In Theorem 1.6.1 we stated a necessary and sufficient condition for the equation

\[ Ax = b \quad (1) \]

to have a solution. As we noted then, this theorem and its proof do not suggest how a solution may be computed. In this chapter we shall be concerned with deriving algorithms for solving (1). The algorithms have in common the idea of factoring \( A \) in the form

\[ A = R_1 R_2 \cdots R_k, \]

where each matrix \( R_i \) is so simple that the equation \( Ry = c \) can be easily solved. The equation \( Ax = b \) can then be solved by taking \( y_0 = b \) and for \( i = 1, 2, \ldots, k \) computing \( y_i \) as the solution of the equation

\[ R_i y_i = y_{i-1}. \]

Obviously \( y_k \) is the desired solution \( x \).
The matrices $R_i$ arising in the factorization of $A$ are often triangular. Hence Section 1 will be devoted to algorithms for solving triangular systems. Sections 2 and 3 will be devoted to algorithms for factoring $A$, and in Section 4 we will apply these factorizations to the solution of linear systems. The last section of this chapter will be devoted to a discussion of the numerical properties of the algorithms.

When $A$ is nonsingular, the methods of this chapter can be adapted to give algorithms for computing $A^{-1}$. However, we shall be at pains to stress that in most applications the computation of $A^{-1}$ is both unnecessary and inordinately time consuming. For example, if it is desired to compute a solution of (1), the methods of this chapter will be cheaper than first computing $A^{-1}$ and then forming $A^{-1}b$.

1. TRIANGULAR MATRICES AND SYSTEMS

In this section we shall give algorithms for inverting triangular matrices and solving triangular systems of equations. There are several reasons for considering this special case first. In the first place, our subsequent algorithms for solving general linear systems will presuppose the ability to solve triangular systems. Second, the algorithms for triangular systems are comparatively simple. Third, the algorithms furnish good examples in a simple setting of some of the practical considerations concerning storage and operation counts that were discussed in Section 2.3. We shall restrict our attention to upper triangular matrices, the modifications for lower triangular matrices being obvious.

The first problem is to determine when $T$ is nonsingular. We begin our investigations with the following formula for inverting a block triangular matrix.

**THEOREM 1.1.** Let $A$ and $C$ be of order $l$ and $m$, respectively, and $B$ be an $l \times m$ matrix. Then the matrix

$$T = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

is nonsingular if and only if $A$ and $C$ are nonsingular. In this case

$$T^{-1} = \begin{pmatrix} A & B \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}BC^{-1} \\ 0 & C^{-1} \end{pmatrix}.$$  \hspace{1cm} (1.1)
PROOF. If $A$ and $C$ are nonsingular, the matrix on the right-hand side of (1.1) is well defined, and it is easily verified that it is an inverse for $T$. Conversely suppose that, say, $C$ is singular. Then there is a nonzero vector $x$ such that $x^TC = 0$. If $0$ denotes the zero row vector with $l$ components, then $(0, x^T) \neq 0$ and

$$(0, x^T)T = (0, x^T) \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

$$=(0A + x^T0, 0B + x^TC) = 0.$$

Hence $T$ is singular. The case where $A$ is singular is treated similarly. ■

Theorem 1.1 has a rather useful theoretical consequence. Note that the matrix $A$ is the leading principal submatrix of order $l$ of $T$; that is $A = T^{(l)}$ (see the discussion following Definition 1.3.1). From (1.1), the matrix $A^{-1}$ is the leading principal submatrix of order $l$ of $T^{-1}$. In other words, if $T$ is upper triangular, then $(T^{(l)})^{-1} = (T^{-1})^{(l)}$. A similar result is true of the trailing submatrices.

Theorem 1.1 allows us to describe the inverse of a triangular matrix.

**THEOREM 1.2.** Let $T$ be upper triangular. Then $T$ is nonsingular if and only if its diagonal elements are nonzero. In this case $T^{-1}$ is upper triangular.

**PROOF.** The proof is by induction on the order of $T$. The assertion is obviously true for matrices of order 1. Suppose the theorem is true of all triangular matrices of order less than $n > 1$ and $T$ is of order $n$. Then $T$ can be partitioned in the form

$$T = \begin{pmatrix} T_{n-1} & t_n \\ 0 & \tau_{nn} \end{pmatrix},$$

where $T_{n-1}$ is the leading principal submatrix of order $n - 1$ of $T$ and $t_n = (\tau_{1n}, \tau_{2n}, \ldots, \tau_{n-1,n})^T$. Now $T_{n-1}$ and $\tau_{nn}$ are upper triangular matrices of order less than $n$. They have no nonzero diagonal entries if and only if $T$ has no nonzero diagonal elements. By the induction hypothesis they are nonsingular if and only if they have no nonzero diagonal elements. Finally by Theorem 1.1, $T$ is nonsingular if and only if $T_{n-1}$ and $\tau_{nn}$ are nonsingular. To see that $T^{-1}$ is upper triangular, note that by Theorem 1.1

$$T^{-1} = \begin{pmatrix} T_{n-1}^{-1} & -\tau_{nn}^{-1} & T_{n-1}^{-1} & t_n \\ 0 & \tau_{nn}^{-1} \end{pmatrix}. \quad (1.2)$$
By the induction hypothesis $T_{n-1}^{-1}$ is upper triangular. Hence $T^{-1}$ is upper triangular.  

We now turn to the problem of solving the linear system

$$Tx = b,$$  \hspace{1cm} (1.3)

where $T$ is a nonsingular upper triangular matrix of order $n$. This may be done as follows. The last equation of the system (1.3) is

$$\tau_{nn} \xi_n = \beta_n,$$

from which

$$\xi_n = \tau_{nn}^{-1} \beta_n.$$  \hspace{1cm} (1.4)

In general, if we know $\xi_n$, $\xi_{n-1}$, $\ldots$, $\xi_{i+1}$, we can solve the $i$th equation of (1.3), which is

$$\tau_{ii} \xi_i + \tau_{i,i+1} \xi_{i+1} + \cdots + \tau_{in} \xi_n = \beta_i,$$

to obtain

$$\xi_i = \tau_{ii}^{-1} \left( \beta_i - \sum_{j=i+1}^{n} \tau_{ij} \xi_j \right).$$  \hspace{1cm} (1.5)

The formulas (1.4) and (1.5) define an algorithm for the computation of the solution of (1.3).

**Algorithm 1.3.** Let $T \in \mathbb{R}^{n \times n}$ be upper triangular and nonsingular, and let $b \in \mathbb{R}^n$. This algorithm computes the solution of the equation $Tx = b$.

1) For $i = n, n - 1, \ldots, 1$

$$\xi_i = \tau_{ii}^{-1} \left( \beta_i - \sum_{j=i+1}^{n} \tau_{ij} \xi_j \right)$$

Algorithm 1.3 for solving an upper triangular system is sometimes called the method of back substitution. Mathematically, the algorithm can fail only when some $\tau_{ii} = 0$, in which case $\tau_{ii}^{-1}$ in statement 1.1 is not defined. By Theorem 1.2 this can happen only when $T$ is singular. In other words, if $T$ is nonsingular, Algorithm 1.3 can be carried to completion.

In practice, however, the calculations must be performed on a computer, usually in floating-point arithmetic. We shall postpone discussing the effects of rounding errors until Section 5; however, we note that in statement 1.1 of the algorithm an inner product must be computed and some
accuracy may be gained by accumulating it in double precision. (Incidentally, note that in accordance with our INFL conventions no inner product is computed in the first step, when i = n.)

Another important practical consideration is the possibility of the algorithm failing because of overflow, even though T is nonsingular. For example, if

\[
T = \begin{pmatrix}
10^{-60} & 1 \\
0 & 10^{-60}
\end{pmatrix}
\]

and \( b = (0, 1)^T \), then \( x = (-10^{120}, 10^{60})^T \). On a computer whose floating-point word has a characteristic bounded by 100, the first component of \( x \) cannot be represented as a floating-point number, and any attempt to execute Algorithm 1.3 for this data will result in an overflow. At the very least, such an overflow should cause the computations to stop and a diagnostic message to be printed. On many computers, this is done by the system, which regards any floating-point overflow as a fatal error.

Like Algorithm 3.6 of the preceding chapter, Algorithm 1.3 makes no reference to the elements in the lower part of the array \( T \). Thus this storage is free for other purposes, say to store the elements of a strictly lower triangular matrix.

Finally, note that Algorithm 1.3 requires about \( n^2/2 \) multiplications.

Algorithm 1.3 can be adapted to compute the inverse \( S \) of an upper triangular matrix \( T \). Specifically, if \( S = (s_1, s_2, \ldots, s_n) \), then the equation \( TS = I \) is equivalent to the set of equations

\[
T s_k = e_k \quad (k = 1, 2, \ldots, n).
\]

Each of these equations may be solved by Algorithm 1.3 to yield the inverse \( S \).

However, this is an inefficient procedure, for half of the elements of \( S \) are known to be zero and it is senseless to compute them. To circumvent this difficulty, let \( s'_k = (s_{1k}, s_{2k}, \ldots, s_{kk})^T \). Then it is easily verified that

\[
T^{(k)} s'_k = e_k \quad (k = 1, 2, \ldots, n),
\]

where \( e_k \) now is a \( k \)-vector. Equation (1.6) is an upper triangular system of order \( k \) for the nonzero elements of \( s_k \) which may be solved by Algorithm 1.3. If this is done for each \( k \), there results an algorithm for inverting an upper triangular matrix. For reasons that will become apparent later, we solve the equations in the order \( k = n, n - 1, \ldots, 1 \) (cf. Algorithm 1.5).
**Algorithm 1.4.** Let \( T \in \mathbb{R}^{n \times n} \) be a nonsingular upper triangular matrix. This algorithm computes \( T^{-1} \) and returns it in the upper half of the array \( S \).

1) For \( k = n, n - 1, \ldots, 1 \)

1) \( \sigma_{kk} = \tau_{kk}^{-1} \)

2) \( \sigma_{ik} = -\tau_{ii}^{-1} \sum_{j=i+1}^{k} \tau_{ij} \sigma_{jk} \quad (i = k - 1, k - 2, \ldots, 1) \)

In deriving this algorithm we have taken advantage of the fact that all components but the last of \( e_k \) are zero. Like Algorithm 1.3, this algorithm is mathematically well defined when \( T \) is nonsingular. Practically, overflow may prevent its complete execution. Some accuracy may be gained by accumulating inner products in double precision.

When it is not required to save the elements of the matrix \( T \), they may be overwritten by the elements of \( S = T^{-1} \). In fact, in Algorithm 1.4, once \( \tau_{ik} \) has been used to compute \( \sigma_{ik} \) it is no longer needed, and hence we may overwrite \( \tau_{ik} \) by \( \sigma_{ik} \). This results in the following algorithm for inverting \( T \) in its own array.

**Algorithm 1.5.** Let \( T \in \mathbb{R}^{n \times n} \) be a nonsingular upper triangular matrix. This algorithm overwrites \( T \) with its inverse \( S \).

