$

$$\ell_{ik} = \begin{cases} 0 & i < k \\ 1 & i = k \\ a_{ik}^{(k)}/a_{kk}^{(k)} & i > k \end{cases}$$

The process begins with $A^{(1)} = A$ and ends with $A^{(n)} = U$. The matrices L and U are unit lower triangular and upper triangular, respectively.

The numbers that actually occur in the computer will be distinguished notationally with a tilde. Thus, for example, $\tilde{\ell}_{ik}$ is the number computed and stored in the memory location assigned to ℓ_{ik}. The meaning of the tilde is therefore machine-dependent. We shall assume that no overflow or underflow occurs in the course of the computations.

In describing the actual calculations in the computer, we use the function fl defined in Section 2.1. Recall that if x and y are machine numbers and if $\odot$ denotes an arithmetic operation, then the computer will produce, instead of $x \odot y$, a machine number denoted by $fl(x \odot y)$. By Equation (8) and by Problem 8 in Section 2.1, the relation between $x \odot y$ and $fl(x \odot y)$ can be expressed by the equation

$$fl(x \odot y) = (x \odot y)(1 - \delta) = (x \odot y)/(1 - \delta') \tag{1}$$

in which δ and δ' are numbers whose magnitudes do not exceed the unit roundoff, ε, in the particular computer being considered.

When these matters are taken into account, the computed numbers in the Gaussian

algorithm can be described by the equations

$$\tilde{a}_{ij}^{(k+1)} = \begin{cases} \tilde{a}_{ij}^{(k)} & i \leqq k \\ fl\left[\tilde{a}_{ij}^{(k)} - fl(\tilde{\ell}_{ik}\tilde{a}_{kj}^{(k)})\right] & i > k\,,\ j > k \end{cases}$$

$$\tilde{\ell}_{ik} = \begin{cases} 1 & i = k \\ fl\left(\tilde{a}_{ik}^{(k)}/\tilde{a}_{kk}^{(k)}\right) & i > k \end{cases}$$

THEOREM 1 *Let A be an $n \times n$ nonsingular matrix whose elements are machine numbers in a computer with unit roundoff ε. The Gaussian algorithm with row pivoting produces matrices $\tilde{L}$ and $\tilde{U}$ such that*

$$\tilde{L}\tilde{U} = A + E \qquad \textit{where} \qquad |e_{ij}| \leqq 2n\varepsilon \max_{1 \leqq i,j,k \leqq n} |a_{ij}^{(k)}|$$

Proof Introduce quantities δ_{ij} and δ'_{ij} to play the rôles of $\pm\delta$ or $\pm\delta'$ in Equation (1). The computed numbers in the algorithm then obey these equations:

$$\tilde{a}_{ij}^{(k+1)} = \left[\tilde{a}_{ij}^{(k)} - (1-\delta_{ij})\tilde{\ell}_{ik}\tilde{a}_{kj}^{(k)}\right]/(1-\delta'_{ij}) \qquad (i > k\,,\ j > k)$$

$$\tilde{\ell}_{ik} = (1+\delta_{ik})\tilde{a}_{ik}^{(k)}/\tilde{a}_{kk}^{(k)} \qquad (i > k)$$

The first of these two equations can also be written in the form

$$\tilde{a}_{ij}^{(k+1)} = \tilde{a}_{ij}^{(k)} - \tilde{\ell}_{ik}\tilde{a}_{kj}^{(k)} + \delta_{ij}\tilde{\ell}_{ik}\tilde{a}_{kj}^{(k)} + \delta'_{ij}\tilde{a}_{ij}^{(k+1)} \qquad (i > k\,,\ j > k)$$

Now introduce an $n \times n$ matrix $\tilde{L}^{(k)}$ that consists entirely of zeros except in column k below the diagonal:

$$\tilde{L}^{(k)} = \begin{bmatrix} 0 & & & & \\ & \ddots & & & \\ & & 0 & & \\ & & \tilde{\ell}_{k+1,k} & & \\ & & \vdots & \ddots & \\ & & \tilde{\ell}_{n,k} & & 0 \end{bmatrix}$$