1) For \( k = n, n - 1, \ldots, 1 \)

1) \( \tau_{kk} \leftarrow \sigma_{kk} = \tau_{kk}^{-1} \)

2) \( \tau_{ik} \leftarrow \sigma_{ik} = -\tau_{ii}^{-1} \sum_{j=i+1}^{k} \tau_{ij} \sigma_{jk} \quad (i = k - 1, k - 2, \ldots, 1) \)

Both Algorithms 1.4 and 1.5 require about \( n^3/6 \) multiplications which differs from the multiplication count for Algorithm 1.3 by a factor of \( n/3 \). This fact has important consequences for the solution of triangular systems. A naive scheme for solving, say, Equation (1.3) might go as follows.

1) \( S = T^{-1} \)

2) \( x = Sb \)

where the first step is accomplished by Algorithm 1.5. However, statement 1 requires \( n^3/6 \) multiplications, beside which the \( n^2/2 \) multiplications of statement 2 are negligible. Obviously, when \( n \) is large, this procedure is far more expensive than Algorithm 1.3. In other words, if one requires only the solution of (1.3), one should not compute \( T^{-1} \).
It should be stressed that the inverse of a matrix is seldom required in matrix computations. For whenever we are asked to compute

\[ x = T^{-1}b, \]

we can alternatively calculate the solution of

\[ Tx = b. \]

Such recasting of a problem can save a good deal of computations, as the following examples show.

**EXAMPLE 1.6.** We wish to calculate the expression

\[ \alpha = x^T T^{-1} y, \]

where \( T \) is an upper triangular matrix of order \( n \). Rather than inverting \( T \), we may proceed as follows.

1) Solve the equation \( Tz = y \)
2) Calculate \( \alpha = x^T z \)

The naive way of computing \( \alpha \) via \( T^{-1} \) requires \( n^3/6 \) multiplications to invert \( T \), beside which the remaining calculations are insignificant. The alternative procedure, on the other hand, requires only \( n^2/2 \) multiplications.

**EXAMPLE 1.7.** We wish to compute

\[ C = T^{-1} B. \]

where \( T \) is of order \( n \) and \( B \in \mathbb{R}^{n \times m} \). If we partition \( B \) and \( C \) by columns, then

\[ (c_1, c_2, \ldots, c_m) = (T^{-1}b_1, T^{-1}b_2, \ldots, T^{-1}b_m), \]

whence \( c_i = T^{-1}b_i \). Thus the columns of \( C \) can be found by solving the systems

\[ Tc_i = b_i \quad (i = 1, 2, \ldots, m). \]

If \( T \) is upper triangular, the naive computation requires \( n^3/6 \) multiplications for the inversion of \( T \) and another \( mn^2/2 \) multiplications to compute \( C = T^{-1}B \), giving a total of \((\frac{1}{3}n + \frac{1}{2}m)n^3 \) multiplications. The alternative procedure requires \( mn^2/2 \) multiplications. Comparing these operation
counts, we see that the procedure involving $T^{-1}$ always requires more work than the alternative procedure, although the difference becomes negligible for $m \gg n$.

EXERCISES

1. Show that if $T = D - U$, where $D$ is a nonsingular diagonal matrix of order $n$ and $U$ is a strictly upper triangular matrix, then
   
   $$T^{-1} = D^{-1}[I + UD^{-1} + (UD^{-1})^2 + \cdots + (UD^{-1})^{n-1}]$$

2. Describe an equivalent of Algorithm 1.1 for solving lower triangular systems.

3. Describe an equivalent of Algorithm 1.4 for inverting lower triangular matrices.

4. Write an INFL program to generate the inverse of an upper triangular matrix $T$ row-wise. [Hint: Solve the systems $s_k^T = e_k^n$ ($k = 1, 2, \ldots, n$).]

5. A lower triangular matrix is a Stieltjes matrix if its nonzero elements lie on its diagonal and first subdiagonal. Describe an efficient algorithm for inverting a lower Stieltjes matrix $L$; for solving the system $Lx = b$.

6. Call a matrix of order $n$ "upper Stieltjes of width $k$" if its nonzero elements lie on its diagonal and first $k - 1$ superdiagonals. Describe an efficient algorithm to invert an upper Stieltjes matrix $U$ of width $k$; to solve the equation $Ux = b$.

7. Equation (1.2) shows that $(T^{(k)})^{-1}$ may be easily calculated from $(T^{(k-1)})^{-1}$. Describe an algorithm that computes $T^{-1}$ by computing successively $(T^{(1)})^{-1}, (T^{(2)})^{-1}, \ldots, (T^{(n)})^{-1} = T^{-1}$.

8. A matrix $T$ is block upper triangular if it can be partitioned in the form

   $$T = \begin{pmatrix} T_{11} & T_{12} & \cdots & T_{1m} \\ 0 & T_{22} & \cdots & T_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & T_{mm} \end{pmatrix}$$

   where each diagonal block $T_{ii}$ is square. Prove that $T$ is nonsingular
if and only if its diagonal blocks are nonsingular, and that its inverse is block triangular with the same partitioning. Assuming that the matrices $T_{11}$ are known, generalize Algorithm 1.5 to find the inverse of a block upper triangular matrix. Give two INFL descriptions: one in terms of blocks and the other in terms of elements.

2. GAUSSIAN ELIMINATION

In this section we shall give algorithms for reducing a matrix to upper trapezoidal form. The basic algorithm, called Gaussian elimination, mimics the process of eliminating unknowns from a system of linear equations. As such it provides a direct method for solving linear equations. Moreover, the algorithms may be described in terms of matrix operations, and these descriptions in turn lead to the factorizations to be described in Section 3.

To illustrate these ideas we consider the following system of linear equations

$$
\begin{align*}
2\xi_1 + 4\xi_2 - 2\xi_3 &= 6 \\
\xi_1 - \xi_2 + 5\xi_3 &= 0 \\
4\xi_1 + \xi_2 - 2\xi_3 &= 2
\end{align*}
$$

(2.1)

If $\frac{1}{2}$ times the first equation is subtracted from the second and 2 times the first subtracted from the third, the result is the system

$$
\begin{align*}
2\xi_1 + 4\xi_2 - 2\xi_3 &= 6 \\
-3\xi_2 + 6\xi_3 &= -3 \\
-7\xi_2 + 2\xi_3 &= -10
\end{align*}
$$

(2.2)

in which the variable $\xi_1$ does not appear in the second and third equations. Likewise the variable $\xi_2$ can be eliminated from the third equation by subtracting 7/3 times the second equation from the third to obtain

$$
\begin{align*}
2\xi_1 + 4\xi_2 - 2\xi_3 &= 6 \\
-3\xi_2 + 6\xi_3 &= -3 \\
12\xi_3 &= -3
\end{align*}
$$

(2.3)

The system (4.3) is upper triangular and can be solved by the techniques of Section 1. Obviously this idea of elimination can be extended to give a
3. THE DIRECT SOLUTION OF LINEAR SYSTEMS

general method for solving systems of any order. However, there is more to the process than this.

Let $A = A_1, A_2,$ and $A_3$ be the matrices of the systems (2.1), (2.2), and (2.3), respectively; e.g.,

$$A_3 = \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & -7 & -2 \end{pmatrix}.$$

Then $A_2$ is obtained from $A_1$ by subtracting $\frac{1}{2}$ the first row of $A_1$ from the second row and 2 times the first row from the third. From the discussion of matrix multiplication in Section 1.4, we know that $A_2$ can be obtained by premultiplying $A_1$ by a suitable matrix, and in fact it is easy to verify that

$$A_2 = M_1 A_1, \quad (2.4)$$

where

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}.$$  

Similarly

$$A_3 = M_2 A_2, \quad (2.5)$$

where

$$M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{pmatrix}.$$  

Combining (2.4) and (2.5), we see that

$$A_3 = M_2 M_1 A. \quad (2.6)$$

Thus to solve the system $Ax = b$ we need only calculate the vector $c = M_1 M_2 b$ and solve the upper triangular system

$$A_3 x = M_2 M_1 A x = M_2 M_1 b = c.$$  

Moreover, the expression (2.6) can be rearranged to give a factorization of $A$ into a lower and upper triangular matrix. Since the $M_i$ are nonsingular, we have from (2.6)

$$A = M_1^{-1} M_2^{-1} A_3. \quad (2.7)$$
Since $M_1$ and $M_2$ are unit lower triangular, so is the product of their inverses. Hence, if we set $L = M_1^{-1}M_2^{-1}$ and $U = A_3$, Equation (2.7) becomes

$$A = LU,$$

where $L$ is unit lower triangular and $U$ is upper triangular. Since

$$M_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ 2 & 0 & 1 \end{pmatrix}, \quad M_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{4} & 1 \end{pmatrix},$$

$L$ is itself the product of very simple triangular matrices.

Obviously the above discussion depends critically on the matrices $M_i$, which are called elementary lower triangular matrices. We begin our formal exposition of Gaussian elimination with a discussion of their properties.

**Definition 2.1.** An elementary lower triangular matrix of order $n$ and index $k$ is a matrix of the form

$$M = I_n - m e_k^T,$$

where

$$e_i^T m = 0 \quad (i = 1, 2, \ldots, k). \quad (2.8)$$

The conditions (2.8) say that the first $k$ components of $m$ are zero; that is, $m$ has the form $m = (0, 0, \ldots, 0, \mu_{k+1}, \mu_{k+2}, \ldots, \mu_n)^T$. In general an elementary lower triangular matrix has the form

$$M = \begin{bmatrix}
1 & 0 & \cdots & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 & \cdots & 0 \\
0 & 0 & \cdots & -\mu_{k+1} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & -\mu_n & \cdots & 1
\end{bmatrix}$$

Thus an elementary lower triangular matrix is an identity matrix with some additional nonzero elements in the $k$th column below the diagonal.

Elementary lower triangular matrices are easily inverted.
THEOREM 2.2. Let $M = I - me_k^T$ be an elementary lower triangular matrix. Then

$$M^{-1} = I + me_k^T.$$ 

PROOF. Let $X = I + me_k^T$. Then

$$MX = (I - me_k^T)(I + me_k^T)$$

$$= I - me_k^T + me_k^T - me_k^Tme_k^T$$

$$= I - m(e_k^Tm)e_k^T.$$ 

Since $M$ is an elementary lower triangular matrix, $e_k^Tm = 0$. Hence $MX = I$ and $X$ is the inverse of $M$. □

The computational significance of elementary lower triangular matrices is that they can be used to introduce zero components into a vector.