Then $\tilde{A}^{(k)} - \tilde{L}^{(k)}\tilde{A}^{(k)}$ is the result of applying certain elementary row operations to $\tilde{A}^{(k)}$. Namely, $\tilde{\ell}_{ik}$ times row k is subtracted from row i, for $k+1 \leqq i \leqq n$. This is almost the definition of $\tilde{A}^{(k+1)}$. The difference arises from the roundoff and from the fact that in $\tilde{A}^{(k+1)}$ we set $\tilde{a}_{ik}^{(k+1)} = 0$ for $k+1 \leqq i \leqq n$. It is therefore possible to write

$$\tilde{A}^{(k+1)} = \tilde{A}^{(k)} - \tilde{L}^{(k)}\tilde{A}^{(k)} + E^{(k)} \tag{2}$$

where $E^{(k)}$ is a matrix that compensates for the errors.

To analyze the matrix $E^{(k)}$, consider first the case $i > k$, $j = k$. From previous equations, we have

$$\begin{aligned} e_{ik}^{(k)} &= \tilde{a}_{ik}^{(k+1)} - a_{ik}^{(k)} + \tilde{\ell}_{ik}^{(k)}\tilde{a}_{kk}^{(k)} \\ &= 0 - \tilde{a}_{ik}^{(k)} + \tilde{\ell}_{ik}\tilde{a}_{kk}^{(k)} \\ &= -\tilde{a}_{ik}^{(k)} + (\tilde{a}_{ik}^{(k)}/\tilde{a}_{kk}^{(k)})(1+\delta_{ik})\tilde{a}_{kk}^{(k)} \\ &= \delta_{ik}\tilde{a}_{ik}^{(k)} \qquad (i > k) \end{aligned}$$

Next, consider the case $i > k$, $j > k$. Here we obtain

$$\begin{aligned} e_{ij}^{(k)} &= \tilde{a}_{ij}^{(k+1)} - \tilde{a}_{ij}^{(k)} + \tilde{\ell}_{ik}^{(k)}\tilde{a}_{kj}^{(k)} \\ &= \delta_{ij}\tilde{\ell}_{ik}\tilde{a}_{kj}^{(k)} + \delta'_{ij}\tilde{a}_{ij}^{(k+1)} \qquad (i > k\,,\ j > k) \end{aligned}$$

All other elements of $E^{(k)}$ are zero. Now add Equations (2) for $k = 1, 2, \ldots, n-1$. The result can be written

$$\tilde{L}^{(1)}\tilde{A}^{(1)} + \cdots + \tilde{L}^{(n-1)}\tilde{A}^{(n-1)} + I\tilde{A}^{(n)} = A^{(1)} + E^{(1)} + \cdots + E^{(n-1)}$$

Notice that $\tilde{L}^{(k)}\tilde{A}^{(k)}$ is a matrix whose rows are simply various multiples of the single row vector

$$[\tilde{a}_{k1}^{(k)}, \tilde{a}_{k2}^{(k)}, \ldots, \tilde{a}_{kn}^{(k)}]$$

Since this row vector is the same as

$$[\tilde{a}_{k1}^{(n)}, \tilde{a}_{k2}^{(n)}, \ldots, \tilde{a}_{kn}^{(n)}]$$

we conclude that $\tilde{L}^{(k)}\tilde{A}^{(k)} = \tilde{L}^{(k)}\tilde{A}^{(n)}$. Recall also that $A^{(1)} = A$ and that the computed upper triangular factor U is $\tilde{U} = \tilde{A}^{(n)}$. Putting $E = \sum_{k=1}^{n-1} E^{(k)}$, we have

$$\left(\tilde{L}^{(1)} + \tilde{L}^{(2)} + \cdots + \tilde{L}^{(n-1)} + I\right)\tilde{A}^{(n)} = A^{(1)} + E$$

or

$$\tilde{L}\tilde{U} = A + E$$

All that remains now is to produce a bound on $\|E\|$. Let $\rho = \max_{1 \leqq i,j,k \leqq n} |A_{ij}^{(k)}|$. The preliminary row interchanges have ensured that all the multipliers satisfy $|\tilde{\ell}_{ik}| \leqq 1$. Thus, from equations given previously for $e_{ij}^{(k)}$, we have

$$\begin{aligned} |e_{ik}^{(k)}| &= |\delta_{ik}|\,|\tilde{a}_{ik}^{(k)}| \leqq \varepsilon\rho \qquad (i > k) \\ |e_{ij}^{(k)}| &= |\delta_{ij}\tilde{\ell}_{ik}\tilde{a}_{kj}^{(k)} + \delta'_{ij}\tilde{a}_{ij}^{(k+1)}| \leqq 2\varepsilon\rho \qquad (i > k\,,\ j > k) \end{aligned}$$

From the definition of E it follows that

$$|e_{ij}| = \left|\sum_{k=1}^{n-1} e_{ij}^{(k)}\right| \leqq \sum_{k=1}^{n-1} |e_{ij}^{(k)}| \leqq 2n\varepsilon\rho$$

∎

To apply this theorem in practice, we must first know the number

$$\rho = \max_{1 \leqq i,j \leqq n} |a_{ij}^{(k)}|$$

This can be computed during the progress of the algorithm by adding suitable code.

THEOREM 2 *If $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ are machine numbers, then the machine value of*

$$\sum_{i=1}^{n} x_i y_i$$

computed in the natural way can be expressed as $\sum_{i=1}^{n} x_i y_i (1+\delta_i)$, in which the δ_i's satisfy $|\delta_i| \leqq \frac{6}{5}(n+1)\varepsilon$. (The number ε is the unit roundoff error of the machine, and we assume that $n\varepsilon < \frac{1}{3}$.)

Proof The hypothesis means that the computation will proceed as follows:

$$z_0 = 0 \qquad z_k = fl\big[z_{k-1} + fl(x_k y_k)\big] \qquad (1 \leqq k \leqq n)$$

Let us use induction on n to prove that $|\delta_i| \leqq (1+\varepsilon)^{n+2-i} - 1$. For $n = 1$, we have

$$z_1 = fl(x_1 y_1) = x_1 y_1 (1 + \delta_1) \qquad |\delta_1| \leqq \varepsilon$$

In this case, the assertion being proved is that $|\delta_1| \leqq (1+\varepsilon)^2 - 1$, and this is true since $\varepsilon \leqq (1+\varepsilon)^2 - 1$. If the theorem is true for $n = k-1$, then its proof for $n = k$ goes like this:

$$\begin{aligned} z_k &= \big[z_{k-1} + x_k y_k (1+\delta')\big](1+\delta) \\ &= \left[\sum_{i=1}^{k-1} x_i y_i (1+\delta_i) + x_k y_k (1+\delta')\right](1+\delta) \\ &= \sum_{i=1}^{k-1} x_i y_i (1 + \delta_i + \delta + \delta_i \delta) + x_k y_k (1 + \delta + \delta' + \delta\delta') \end{aligned}$$

In the preceding equation, we have these bounds:

$$|\delta| \leqq \varepsilon \qquad |\delta'| \leqq \varepsilon \qquad |\delta_i| \leqq (1+\varepsilon)^{k+1-i} - 1 \qquad (1 \leqq i \leqq k-1)$$

We need to prove that

$$|\delta_i + \delta + \delta_i \delta| \leqq (1+\varepsilon)^{k+2-i} - 1 \qquad |\delta + \delta' + \delta\delta'| \leqq (1+\varepsilon)^2 - 1$$

The first of these inequalities is established by writing

$$\begin{aligned} |\delta_i + \delta + \delta_i \delta| &\leqq |\delta_i| + |\delta|(1 + |\delta_i|) \\ &\leqq (1+\varepsilon)^{k+1-i} - 1 + \varepsilon(1+\varepsilon)^{k+1-i} \\ &= (1+\varepsilon)^{k+2-i} - 1 \end{aligned}$$