THEOREM 2.3. Let $e_k^Tx = \xi_k \neq 0$. Then there is a unique elementary lower triangular matrix $M$ of index $k$ such that

$$Mx = (\xi_1, \xi_2, \ldots, \xi_k, 0, 0, \ldots, 0)^T. \quad (2.9)$$

PROOF. We seek $M$ in the form $M = I - me_k^T$. Since $M$ is to be of index $k$, we must have

$$\mu_i = 0 \quad (i = 1, 2, \ldots, k). \quad (2.10)$$

Now

$$Mx = (I - me_k^T)x = x - (e_k^Tx)m.$$ 

Since the last $n - k$ components of $Mx$ are to be zero, we must have

$$\xi_i - \xi_k\mu_i = 0 \quad (i = k + 1, k + 2, \ldots, n),$$

and if $\xi_k \neq 0$,

$$\mu_i = \frac{\xi_i}{\xi_k} \quad (i = k + 1, k + 2, \ldots, n). \quad (2.11)$$

Thus if $M$ exists, it is uniquely determined by (2.10) and (2.11). On the other hand, if $\xi_k \neq 0$, then the vector $m$ can be determined from (2.10).
and (2.11), and it is easy to verify that the associated matrix \( M = I - me_k^T \) satisfies (2.9).

Thus when \( \xi_k \neq 0 \), we can multiply \( x \) by an elementary lower triangular matrix chosen so that the last \( n - k \) components of \( x \) are replaced by zeros and the other components are left unaltered. When \( \xi_k = 0 \), such a matrix does not exist, unless \( \xi_{k+1}, \ldots, \xi_n \) are also zero, in which case any elementary lower triangular matrix of index \( k \) will do the job. It should be noted that Equations (2.10) and (2.11) define an algorithm for computing \( m \) and hence \( M \).

In the method of Gaussian elimination for reducing a matrix to upper trapezoidal form, the matrix is premultiplied by a sequence of elementary lower triangular matrices, each chosen to introduce a column with zeros below the diagonal. The method revolves around Theorem 2.3, which allows us to calculate the necessary elementary lower triangular matrices. It should be stressed that any class of matrices satisfying something like Theorem 2.3 can be used for the reduction: the elementary lower triangular matrices are used here because they are easy to compute and multiply. An important alternative will be discussed in Chapter 5.

Let \( A_1 = A \) be an \( m \times n \) matrix. If \( a_{11}^{(1)} = \alpha_{11} \) is nonzero, then by Theorem 2.3 there is a unique elementary lower triangular matrix \( M_1 \) of index 1 that annihilates the last \( n - 1 \) elements of the first column of \( A_1 \). If \( A_1 \) is premultiplied by \( M_1 \), there results a matrix \( A_2 \) of the form

\[
A_2 = M_1 A_1 = \begin{pmatrix}
\alpha_{11}^{(1)} & \alpha_{12}^{(1)} & \alpha_{13}^{(1)} & \cdots & \alpha_{1n}^{(1)} \\
0 & \alpha_{22}^{(2)} & \alpha_{23}^{(2)} & \cdots & \alpha_{2n}^{(2)} \\
0 & \alpha_{32}^{(2)} & \alpha_{33}^{(2)} & \cdots & \alpha_{3n}^{(2)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \alpha_{m2}^{(2)} & \alpha_{m3}^{(2)} & \cdots & \alpha_{mn}^{(2)}
\end{pmatrix}
\]

Note that the first row of \( A_2 \) is the same as the first row of \( A_1 \).

In general, suppose \( A_k = M_{k-1} \cdots M_1 A_1 \) has the form

\[
A_k = \begin{pmatrix}
A_{11}^{(k)} & A_{12}^{(k)} \\
0 & A_{22}^{(k)}
\end{pmatrix},
\]

where \( A_{11}^{(k)} \) is an upper triangular matrix of order \( k - 1 \) and for \( i = 1, 2, \ldots, k - 1 \) the \( i \)th row of \( A_k \) is the same as the \( i \)th row of \( A_i \). For example,
with \( m = n = 5 \) we have

\[
A_3 = \begin{pmatrix}
\alpha_{11}^{(1)} & \alpha_{12}^{(1)} & \alpha_{13}^{(1)} & \alpha_{14}^{(1)} & \alpha_{15}^{(1)} \\
0 & \alpha_{22}^{(2)} & \alpha_{23}^{(2)} & \alpha_{24}^{(2)} & \alpha_{25}^{(2)} \\
0 & 0 & \alpha_{33}^{(3)} & \alpha_{34}^{(3)} & \alpha_{35}^{(3)} \\
0 & 0 & \alpha_{43}^{(3)} & \alpha_{44}^{(3)} & \alpha_{45}^{(3)} \\
0 & 0 & \alpha_{53}^{(3)} & \alpha_{54}^{(3)} & \alpha_{55}^{(3)} 
\end{pmatrix}
\]

If \( \alpha_{kk}^{(k)} \neq 0 \), then there is an elementary lower triangular matrix \( M_k \) of index \( k \) that annihilates the last \( m - k \) elements of the \( k \)th column of \( A_k \). Such a matrix can be written in the form

\[
M_k = \begin{pmatrix}
I_{k-1} & 0 \\
0 & M'_k
\end{pmatrix},
\]

where \( M'_k \) is an elementary lower triangular matrix of index 1.

Now

\[
A_{k+1} = M_k A_k = \begin{pmatrix}
I & 0 \\
0 & M'_k
\end{pmatrix} \begin{pmatrix}
A_{11}^{(k)} & A_{12}^{(k)} \\
0 & A_{22}^{(k)}
\end{pmatrix} = \begin{pmatrix}
A_{11}^{(k)} & A_{12}^{(k)} \\
0 & M'_k A_{22}^{(k)}
\end{pmatrix}.
\]

Thus first \( k - 1 \) rows of \( A_{k+1} \) are the same as those of \( A_k \). Since the first row of \( A_{22}^{(k)} \) and \( M'_k A_{22}^{(k)} \) are the same, the first \( k \) rows of \( A_k \) and \( A_{k+1} \) are the same. Also by the construction of \( M_k \), \( M'_k A_{22}^{(k)} \) has the form

\[
\begin{pmatrix}
x & x & \cdots & x \\
0 & x & \cdots & x \\
\vdots & \vdots & \ddots & \vdots \\
0 & x & \cdots & x
\end{pmatrix}
\]

Hence the leading principal submatrix of order \( k \) of \( A_{k+1} \) is upper triangular. In other words, \( A_{k+1} \) is one step further along in the reduction.

The reduction will terminate when we run out of rows or columns. When \( m > n \), this happens after the \( n \)th step. When \( m \leq n \), the matrix is in lower trapezoidal form after the \( (m - 1) \)th step. For example, when \( m = 1 \), no reduction is required, since a row vector is trivially in upper trapezoidal form. Thus the reduction ends with the matrix \( A_{r+1} \), where

\[
r = \min \{m - 1, n\}.
\]
The reduction of \( A \) is constructive in the sense that we can specify an algorithm for calculating the \( A_k \) and the \( M_k \). Specifically, the \( k \)-th column of \( A_k \) is the vector

\[
a_k^{(k)} = (\alpha_{1k}^{(k)}, \alpha_{2k}^{(k)}, \ldots, \alpha_{kk}^{(k)}, \ldots, \alpha_{mk}^{(k)})^T,
\]

and \( M_k \) is the elementary lower triangular matrix of index \( k \) such that

\[
M_k a_k^{(k)} = (\alpha_{1k}^{(k)}, \alpha_{2k}^{(k)}, \ldots, \alpha_{kk}^{(k)}, 0, \ldots, 0)^T.
\]

By Theorem 2.3, \( M_k = I - m_k e_k^T \), where

\[
m_k = (0, 0, \ldots, 0, \mu_{k+1,k}, \ldots, \mu_{mk})^T
\]

and

\[
\mu_{ik} = \frac{\alpha_{ik}^{(k)}}{\alpha_{kk}^{(k)}} \quad (i = k + 1, k + 2, \ldots, m). \quad (2.12)
\]

Thus \( M_k \) is uniquely determined, provided \( \alpha_{kk}^{(k)} \neq 0 \).

In principle \( A_{k+1} \) may be calculated by forming \( M_k \) and computing the product \( M_k A_k \). However, considerable savings in operations and storage may be obtained by taking advantage of the special form of \( M_k \). In fact,

\[
A_{k+1} = M_k A_k = (I - m_k e_k^T) A_k = A_k - m_k e_k^T A_k.
\]

Now \( e_k^T \) is the \( k \)-th row of \( A_k \). Hence the \( i \)-th row of \( A_{k+1} \) may be formed by subtracting \( \mu_{ik} \times \) the \( k \)-th row of \( A_k \) from the \( i \)-th row of \( A_k \). Since \( \mu_{ik} = \mu_{ik} = \cdots = \mu_{kk} = 0 \), only rows \( k + 1, k + 2, \ldots, m \) are altered. In terms of elements this becomes

\[
\alpha_{ik}^{(k+1)} = 0 \quad (i = k + 1, k + 2, \ldots, m) \quad (2.13)
\]

and

\[
\alpha_{ij}^{(k+1)} = \alpha_{ij}^{(k)} - \mu_{ik} \alpha_{kj}^{(k)} \quad (i = k + 1, k + 2, \ldots, m; \ j = k + 1, k + 2, \ldots, n). \quad (2.14)
\]

Equations (2.12)-(2.14) completely specify the computations involved in the reduction. As far as storage is concerned, we can allow the elements of \( A_{k+1} \) to overwrite the corresponding elements of \( A_k \). In applications, such as solving linear systems, it is important to know the matrices \( M_k \). This can be done by saving the numbers \( \mu_{ij} \). After \( \alpha_{ik}^{(k)} \) is used to compute
μ_{ik}$, it is not used again, and in fact it is reduced to zero in the transformation from $A_k$ to $A_{k+1}$. Hence, if we agree to store only the nonzero entries of $A_{k+1}$, the number $μ_{ik}$ may be placed in the location originally occupied by $a_{ik}$.

These considerations may be summed up in the following algorithm.

**ALGORITHM 2.4. (Gaussian elimination).** Let the $m \times n$ matrix $A$ be given with $m > 1$ and let $r = \min\{m - 1, n\}$. This algorithm overwrites $A$ with the upper trapezoidal matrix $A_{r+1}$. The multipliers $μ_{ik}$ overwrite $a_{ik}$.

1) For $k = 1, 2, \ldots, r$

1) $a_{ik} \leftarrow μ_{ik} = \frac{a_{ik}}{a_{kk}} \quad (i = k + 1, k + 2, \ldots, m)$

2) $a_{ij} \leftarrow a_{ij} - μ_{ik}a_{kj} \quad (i = k + 1, \ldots, m; \ j = k + 1, \ldots, n)$

For $r$ large, the algorithm requires approximately

$$\frac{r^3}{3} - (m + n) \frac{r^2}{2} + mnr$$

multiplications. For $m = n$, this amounts to approximately $n^3/3$ multiplications.

The numbers $α_{kk}^{(k)}$ ($k = 1, 2, \ldots, r$) are called the pivot elements of the reduction. As it is given, Algorithm 2.4 can be carried to completion only if all the pivot elements are nonzero. For purposes of investigating the algorithm, it is convenient to have a condition on the original matrix under which the pivots are guaranteed to be nonzero. Such a condition is given in the following theorem.