The second inequality is proved in a similar manner. This completes the induction. The following estimate is now required for $k \leqq n+1$:

$$\begin{aligned}(1+\varepsilon)^k - 1 &= \left[1 + k\varepsilon + \frac{k(k-1)}{2}\varepsilon^2 + \cdots + \varepsilon^k\right] - 1 \\ &= k\varepsilon\left[1 + \frac{k-1}{2}\varepsilon + \frac{(k-1)(k-2)}{2\cdot 3}\varepsilon^2 + \cdots\right] \\ &\leqq k\varepsilon\left[1 + \frac{k\varepsilon}{2} + \left(\frac{k\varepsilon}{2}\right)^2 + \cdots\right] \\ &= k\varepsilon/(1 - \tfrac{1}{2}k\varepsilon) < \tfrac{6}{5}k\varepsilon\end{aligned}$$

In the last step, the assumption $k\varepsilon \leqq n\varepsilon < \frac{1}{3}$ was used. Finally,

$$|\delta_i| \leqq (1+\varepsilon)^{n+2-i} - 1 \leqq \tfrac{6}{5}(n+2-i)\varepsilon \leqq \tfrac{6}{5}(n+1)\varepsilon$$ ∎

THEOREM 3 *Let L be an $n \times n$ unit lower triangular matrix whose elements are machine numbers. Let b be a vector whose components are machine numbers. The computed solution of $Ly = b$ is a vector $\tilde{y}$ that is the exact solution of*

$$(L + \Delta)\tilde{y} = b \qquad \textit{with} \qquad |\Delta_{ij}| \leqq \tfrac{6}{5}(n+1)\varepsilon|\ell_{ij}| \tag{3}$$

Here ε is the machine's unit roundoff error, and it is assumed that $n\varepsilon < \frac{1}{3}$.

Proof The exact solution of the system $Ly = b$ is computed from the formula

$$y_i = b_i - \sum_{j=1}^{i-1} \ell_{ij} y_j \qquad (1 \leqq i \leqq n)$$

The computed values $\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n$ will satisfy an equation of the form

$$\tilde{y}_i = \left[b_i - \sum_{j=1}^{i-1} \ell_{ij}\tilde{y}_j(1+\delta_{ij})\right] \Big/ (1+\delta_{ii}) \tag{4}$$

in which the δ_{ij} are numbers satisfying

$$|\delta_{ij}| \leqq \tfrac{6}{5}(n+1)\varepsilon \qquad (1 \leqq j \leqq i \leqq n) \tag{5}$$

This assertion is a consequence of the assumptions made in Section 2.1 and of the preceding theorem. Equation (4) can be rearranged like this:

$$(1+\delta_{ii})\tilde{y}_i = b_i - \sum_{j=1}^{i-1} \ell_{ij}\tilde{y}_j(1+\delta_{ij})$$

and then rewritten like this:

$$\sum_{j=1}^{i} \ell_{ij}\tilde{y}_j(1+\delta_{ij}) = b_i$$

We interpret this as a matrix equation

$$(L + \Delta)\widetilde{y} = b \tag{6}$$

with Δ the lower triangular matrix whose elements are $\ell_{ij}\delta_{ij}$ whenever $1 \leqq j \leqq i \leqq n$. Thus

$$|\Delta_{ij}| = |\ell_{ij}|\,|\delta_{ij}| \leqq \tfrac{6}{5}(n+1)\varepsilon|\ell_{ij}| \qquad \blacksquare$$

THEOREM 4 *Let U be an $n \times n$, upper triangular, nonsingular matrix. If the elements of U and c are machine numbers, and if $n\varepsilon < \frac{1}{3}$, then the computed solution $\widetilde{y}$ of $Uy = c$ satisfies exactly a perturbed system*

$$(U + \Delta)\widetilde{y} = c \qquad \textit{with} \qquad |\Delta_{ij}| \leqq \tfrac{6}{5}(n+1)\varepsilon|u_{ij}|$$