**THEOREM 2.5.** The pivot elements $α_{ii}^{(i)}$ ($i = 1, 2, \ldots, k$), are nonzero if and only if the leading principal submatrices $A^{(i)}$ ($i = 1, 2, \ldots, k$), are nonsingular.

**PROOF.** The proof is by induction on $k$. For $k = 1$, the theorem is trivially true since $A^{(1)} = α_{11}^{(1)}$. For the induction step it is sufficient to assume that $A^{(1)}, \ldots, A^{(k-1)}$ are nonsingular and show that $A^{(k)}$ is nonsingular if and only if $α_{kk}^{(k)}$ is nonzero.

By the induction hypothesis, $α_{11}^{(1)}, α_{22}^{(2)}, \ldots, α_{k-1,k-1}^{(k-1)} \neq 0$. Hence Algorithm 2.4 may be carried out through its $(k - 1)$th step to give elementary lower
triangular matrices $M_i$ of index $i$ ($i = 1, 2, \ldots, k - 1$) such that

$$A_k = M_{k-1}M_{k-2} \cdots M_1A = \begin{pmatrix} A^{(k)}_{11} & A^{(k)}_{12} \\ 0 & A^{(k)}_{22} \end{pmatrix},$$

where $A^{(k)}_{11}$ is upper triangular with diagonal elements $\alpha^{(1)}_{11}, \ldots, \alpha^{(k-1)}_{k-1,k-1}$. It follows that $A^{(k)}_k$ is upper triangular with diagonal elements $\alpha^{(1)}_{11}, \ldots, \alpha^{(k-1)}_{k-1,k-1}, \alpha^{(k)}_{kk}$. Since $\alpha^{(1)}_{11}, \ldots, \alpha^{(k-1)}_{k-1,k-1} \neq 0$, $A^{(k)}_k$ is nonsingular if and only if $\alpha^{(k)}_{kk}$ is nonzero. Now since $M_1, \ldots, M_{k-1}$ are lower triangular, $A^{(k)}_k = M^{(k)}_{k-1} \cdots M^{(k)}_1A^{(k)}$ (Exercise 1.4.18). Since $M_i$ is unit lower triangular, so is $M^{(k)}_i$. Thus $A^{(k)}_i$ is nonsingular if and only if $A^{(k)}_k$ is nonsingular, which, we have seen, is true if and only if $\alpha^{(k)}_{kk} \neq 0$.

Carried through its $k$th step, the method of Gaussian elimination produces elementary lower triangular matrices $M_i$ of index $i$ ($i = 1, 2, \ldots, k$) such that $A_{k+1} = M_kM_{k-1} \cdots M_1A$ is zero below its first $k$ diagonal elements. Once started, the method proceeds deterministically, provided the pivots are nonzero. This suggests that the matrices $M_1, M_2, \ldots, M_k$ are uniquely determined by the requirement that $M_kM_{k-1} \cdots M_1A$ be zero below its first $k$ diagonal elements. The following theorem shows that this is indeed the case.

**THEOREM 2.6.** Let $A \in \mathbb{R}^{m \times n}$ and suppose that $A^{(1)}_1, A^{(2)}_2, \ldots, A^{(k)}_k$ are nonsingular. For $i = 1, 2, \ldots, k$ let $M_i$ and $N_i$ be elementary lower triangular matrices of index $i$. If $M_kM_{k-1} \cdots M_1A$ and $N_kN_{k-1} \cdots N_1A$ are both zero below their first $k$ diagonal elements, then $M_i = N_i$ ($i = 1, 2, \ldots, k$).

**PROOF.** The proof is by induction on $k$. For $k = 1$, $M_1$ and $N_1$ are elementary lower triangular matrices of index 1 that introduce zeros into the last $m - 1$ elements of the first column of $A$. Since $\alpha_{11} \neq 0$, this requirement uniquely determines $M_1$ and $N_1$ (Theorem 2.3) which are thereby equal.

For the induction step, assume that $A^{(1)}_1, \ldots, A^{(k)}_k$ are nonsingular and that $M_k \cdots M_1A$ and $N_k \cdots N_1A$ are both zero below their first $k$ diagonal elements. Since $M_k$ is an elementary lower triangular matrix of index $k$, so is $M_k^{-1}$, and it is easily verified that

$$M_{k-1} \cdots M_1A = M_k^{-1}N_k \cdots N_1A$$

is zero below its first $k - 1$ diagonal elements. Likewise $N_{k-1} \cdots N_1A$ is
zero below its first \( k - 1 \) diagonal elements. By the induction hypothesis \( M_i = N_i \) \((i = 1, 2, \ldots, k - 1)\). Hence \( M_{k-1} \cdots M_1 A = N_{k-1} \cdots N_1 A = A_k \), where \( A_k \) is the matrix resulting from \( k - 1 \) steps of Gaussian elimination. By Theorem 2.5, \( \alpha_{kk}^{(k)} \neq 0 \). Hence \( M_k \) and \( N_k \) are the unique elementary lower triangular matrices of index \( k \) that introduce zeros below the \( k \)th diagonal of \( A_k \). It follows that \( M_k = N_k \).

Algorithm 2.4 breaks down at step 1.1 when the pivot \( \alpha_{kk} = 0 \). If some \( \alpha_{ik} \neq 0 \) \((i = k + 1, \ldots, m)\), then the reduction cannot be continued. However, if \( \alpha_{ik} = 0 \) \((i = k + 1, \ldots, m)\), then \( A_k \) is already zero below its \( k \)th diagonal element, and any elementary lower triangular matrix of index \( k \) can be used for \( M_k \), thus if a zero pivot emerges at step \( k \), it may be possible to continue the reduction, but \( M_k, M_{k+1}, \ldots, M_r \) are no longer unique.

**EXAMPLE 2.7.** We illustrate the various situations that can arise in the reduction with the following \( 2 \times 2 \) examples. Note that \( r = 1 \) and hence \( \alpha_{11} \) is the deciding factor in the reduction.

1. \( \alpha_{11} \neq 0 \).
   \[
   A_1 = \begin{pmatrix} 1 & 2 \\ 1 & 3 \end{pmatrix}, \quad M_1 = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}.
   \]

2. \( \alpha_{11} \neq 0 \).
   \[
   A_1 = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}, \quad M_1 = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix}.
   \]

Here \( A_1 \) is singular. Nonetheless \( M_1 \) is uniquely determined.

3. \( \alpha_{11} = 0 \).
   \[
   A_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.
   \]

The reduction cannot be carried out.

4. \( \alpha_{11} = 0 \).
   \[
   A_1 = \begin{pmatrix} 0 & 1 \\ 0 & 2 \end{pmatrix}.
   \]

Here the reduction can be carried out, but it is not unique. In fact for any \( \beta \) we may set
   \[
   M_1 = \begin{pmatrix} 1 & 0 \\ \beta & 1 \end{pmatrix},
   \]
When a mathematical process fails to be defined for a particular value of a parameter, there is a good chance that the corresponding numerical process will be unstable when the parameter is near the offending value. We have seen that the above reduction of a matrix to trapezoidal form fails when the pivot element \( a_{kk}^{(k)} \) is zero. The following example illustrates the consequences of computing with a small pivot.

**EXAMPLE 2.8.** Let

\[
A_1 = \begin{pmatrix}
0.001 & 2.000 & 3.000 \\
-1.000 & 3.712 & 4.623 \\
-2.000 & 1.072 & 5.643
\end{pmatrix},
\]

Then a four-digit realization of the reduction of \( A \) to upper triangular form will produce the following matrices.

\[
M_1 = \begin{pmatrix}
1.000 & 0.000 & 0.000 \\
1000. & 1.000 & 0.000 \\
2000. & 0.000 & 1.000
\end{pmatrix},
\]

\[
A_2 = \begin{pmatrix}
0.001 & 2.000 & 3.000 \\
0.000 & 2004. & 3005. \\
0.000 & 4001. & 6006.
\end{pmatrix},
\]

\[
M_2 = \begin{pmatrix}
1.000 & 0.000 & 0.000 \\
0.000 & 1.000 & 0.000 \\
0.000 & -1.997 & 1.000
\end{pmatrix},
\]

\[
A_3 = \begin{pmatrix}
0.001 & 2.000 & 3.000 \\
0.000 & 2004. & 2005. \\
0.000 & 0.000 & 5.000
\end{pmatrix}.
\]

There is reason to suspect this calculation, for severe cancellation occurred in the calculation of the \((3, 3)\)-element of \( A_3 \) by the formula

\[
\text{fl}[6006. - (1.997)(3005.)].
\]

In fact the correct value of the \((3, 3)\)-element rounded to four figures is 5.922, which disagrees with the computed value in the second figure. If this
computed value is used in subsequent calculations, the results will in general be accurate to only one or two figures.

The above example suggests that the emergence of a small pivot in the course of Algorithm 2.4 may be a harbinger of disaster. If the problem at hand requires the matrices $M_1, M_2, \ldots, M_r, A_{r+1}$, there is very little that can be done other than redoing the calculations in higher precision. However, in many applications the reduction is an intermediate step in the solution of a larger problem and it may be possible to rearrange the problem so that the reduction is performed on a different matrix, one for which no small pivots emerge. For example, suppose the matrix in Example 2.8 had arisen in connection with solving the linear system

$$
0.001\xi_1 + 2.000\xi_2 + 3.000\xi_3 = 1.000 \\
-1.000\xi_1 + 3.712\xi_2 + 4.623\xi_3 = 2.000 \\
-2.000\xi_1 + 1.072\xi_2 + 5.643\xi_3 = 3.000
$$

If the first and second equations are interchanged, there results a linear system whose matrix

$$
\begin{pmatrix}
-1.000 & 3.712 & 4.623 \\
0.001 & 2.000 & 3.000 \\
-2.000 & 1.072 & 5.643
\end{pmatrix}
$$

can be reduced without difficulty.

The above considerations suggest that we modify Algorithm 2.4 to avoid small pivotal elements by interchanging the rows and columns of the matrix $A$. Specifically, if at the $k$th stage of the reduction, the element $a_{kk}^{(k)}$ is unsatisfactorily small, we may choose some other element, say $a_{\theta_k, \gamma_k}^{(k)} \neq 0$, as the pivot. If we interchange rows $k$ and $\theta_k$ and then columns $k$ and $\gamma_k$, the result is to move $a_{\theta_k, \gamma_k}^{(k)}$ into the $(k, k)$-position, and the reduction may be continued with the new pivot. Obviously we must have $\theta_k \geq k$ and $\gamma_k \geq k$, otherwise the interchanges will disturb the zeros previously introduced by the reduction.

In matrix terms the modified algorithm may be described as follows. At the $k$th stage, $A_k$ is premultiplied and postmultiplied, respectively, by the elementary permutations $I_k, e_{\theta_k}$ and $I_k, e_{\gamma_k}$ (cf. Example 1.4.26), and then premultiplied by an elementary lower triangular matrix $M_k$ of index $k$ chosen to introduce a new column of zeros. Thus

$$
A_{k+1} = M_k I_k, e_{\theta_k} A_k I_k, e_{\gamma_k}.
$$
For simplicity of notation, we shall denote the elementary permutation
$I_{k, e_k}$ and $I_{k, y_k}$ by $P_k$ and $Q_k$, respectively, so that

$$A_{k+1} = M_k P_k A_k Q_k. \tag{2.15}$$

The incorporation of interchanges into Algorithm 2.4 complicates the
equation for $A_k$. It might be thought that it would be considerably more
difficult to analyze the modified algorithm. Fortunately, as the following
theorem shows, the modified algorithm is equivalent to making all the
interchanges first and then applying Algorithm 2.4. Of course in practice
we must make the interchanges as we go along, since there is no way to
know if a pivot is zero until it has been computed; but for theoretical
purposes (say a rounding-error analysis) we may assume that all the
interchanges have already been performed.

**THEOREM 2.9.** Let the modified algorithm be applied to $A$ producing
matrices $Q_i, P_i, M_i,$ and $A_i$ ($i = 1, 2, \ldots, k$). Let

$$A' = P_k P_{k-1} \cdots P_i A Q_i Q_2 \cdots Q_k.$$  

Then Algorithm 2.4 can be applied to $A'$ through its $k$th step. If $M_i'$ and
$A_{i+1}$ ($i = 1, 2, \ldots, k$), are the matrices produced by Algorithm 2.4, then

$$A'_{k+1} = A_{k+1}$$

and

$$M_i' = P_k P_{k-1} \cdots P_i M_i P_i+1 \cdots P_{k-1} P_k. \tag{2.16}$$

**PROOF.** From Equation (2.15) it follows that

$$A_{k+1} = M_k P_k M_{k-1} P_{k-1} \cdots M_i P_i A Q_i Q_2 \cdots Q_k.$$  

Because $P_i^2 = I$, this equation is equivalent to

$$A_{k+1} = M_k M_{k-1} \cdots M_i A', \tag{2.17}$$

where $M_i'$ is defined by (2.16). Now it is easily verified that if $M_k$ is an
elementary lower triangular matrix of index $k$ and $i, j > k$, then $I_{ij} M_k I_{ij}$
is also an elementary lower triangular matrix of index $k$. It follows that
$M_i'$ is an elementary lower triangular matrix of index $k$. The first $k$ diagonal
elements of $A_{k+1}$ are nonzero and hence the first $k$ leading principal sub-
matrices of $A'$ are nonsingular. Thus by Theorem 2.6, $M_1', M_2', \ldots, M_k'$,
and $A_{k+1}$ are the result of applying $k$ steps of Algorithm 2.4 to $A'$.  ■
There remains the problem of specifying a criterion for choosing the pivot element. Example 2.8 suggests that it may be desirable to choose the pivot as large as possible. In other words we wish to choose $q_k$ and $y_k$, both not less than $k$, so that

$$\left| a^{(k)}_{r_k,q_k} \right| \geq \left| a^{(k)}_{ij} \right| \quad (i = k, k+1, \ldots, m; \quad j = k, k+1, \ldots, n).$$

There is the possibility that this maximum element is zero; however, this can only happen when the last $m-k$ rows of $A_k$ are zero, and in this case $A_k$ is already in upper trapezoidal form. Incidentally, note that the above criterion does not completely specify $q_k$ and $y_k$, for the maximum may be attained for more than one element. Thus, in contrast with Algorithm 2.4 the modification does not determine a unique sequence of calculations for reducing $A$ to upper trapezoidal form.

The modified algorithm with the above choice of pivots is called Gaussian elimination with complete pivoting.

**Algorithm 2.10.** (Gaussian elimination with complete pivoting). Let $A \in \mathbb{R}^{m \times n}$ with $m > 1$. Let $r = \min \{m - 1, n\}$. This algorithm overwrites $A$ with the Gaussian decomposition of $A$ with its rows and columns permuted. The multiplier $\mu_{ij}$ overwrites $a_{ij}$. The row and column interchange indices are $q_k$ and $y_k$, respectively.

1) For $k = 1, 2, \ldots, r$

1) Find $q_k, y_k \geq k$ such that

$$\left| a_{q_k,y_k} \right| = \max \{ \left| a_{ij} \right| : i, j \geq k \}$$

2) If $a_{q_k,y_k} = 0$, set $r = k - 1$ and end the calculations

3) $\alpha_{kj} \leftarrow a_{q_k,j} \quad (j = k, k+1, \ldots, n)$

4) $\alpha_{ik} \leftarrow a_{i,y_k} \quad (i = 1, 2, \ldots, m)$

5) $\alpha_{ik} \leftarrow \mu_{ik} = \frac{\alpha_{ik}}{\alpha_{kk}} \quad (i = k+1, k+2, \ldots, m)$

6) $\alpha_{ij} \leftarrow \alpha_{ij} - \mu_{ik} \alpha_{kj} \quad (i = k+1, \ldots, m; \quad j = k+1, \ldots, n)$

The algorithm can always be carried to completion. The final matrix is in upper trapezoidal form. For some $k$, depending on where the algorithm stops, its last $m-k$ rows are zero and its leading principal submatrix of order $k$ is nonsingular. By Theorem 2.9, the same matrix may be obtained
by applying $k$ steps of Gaussian elimination to the $A$ with its rows and columns suitably permuted. Thus Gaussian elimination with complete pivoting is a procedure for calculating the decomposition described in the following theorem.

**THEOREM 2.11.** Let $A$ be an $m \times n$ matrix. Then if $A \neq 0$, there is a number $k \leq \min\{m, n\}$, elementary permutation matrices $P_i, Q_i$ ($i = 1, 2, \ldots, r$), and elementary lower triangular matrices $M_i$ of index $i$ such that

$$A_{r+1} = M_r M_{r-1} \cdots M_1 P_r P_{r-1} \cdots P_1 AQ_1 Q_1 \cdots Q_r$$

where $A_{r+1}$ is an $r \times r$ nonsingular upper triangular matrix.

In practice, step 1.1 of Algorithm 2.10 may consume a good deal of time, since it involves searching among $(m - k + 1)(n - k + 1)$ elements of $A$ for the largest. A frequently used alternative is to search only the $k$th column for the largest element and perform a row interchange to bring it into the pivotal position. If, at the $k$th step, the pivot is zero, the required zeros are already in the $k$th column, and we may take $M_k = I$. This pivoting strategy is called Gaussian elimination with partial pivoting. It yields an upper trapezoidal matrix, which, however, has no nice distribution of zero and nonzero diagonal elements.

**ALGORITHM 2.12.** (Gaussian elimination with partial pivoting). Let $A$ be an $m \times n$ matrix and $r = \min\{m - 1, n\}$. This algorithm overwrites $A$ with the Gaussian decomposition of $A$ with its rows permuted. The multiplier $\mu_{ij}$ overwrites $\alpha_{ij}$. The row interchange indices are $q_k$.

1) For $k = 1, 2, \ldots, r$

1) Find $q_k \geq k$ such that $|\alpha_{q_k,k}| = \max\{|\alpha_{ik}| : i \geq k\}$

2) If $\alpha_{q_k,k} = 0$, step $k$

3) $\alpha_{kj} \leftrightarrow \alpha_{q_k,j}$

(j = $k, k + 1, \ldots, n$)

4) $\alpha_{ik} \leftarrow \mu_{ik} = \frac{\alpha_{ik}}{\alpha_{kk}}$ (i = $k + 1, k + 2, \ldots, m$)

5) $\alpha_{ij} \leftarrow \alpha_{ij} - \mu_{ik} \alpha_{kj}$ (i = $k + 1, \ldots, m$; j = $k + 1, \ldots, n$)
THEOREM 2.13. Let $A$ be an $m \times n$ matrix and let $r = \min\{m - 1, n\}$. Then there are elementary permutations $P_i$ $(i = 1, 2, \ldots, r)$, and elementary lower triangular matrices $M_i$ of index $i$ $(i = 1, 2, \ldots, r)$, such that

$$A_{r+1} = M_rM_{r-1} \cdots M_1P_rP_{r-1} \cdots P_1A$$

is upper trapezoidal.

EXERCISES

1. Let $L \in \mathbb{R}^{n \times n}$ be unit lower triangular, and let $M_k$ be the elementary lower triangular matrix of index $k$ whose $k$th column is the same as the $k$th column of $L$. Show that $L = M_1M_2 \cdots M_{n-1}$. Apply Theorem 2.2 to derive algorithms for inverting $L$ and for solving the system $Lx = b$. How are these algorithms related to those of Exercises 1.2 and 1.3?

2. Let $M$ be an elementary lower triangular matrix of index $k$ and let $i, j > k$. Show that $M' = I_{ij}MI_{ij}$ is also an elementary lower triangular matrix of index $k$. How is $M'$ obtained from $M$?

3. Let $A, M \in \mathbb{R}^{n \times n}$ with $M$ an elementary lower triangular matrix of index 1. Give efficient INFL algorithms for overwriting $A$ with $AM^{-1}$, $AM^T$, $MAM^{-1}$, and $MAM^T$.

4. Define the notion of an elementary upper triangular matrix of order $n$ in a suitable way. Describe the important properties of such matrices.

5. Describe how a matrix can be reduced to lower trapezoidal form by postmultiplication by elementary upper triangular matrices. Give an INFL algorithm implementing this variation of Gaussian elimination. Discuss the incorporation of interchanges into the algorithm.

6. Give an efficient INFL program for reducing an upper Hessenberg matrix to upper triangular form by Gaussian elimination with partial pivoting.

7. Give an efficient INFL program for reducing a tridiagonal matrix to upper triangular form by Gaussian elimination (note that the result is Stieltjes matrix). Do the same for Gaussian elimination with partial pivoting. (The result is a Stieltjes matrix of width 3.) Also give algorithms in which the elements are suitably stored in linear arrays.
8. Give an efficient INFL program for reducing a band matrix of width $2k + 1$ (cf. Exercise 1.3.6) to upper triangular form by Gaussian elimination with partial pivoting.

9. Let $A$ be symmetric with $\alpha_{11} \neq 0$. After one step of Gaussian elimination $A$ has the form

$$
\begin{pmatrix}
\alpha_{11} & a_1^T \\
0 & A_2
\end{pmatrix}
$$

Show that $A_2$ is symmetric.

10. Give an efficient INFL algorithm for reducing a symmetric matrix $A$ to upper triangular form by Gaussian elimination. Assume that only the upper half of the matrix $A$ is present in the array $A$. Do the same for a symmetric band matrix of width $2k + 1$.

11. The matrix $A \in \mathbb{R}^{n \times n}$ is **diagonally dominant** if $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ ($i = 1, 2, \ldots, n$). Prove that

1. if $A$ is diagonally dominant, then any principal submatrix of $A$ is diagonally dominant,
2. if $A$ is diagonally dominant, then $A$ is nonsingular.

Conclude that Algorithm 2.4 will not fail when applied to a diagonally dominant matrix.