Proof This is left as a problem. $\blacksquare$

THEOREM 5 *Let the elements of A and b be machine numbers. If the Gaussian algorithm with row pivoting is used to solve $Ax = b$, the computed solution $\widetilde{x}$ is the exact solution of a perturbed system*

$$(A + F)\widetilde{x} = b \qquad \textit{in which} \qquad |f_{ij}| \leqq 10n^2\varepsilon\rho$$

Here n is the order of the matrix A, $\rho = \max_{1 \leqq i,j,k \leqq n} |a_{ij}^{(k)}|$, and ε is the machine's unit roundoff error. It is assumed that $n\varepsilon < \frac{1}{3}$.

Proof By Theorems 1, 3, and 4, since $n\varepsilon < \frac{1}{3}$, we have

$$\begin{aligned} A + E &= \widetilde{L}\widetilde{U} & |e_{ij}| &\leqq 2n\varepsilon\rho \\ (\widetilde{L} + \Delta)\widetilde{y} &= b & |\Delta_{ij}| &\leqq \tfrac{6}{5}(n+1)\varepsilon|\widetilde{\ell}_{ij}| \\ (\widetilde{U} + \Delta')\widetilde{x} &= \widetilde{y} & |\Delta'_{ij}| &\leqq \tfrac{6}{5}(n+1)\varepsilon|\widetilde{u}_{ij}| \end{aligned}$$

Putting these equations together yields

$$\begin{aligned} b &= (\widetilde{L} + \Delta)\widetilde{y} = (\widetilde{L} + \Delta)(\widetilde{U} + \Delta')\widetilde{x} = (\widetilde{L}\widetilde{U} + \Delta\widetilde{U} + \widetilde{L}\Delta' + \Delta\Delta')\widetilde{x} \\ &= (A + E + \Delta\widetilde{U} + \widetilde{L}\Delta' + \Delta\Delta')\widetilde{x} = (A + F)\widetilde{x} \end{aligned}$$

where $F = E + \Delta\widetilde{U} + \widetilde{L}\Delta' + \Delta\Delta'$. To obtain an estimate of F, we employ the bounds given above, together with the observations that $|\ell_{ij}| \leqq 1$ and

$$|u_{ij}| = |a_{ij}^{(n)}| \leqq \rho$$

Thus

$$\begin{aligned} |f_{ij}| &\leqq |e_{ij}| + \sum_{\nu=1}^{n} \{|\Delta_{i\nu}|\,|\widetilde{u}_{\nu j}| + |\widetilde{\ell}_{i\nu}|\,|\Delta'_{\nu j}| + |\Delta_{i\nu}|\,|\Delta'_{\nu j}|\} \\ &\leqq 2n\varepsilon\rho + \frac{6}{5}n(n+1)\varepsilon\rho + \frac{6}{5}n(n+1)\varepsilon\rho + \frac{36}{25}n(n+1)^2\varepsilon^2\rho \\ &= n^2\varepsilon\rho\left\{\frac{2}{n} + \frac{12}{5}\,\frac{n+1}{n} + \frac{36}{25}\varepsilon\left(\frac{n+1}{n}\right)^2\right\} \end{aligned}$$

The quantity within the braces cannot exceed 10. ∎

The bound given in Theorem 5 can be improved by paying more attention to details in Theorems 1–5 and by computing $\|F\|_\infty$ instead. The reader should consult Forsythe and Moler [1967], Golub and Van Loan [1989], Wilkinson [1965], and Isaacson and Keller [1966].

PROBLEM SET *4.8

1. Prove Theorem 4.
2. In Theorem 1, prove the inequalities

$$\|E^{(k)}\|_\infty \leqq [1 + 2(n-k)]\,\varepsilon\rho$$
$$\|E\|_\infty \leqq n^2\varepsilon\rho$$

3. In Theorem 3, prove that

$$\|\Delta\|_\infty \leqq \tfrac{3}{5}n(n+1)\varepsilon \max_{1\leqq i,j\leqq n} |\ell_{ij}|$$