12. Let $A$ be diagonally dominant and after one step of Gaussian elimination let $A$ have the form

$$
\begin{pmatrix}
\alpha_{11} & a_1^T \\
0 & A_2
\end{pmatrix}
$$

Prove that $A_2$ is diagonally dominant. Conclude that for symmetric, diagonally dominant matrices Gaussian elimination and Gaussian elimination with partial pivoting amount to the same thing.

13. Describe how Gaussian elimination may be used to compute the decomposition of Theorem 1.7.1. (Note that as a practical procedure this approach has the drawback that one must be able to recognize when a number that has been contaminated by rounding error is zero.)

14. An elementary $R$-matrix of index $k$ is a matrix of the form $R = I - re_k^T$, where $e_k^T r = 0$. Prove that

1. $R^{-1} = I + re_k^T$,
2. $e_k^T x = 0 \Rightarrow Rx = x$.

(There is no standard terminology for such matrices.)
15. Prove that 'if \( e_k^T x \neq 0 \), then there is an elementary \( R \)-matrix of index \( k \) such that
\[
Rx = \xi_k e_k = (0, \ldots, 0, \xi_k, 0, \ldots, 0)^T.
\]

16. Gauss–Jordan reduction. Let \( A \in \mathbb{R}^{n \times n} \). Determine elementary \( R \)-matrices \( R_i \) of index \( i \) such that \( A_{k+1} = R_k R_{k-1} \cdots R_1 A \) has the form
\[
A_{k+1} = \begin{pmatrix} D_{k+1} & A_{12}^{(k+1)} \\ 0 & A_{22}^{(k+1)} \end{pmatrix},
\]
where \( D_{k+1} \) is diagonal of order \( k \). Thus \( A_{n+1} \) is diagonal. Give an INFL program implementing this Gauss–Jordan reduction of \( A \) to diagonal form.

17. Show that the matrices \( A_{22}^{(k+1)} \) in the Gauss–Jordan reduction are the same as the corresponding matrices resulting from Gaussian elimination. Hence derive conditions under which the Gauss–Jordan reduction can be carried to completion.

18. What kind of interchange strategies may be incorporated into the Gauss–Jordan reduction?

19. Strictly speaking, Algorithm 2.12 does not produce the Gaussian decomposition of \( A \) with its rows permuted, since the multipliers \( \mu_{ij} \) do not appear in the correct order. Modify the algorithm so that the \( \mu_{ij} \) do appear in the correct order. [Hint: only one minor change in one statement is required.]

20. The necessity of actually interchanging rows in Algorithm 2.12 can be circumvented by the following trick. Initialize an index vector \( I \) with the values \( \lambda_1 = 1, \lambda_2 = 2, \ldots, \lambda_n = n \). Whenever a row interchange is required, interchange instead the corresponding components of \( I \). Replace all references to \( a_{ij} \) by references to \( a_{\lambda_i, \lambda_j} \). Give INFL code for this variant of Algorithm 2.12. Discuss the situation of the multipliers (cf. Exercise 2.19). Is there any significant difference in work between this variant and Algorithm 2.12?

21. Devise a variant of Algorithm 2.10 that avoids interchanges.

NOTES AND REFERENCES

A variant of Gaussian elimination for systems of order three appears in the Chinese work "Chiu-Chang Suan-Shu," composed about 250 B.C.
The method is readily motivated in terms of systems of linear equations, and in one form or another it often appears in linear algebra texts. Unfortunately many of these texts fail to describe the pivoting strategies that are so necessary to the numerical stability of the methods.

The point of view taken here in which Gaussian elimination is regarded as being accomplished by premultiplications by elementary lower triangular matrices originated with Turing (1948) and has been extensively exploited by Wilkinson (AEP, for example). This approach has the advantage of generalizing readily; any other computationally convenient class of matrices may be used to introduce zeros.

Elementary lower triangular matrices are a special case of a class of matrices that differ from an identity matrix by a matrix of rank unity. Householder (1964) calls all such matrices elementary matrices and shows how many of the transformations used in numerical linear algebra may be reduced to multiplications by elementary matrices.

Gaussian elimination lends itself naturally to the reduction of matrices with special distributions of zeros such as tridiagonal matrices, for it is easy to see what the pivoting and elimination operations will do to the zero elements. Martin and Wilkinson (1967, HACLA/I/6) give algorithms for efficiently reducing band matrices.

A natural extension of Gaussian elimination is to eliminate all the off-diagonal elements in a column at each step, which will finally yield a diagonal matrix. This method is known as the Gauss-Jordan method [the association of Jordan with the method is apparently a mistake; see Householder (1964, page 141)]. In most applications this method is unnecessarily expensive; however, Bauer and Reinsch (HACLA/I/3) have used it to invert positive definite matrices.

An excellent introduction to Gaussian elimination, as well as other algorithms in this chapter, has been given by Forsythe and Moler (1967).

3. TRIANGULAR DECOMPOSITION

In this section we shall consider algorithms for factoring a square matrix \( A \) into the product \( LU \) of a lower and an upper triangular matrix. Such a factorization is often called an \( LU \) decomposition of the matrix \( A \). Since an \( LU \) decomposition expresses a matrix as the product of simpler matrices, it can be used to simplify calculations involving the matrix. For example, in the unlikely event that the inverse of \( A \) is required, it can be computed
in the form $A^{-1} = U^{-1}L^{-1}$ by the techniques of Section 3.1. Throughout this section we shall assume that $A$ is a square matrix of order $n$; however, some of the results may easily be generalized to rectangular matrices.

Actually we have already seen a method for computing an $LU$ factorization of $A$, for if Gaussian elimination is applied to $A$, the result is a sequence $M_1, M_2, \ldots, M_{n-1}$ of unit lower triangular matrices such that

$$A_n = M_{n-1} M_{n-2} \cdots M_1 A$$

is upper triangular. If we set

$$L = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1},$$

then $L$ is unit lower triangular and $A = LA_n$. Moreover, it is easy to verify that

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \alpha_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & 1 \end{pmatrix},$$

(3.1)

where the $\alpha_{ij}$ are multipliers produced by Algorithm 2.4. Thus Algorithm 2.4 can be regarded as an algorithm for overwriting $A$ with an $LU$ decomposition, $L$ being stored in the lower part of the array $A$ and $U = A_n$ in the upper part.

The close connection of Gaussian elimination with an $LU$ decomposition of $A$ raises the question of the existence and uniqueness of such decompositions; for we have seen that the Gaussian decomposition may fail to exist or be unique. There is, in general, no unique $LU$ factorization of a matrix. If $A = LU$ is an $LU$ factorization of $A$ and $D$ is a nonsingular diagonal matrix, then $L' = LD$ is lower triangular and $U' = D^{-1}U$ is upper triangular. Hence

$$A = LU = LDD^{-1}U = L'U',$$

and $L'U'$ is also an $LU$ decomposition. This example suggests the possibility of normalizing the $LU$ decompositions of a matrix by inserting a diagonal matrix. We shall say that

$$A = LDU$$

is an $LDU$ decomposition of $A$ provided that $L$ is unit lower triangular, $D$ is diagonal, and $U$ is unit upper triangular. The question of the existence
and uniqueness of LDU decompositions may be answered in terms of the leading principal submatrices of \( A \).

**THEOREM 3.1.** Let \( A \in \mathbb{R}^{n \times n} \). Then \( A \) has a unique \( LDU \) decomposition if and only if \( A^{[1]}, A^{[2]}, \ldots, A^{[n-1]} \) are nonsingular.

**PROOF.** We prove the theorem under the hypothesis that \( A \) is nonsingular, leaving the case of singular \( A \) as a (rather difficult) exercise.

We first show that if \( A \) has an \( LDU \) decomposition, then it is unique. Let \( A = L_1D_1U_1 \) and \( A = L_2D_2U_2 \) be \( LDU \) decompositions of \( A \). Since \( A \) is nonsingular, so are \( D_1 \) and \( D_2 \). From the equation \( L_1D_1U_1 = L_2D_2U_2 \), it follows that

\[
L_2^{-1}L_1 = D_2U_2U_1^{-1}D_1^{-1}.
\]  
(3.2)

Now the left-hand side of (3.2) is a unit lower triangular matrix and the right-hand side is an upper triangular matrix. Hence both sides are an identity matrix. This means \( L_2^{-1}L_1 = I \) or \( L_1 = L_2 \). A similar argument shows that \( U_1 = U_2 \). Finally, since \( L_1 \) and \( U_1 \) are nonsingular, the equation \( L_1D_1U_1 = L_1D_2U_1 \) implies that \( D_1 = D_2 \).

It remains to show that \( A \) has an \( LDU \) decomposition if and only if \( A^{[1]}, \ldots, A^{[n-1]} \) are nonsingular. First suppose that \( A = LDU \) is an \( LDU \) decomposition of \( A \). Then \( L \), \( D \), and \( U \) are nonsingular. Since \( L \) and \( U \) are triangular and \( D \) is diagonal, \( L^{[k]}, D^{[k]}, \) and \( U^{[k]} \) are nonsingular. However, \( A^{[k]} = L^{[k]}D^{[k]}U^{[k]} \), hence \( A^{[k]} \) is nonsingular.

Conversely, suppose that \( A^{[1]}, A^{[2]}, \ldots, A^{[n-1]} \) are nonsingular. Then by Theorem 2.5 the algorithm of Gaussian elimination can be carried to completion and we can write \( A = LA_n \), where \( L \) is given by (3.1). Now the diagonal elements of \( A_n \) are the pivot elements \( a^{(k)}_{kk} \) \((k = 1, 2, \ldots, n)\), which by Theorem 2.5 are nonzero. Let \( D = \text{diag}(a^{(1)}_{11}, a^{(2)}_{22}, \ldots, a^{(n)}_{nn}) \) and let \( U = D^{-1}A_n \). Then \( A = LDU \) is an \( LDU \) decomposition of \( A \).

From the proof of Theorem 3.1 it is evident that when an \( LDU \) decomposition exists, its elements can be written in terms of the quantities calculated by Algorithm 2.4. Specifically, \( i \neq j \),

\[
\begin{align*}
\lambda_{ij} &= \mu_{ij} \\
\delta_i &= \alpha_i^{(i)} \\
v_{ij} &= \frac{\alpha_{ij}^{(i)}}{\alpha_i^{(i)}}
\end{align*}
\]  
(3.3)

\( i = 1, 2, \ldots, n; \ j = 1, 2, \ldots, i - 1 \),

\( i = 1, 2, \ldots, n \),

\( i = 1, 2, \ldots, n; \ j = i, i + 1, \ldots, n \).
We now turn to the description of algorithms for computing $LU$ decompositions of a nonsingular matrix. Different $LU$ decompositions may be obtained by treating the diagonal in the $LDU$ decomposition differently. There are three important variants. The first associates $D$ with the lower triangular part to give the factorization

$$A = L'U = (LD)U.$$ 

This is known as the Crout decomposition. The second, called the Doolittle decomposition, associates $D$ with the upper triangular part

$$A = LU' = L(DU).$$

When $A$ is symmetric and has a unique $LDU$ decomposition, the decomposition must have the form

$$A = LDL^T.$$ 

If the diagonal elements of $D$ are positive, then we can form the matrix $D^{1/2} = \text{diag}(\delta_{11}^{1/2}, \ldots, \delta_{nn}^{1/2})$. Thus $A$ can be written as

$$A = L'L'^T = (LD^{1/2})(D^{1/2}L^T).$$

This third variant is known as the Cholesky decomposition of $A$.

Any of these three decompositions can be calculated by performing the Gaussian reduction on $A$ and using (3.3) to calculate the $LDU$ decomposition (in fact, Gaussian elimination gives the Doolittle decomposition immediately). However, there are practical advantages to having algorithms that compute the decompositions directly. We shall confine our discussion to the Crout and Cholesky decompositions.

In the Crout reduction, we seek a factorization of the form

$$A = LU,$$ \hfill (3.4)

where $L$ is lower triangular and $U$ is unit upper triangular. This can be accomplished as follows. In terms of scalars, Equation (3.4) can be written

$$\alpha_{ij} = \sum_{k=1}^{\min(i,j)} \lambda_{ik}v_{kj} \quad (i, j = 1, 2, \ldots, n).$$ \hfill (3.5)

In particular, since $v_{11} = 1$,

$$\alpha_{i1} = \lambda_{i1}v_{11} = \lambda_{i1} \quad (i = 1, 2, \ldots, n).$$ \hfill (3.6)
Moreover
\[ \alpha_{ij} = \lambda_{11} v_{1j}, \]
so that
\[ v_{1j} = \frac{\alpha_{1j}}{\lambda_{11}} \quad (j = 2, 3, \ldots, n). \]

Equations (3.6) and (3.7) determine the first column of \( L \) and the first row of \( U \).

Now suppose that the first \( k - 1 \) columns of \( L \) and the first \( k - 1 \) rows of \( U \) have been computed. Then since \( v_{kk} = 1 \),
\[ \alpha_{ik} = \lambda_{ik} + \sum_{l=1}^{k-1} \lambda_{il} v_{lk} \quad (i = k, k + 1, \ldots, n), \]
whence
\[ \lambda_{ik} = \alpha_{ik} - \sum_{l=1}^{k-1} \lambda_{il} v_{lk} \quad (i = k, k + 1, \ldots, n). \]

Similarly
\[ v_{kj} = \lambda_{kk}^{-1} \left( \alpha_{kj} - \sum_{l=1}^{k-1} \lambda_{kl} v_{lj} \right) \quad (j = k + 1, \ldots, n). \]

Equations (3.8) and (3.9) express the elements of the \( k \)th column of \( L \) and the \( k \)th row of \( U \) in terms of known quantities.

Concerning the organization of storage, note that after \( \alpha_{ij} \) is used to compute \( \lambda_{ij} \) or \( v_{ij} \), whichever, it is no longer used. Hence the nonzero elements of \( L \) and \( U \) can overwrite the corresponding elements of \( A \) as they are generated. Thus the Crout decomposition can be computed by the following algorithm.

**Algorithm 3.2. (Crout reduction).** Let \( A \) be of order \( n \). This algorithm overwrites \( A \) with an \( LU \) decomposition, where \( U \) is unit upper triangular.

1) For \( k = 1, 2, \ldots, n \)
\[ 1) \quad \alpha_{ik} \leftarrow \lambda_{ik} = \alpha_{ik} - \sum_{l=1}^{k-1} \lambda_{il} v_{lk} \quad (i = k, \ldots, n) \]
\[ 2) \quad \alpha_{kj} \leftarrow v_{kj} = \lambda_{kk}^{-1} \left( \alpha_{kj} - \sum_{l=1}^{k-1} \lambda_{kl} v_{lj} \right) \quad (j = k + 1, \ldots, n) \]

By (3.3) the elements \( \lambda_{ij} \) are the elements \( \alpha_{ij}^{(2)} \) of the matrix \( A_j \) in the Gaussian reduction. In particular the elements \( \lambda_{kk} \) are the pivots in the Gaussian
reduction and are nonzero provided the leading principal submatrices of $A$ are nonsingular. Hence Algorithm 3.2 can be carried to completion if the leading principal submatrices of $A$ are nonsingular.

The algorithm requires approximately $n^3/3$ multiplications for its completion, which is the same as for Gaussian elimination. Since Equations (3.3) allow us to obtain one decomposition from the other, there would seem to be no reason for preferring one over the other. However, in the Crout reduction the inner products in statements 1.1 and 1.2 can be accumulated in double precision, thereby reducing the effects of rounding errors. An equivalent modification for Gaussian elimination would require one to retain the elements $a_{ij}^{(k)}$ in double precision, which doubles the storage requirements. Thus on a computer that can accumulate inner products in double precision, the Crout reduction is preferable to Gaussian elimination.

The following example illustrates the computations involved in the Crout reduction of a $3 \times 3$ matrix. It also shows the disastrous effects of a small diagonal element.

**EXAMPLE 3.3.** Let

\[ A = \begin{pmatrix} 0.001 & 2.000 & 3.000 \\ -1.000 & 3.712 & 4.623 \\ -2.000 & 1.072 & 5.643 \end{pmatrix} \]

(A is the matrix of Example 2.8). In four-digit arithmetic, the Crout reduction goes as follows.

\[
\lambda_{11} = 0.001, \\
\lambda_{21} = -1.000, \\
\lambda_{31} = -2.000, \\
v_{13} = \text{fl} \left( \frac{2.000}{0.001} \right) = 2000, \\
v_{13} = \text{fl} \left( \frac{3.000}{0.001} \right) = 3000, \\
\lambda_{32} = \text{fl}[3.712 + (1.000)(2000)] = 2004, \\
\lambda_{32} = \text{fl}[1.072 + (2.000)(2000)] = 4001, \\
v_{23} = \text{fl} \left( \frac{4.623 + (1.000)(3000)}{2004} \right) = 1.500, \\
\lambda_{33} = \text{fl}[5.643 + (2.000)(3000) - (4001)(1.500)] = 4.000.
\]
Note that the calculation of \( \lambda_{32} \) entails the cancellation of three significant figures. Hence we can expect the computed value of \( \lambda_{32} \) to be very inaccurate, and indeed the true value of \( \lambda_{32} \), rounded to four figures, is 5.922. The accumulation of inner products will not help, for an error in the fourth place of, say, \( \lambda_{32} \) generates an error in the first place of \( \lambda_{33} \). In other words, the act of rounding \( \lambda_{32} \) completely destroys the possibility of computing \( \lambda_{33} \) accurately.

Example 3.3 indicates the necessity of avoiding small diagonal elements in the Crout reduction. As we did in Gaussian elimination, we shall seek a cure for the problem by permuting the rows and columns of the original matrix \( A \). There is no convenient analog of complete pivoting for the Crout reduction; hence we shall confine ourselves to partial pivoting, that is to row interchanges.

The principal obstacle to incorporating interchanges into the Crout algorithm is that we cannot know that \( \lambda_{kk} \) is small until we have computed it. However, at this stage an interchange in the rows of \( A \) will change its Crout decomposition, which will then have to be recomputed. Fortunately there is a very simple relation between the Crout decompositions of a matrix and the matrix obtained by interchanging two of its rows.

This relation is best derived by considering what happens when Gaussian elimination is applied to \( A \) with the two of its rows are permuted. For definiteness suppose that \( A_1 = A \) is of order 5 and the third and fifth rows of \( A \) are interchanged to give a new matrix

\[
A'_1 = \begin{pmatrix}
\alpha_{11}^{(1)} & \alpha_{12}^{(1)} & \alpha_{13}^{(1)} & \alpha_{14}^{(1)} & \alpha_{15}^{(1)} \\
\alpha_{21}^{(1)} & \alpha_{22}^{(1)} & \alpha_{23}^{(1)} & \alpha_{24}^{(1)} & \alpha_{25}^{(1)} \\
\alpha_{31}^{(1)} & \alpha_{32}^{(1)} & \alpha_{33}^{(1)} & \alpha_{34}^{(1)} & \alpha_{35}^{(1)} \\
\alpha_{41}^{(1)} & \alpha_{42}^{(1)} & \alpha_{43}^{(1)} & \alpha_{44}^{(1)} & \alpha_{45}^{(1)} \\
\alpha_{51}^{(1)} & \alpha_{52}^{(1)} & \alpha_{53}^{(1)} & \alpha_{54}^{(1)} & \alpha_{55}^{(1)}
\end{pmatrix}
\]

By examining Algorithm 2.4, it is easy to verify that after one step of Gaussian elimination, \( A'_1 \) becomes

\[
A'_2 = \begin{pmatrix}
\alpha_{11}^{(1)} & \alpha_{12}^{(1)} & \alpha_{13}^{(1)} & \alpha_{14}^{(1)} & \alpha_{15}^{(1)} \\
\mu_{21}^{(2)} & \alpha_{22}^{(2)} & \alpha_{23}^{(2)} & \alpha_{24}^{(2)} & \alpha_{25}^{(2)} \\
\mu_{31}^{(2)} & \alpha_{32}^{(2)} & \alpha_{33}^{(2)} & \alpha_{34}^{(2)} & \alpha_{35}^{(2)} \\
\mu_{41}^{(2)} & \alpha_{42}^{(2)} & \alpha_{43}^{(2)} & \alpha_{44}^{(2)} & \alpha_{45}^{(2)} \\
\mu_{51}^{(2)} & \alpha_{52}^{(2)} & \alpha_{53}^{(2)} & \alpha_{54}^{(2)} & \alpha_{55}^{(2)}
\end{pmatrix}
\]
where the $\alpha$'s and $\mu$'s are the same quantities that would be obtained by applying Algorithm 2.4 to $A_1$. Another step gives

$$A_3' = \begin{pmatrix}
\alpha_{11}^{(1)} & \alpha_{12}^{(1)} & \alpha_{13}^{(1)} & \alpha_{14}^{(1)} & \alpha_{15}^{(1)} \\
\mu_{21}^{(2)} & \alpha_{22}^{(2)} & \alpha_{23}^{(2)} & \alpha_{24}^{(2)} & \alpha_{25}^{(2)} \\
\mu_{51}^{(3)} & \alpha_{52}^{(3)} & \alpha_{53}^{(3)} & \alpha_{54}^{(3)} & \alpha_{55}^{(3)} \\
\mu_{41}^{(3)} & \alpha_{42}^{(3)} & \alpha_{43}^{(3)} & \alpha_{44}^{(3)} & \alpha_{45}^{(3)} \\
\mu_{31}^{(3)} & \alpha_{32}^{(3)} & \alpha_{33}^{(3)} & \alpha_{34}^{(3)} & \alpha_{35}^{(3)}
\end{pmatrix}.$$

It follows from (3.3) that if we apply Algorithm 3.2 to $A_1'$ and stop after statement 1.1 when $k = 3$, we shall have obtained the array

$$\begin{pmatrix}
\hat{\lambda}_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\
\hat{\lambda}_{21} & \lambda_{22} & v_{23} & v_{24} & v_{25} \\
\hat{\lambda}_{51} & \lambda_{52} & \lambda_{53} & \alpha_{54} & \alpha_{55} \\
\hat{\lambda}_{41} & \lambda_{42} & \lambda_{43} & \alpha_{44} & \alpha_{45} \\
\hat{\lambda}_{31} & \lambda_{32} & \lambda_{33} & \alpha_{34} & \alpha_{35}
\end{pmatrix}.$$

In other words, after statement 1.1 when $k = 3$ the only differences between the arrays obtained from $A$ and $A'$ is that the third and fifth row have been interchanged.

This is generally the case. If after statement 1.1 in Algorithm 3.2 we interchange rows $k$ and $l$ ($l > k$), the effect will be the same as if we had interchanged rows $k$ and $l$ in the original matrix before starting. This means that we can perform interchanges as we go along, as is done in the following algorithm.

**Algorithm 3.4.** Let $A$ be of order $n$. This algorithm overwrites $A$ with the Crout decomposition of $I_{n-1} \cdot I_{n-1} \cdot \ldots \cdot I_{n-1} \cdot A$.

1)  \hspace{1em} For $k = 1, 2, \ldots, n$
   1)  \hspace{1em} $\alpha_{ik} \leftarrow \lambda_{ik} = \alpha_{ik} - \sum_{l=1}^{k-1} \lambda_{il} v_{lk}$ \hspace{1em} ($i = k, \ldots, n$)
   2)  \hspace{1em} Find $\varrho_k$ such that $| \lambda_{\varrho_k,k} | \geq | \lambda_{ik} |$ \hspace{1em} ($i = k, \ldots, n$)
   3)  \hspace{1em} $\alpha_{kj} \leftarrow \alpha_{\varrho_k,j}$ \hspace{1em} ($j = 1, 2, \ldots, n$)
   4)  \hspace{1em} $\alpha_{kj} \leftarrow v_{kj} = \lambda_{kj}^{-1} \left( \alpha_{kj} - \sum_{l=1}^{k-1} \lambda_{kl} v_{lj} \right)$ \hspace{1em} ($j = k + 1, \ldots, n$)
The numbers $\lambda_{kk}, \lambda_{k+1,k}, \ldots, \lambda_{nk}$ are the numbers $\alpha_{kk}^{(k)}, \alpha_{k+1,k}^{(k)}, \ldots, \alpha_{n,k}^{(k)}$ obtained by applying Gaussian elimination to $I_{k-1,e_{k-1}} \cdots I_{e_1} A$. Thus in choosing $\rho_k$ to maximize $|\lambda_{q_k,k}|$ we are making the same choices as we would in Gaussian elimination with partial pivoting. In other words, if for each $k$ there is a unique largest $\lambda_{i,k}$ ($i = k, \ldots, n$), then the pivot indices are the same as those that would result from Gaussian elimination with partial pivoting.

We now turn to the Cholesky decomposition of a symmetric matrix $A$ into the product $LL^T$, where $L$ is lower triangular. Such a decomposition need not exist. For example, since

$$\alpha_{11} = \lambda_{11}^2 \geq 0,$$

no matrix with a negative (1, 1)-element can have a Cholesky decomposition. However, for a very important class of matrices, called positive definite matrices, the decomposition exists. Hence we shall begin our discussion of the Cholesky decomposition with an exposition of the properties of positive definite matrices.

DEFINITION 3.5. A symmetric matrix $A$ is positive definite if

$$x \neq 0 \Rightarrow x^T Ax > 0,$$

and positive semidefinite if

$$x \neq 0 \Rightarrow x^T Ax \geq 0.$$

It might be felt that the criterion (3.10) is not very practical for establishing whether a given matrix is positive definite. In the following example we use (3.10) to show that the members of a very broad class of matrices are automatically positive definite.

EXAMPLE 3.6. Let $A$ be an $m \times n$ matrix and let

$$B = A^T A.$$

In Example 1.4.21 we showed that $B$ is symmetric. We shall now show that $B$ is positive semidefinite and that if $\text{rank}(A) = n$, it is positive definite. To see this let $x \in \mathbb{R}^n$ be nonzero and $y = Ax$. Then

$$x^T B x = x^T A^T A x = y^T y = \sum_{i=1}^n \eta_i^2 \geq 0.$$
Hence $B$ is positive semidefinite. If $A$ is of rank $n$, then $x \neq 0$ implies $y = Ax \neq 0$. Hence from (3.11), $x^T B x > 0$, and $B$ is positive definite.

Before establishing the existence of $LL^T$ decomposition for positive definite matrices we prove the following lemma.

**Lemma 3.7.** A principal submatrix of a positive definite matrix is positive definite.

**Proof.** Let $A'$ be the principal submatrix formed from the intersection of rows and columns $i_1 < i_2 < \cdots < i_r$. Let $x' \neq 0$ be an $r$-vector. Let $x$ be the $n$-vector defined by

$$
\xi_k = \xi_k' \quad (k = 1, 2, \ldots, r),
$$

$$
\xi_j = 0 \quad (j \neq i_1, i_2, \ldots, i_r).
$$

Then $x \neq 0$, and it is easily verified that $x^T A x = x'^T A' x'$. Since $A$ is positive definite

$$
0 < x^T A x = x'^T A' x',
$$

and $A'$ is positive definite.

**Theorem 3.8.** If $A$ is positive definite, then there is a unique lower triangular matrix $L$ with positive diagonal elements such that $A = LL^T$.

**Proof.** The proof is by induction on the order of $A$. If $A$ is of order unity and positive definite, then $\alpha_{11} > 0$, and $L$ is uniquely defined by $\lambda_{11} = \sqrt{\alpha_{11}}$.

Suppose the assertion is true of matrices of order $n - 1$ and $A'$ is positive definite of order $n$. Because $A'$ is symmetric, it can be partitioned in the form

$$
A' = \begin{pmatrix} A & a \\ a^T & \alpha \end{pmatrix}.
$$

By Lemma 3.7, $A$ is positive definite. We seek a lower triangular matrix $L'$ satisfying $L'L'^T = A'$. Let $L'$ be partitioned in the form

$$
L' = \begin{pmatrix} L & 0 \\ 0 & \lambda \end{pmatrix}.
$$

...
Then we require that
\[ A = LL^T, \quad (3.12) \]
\[ Ll = a, \quad (3.13) \]
\[ f^TL^T = a^T, \quad (3.14) \]
\[ f^Tl + \lambda^2 = \alpha. \quad (3.15) \]

By the induction hypothesis, there is a unique lower triangular matrix \( L \) with positive diagonal elements satisfying (3.12). Since \( L \) is nonsingular, \( l = L^{-1}a \) is the unique vector satisfying (3.13) and (3.14). Finally if \( \alpha - f^Tl > 0 \), then \( \lambda \) will be uniquely defined by \( \lambda = \sqrt{\alpha - f^Tl} \).

To show that \( \alpha - f^Tl > 0 \), note that the nonsingularity of \( L \) implies that \( A \) is nonsingular. Let \( b = A^{-1}a \). Then because \( A \) is positive definite

\[
0 < (b^T, -1) \begin{pmatrix} A & a \\ a^T & \alpha \end{pmatrix} \begin{pmatrix} b \\ -1 \end{pmatrix} \\
= b^TAb - 2b^Ta + \alpha \\
= \alpha - b^Ta \\
= \alpha - a^T A^{-1}a \\
= \alpha - a^T (LL^T)^{-1}a \\
= \alpha - (L^{-1}a)^T(L^{-1}a) \\
= \alpha - f^Tl. \quad \blacksquare
\]

The proof of Theorem 3.8 allows us to construct the Cholesky decomposition of a matrix by computing successively the decompositions of its leading principal submatrices. Let \( A_k \) be the leading principal submatrix of order \( k \) and let \( A_k = L_kL_k^T \) be the Cholesky decomposition of \( A_k \). Then if

\[
A_k = \begin{pmatrix} A_{k-1} & a_k \\ a_k^T & \alpha_{kk} \end{pmatrix},
\]

it follows from the proof of the theorem that

\[
L_k = \begin{pmatrix} L_{k-1} & 0 \\ l_k^T & \lambda_{kk} \end{pmatrix},
\]
where
\[ l_k = L_k^{-1}a_k \] (3.16)
and
\[ \lambda_{kk} = \sqrt{\alpha_{kk} - l_k^Tl_k}. \]

In constructing an algorithm for computing the Cholesky decomposition of a positive definite matrix \( A \), we note that, since \( A \) is symmetric, we need only work with, say, its lower half. Moreover, the elements of \( L \) can overwrite the corresponding elements of \( A \). Of course, in computing \( l_k \) from (3.16), we do not form \( L_k^{-1} \); rather we solve the system
\[ L_k l_k = a_k. \]

**Algorithm 3.9. (Cholesky reduction).** Let \( A \) be positive definite of order \( n \). This algorithm returns the matrix \( L \) of the Cholesky decomposition of \( A \) in the lower half of the array \( A \).

1) For \( k = 1, 2, \ldots, n \)
   
   1) For \( i = 1, 2, \ldots, k - 1 \)
      
      1) \( \alpha_{ki} \leftarrow \lambda_{ki} = \lambda_{ii}^{-1}\left(\alpha_{ki} - \sum_{j=1}^{i-1} \lambda_{ij}\lambda_{kj}\right) \)
      
      2) \( \alpha_{kk} \leftarrow \lambda_{kk} = \sqrt{\alpha_{kk} - \sum_{j=1}^{k-1} \lambda_{kj}^2} \)

If \( A \) is positive definite, the algorithm can always be carried to completion. It requires about \( n^3/6 \) multiplications, one half the number required for Gaussian elimination or the Crout reduction. This is to be expected, since the algorithm takes advantage of the symmetry of \( A \) to reduce the calculations involved. As in the Crout reduction, inner products may be accumulated in double precision for additional accuracy.

**Exercises**

1. Let \( A \) be symmetric and nonsingular and let \( A = LDU \) be an \( LDU \) decomposition of \( A \). Show that \( L = U^T \).

2. Give an efficient INFL program to calculate the Crout decomposition of an upper Hessenberg matrix.
3. In the Crout reduction let

\[
L_k = \begin{pmatrix}
\lambda_{11} & 0 \\
0 & 1 \\
\vdots & \ddots & \ddots \\
\lambda_{k1} & \cdots & \lambda_{kk}
\end{pmatrix}, \quad M_k = \begin{pmatrix}
\lambda_{k+1,1} & \cdots & \lambda_{k+1,k} \\
\vdots & \ddots & \vdots \\
\lambda_{n1} & \cdots & \lambda_{nk}
\end{pmatrix}, \\
U_k = \begin{pmatrix}
1 & \cdots & v_{1k} \\
\vdots & \ddots & \vdots \\
0 & \cdots & 1
\end{pmatrix}, \quad V_k = \begin{pmatrix}
v_{1,k+1} & \cdots & v_{1n} \\
\vdots & \ddots & \vdots \\
v_{k,k+1} & \cdots & v_{kn}
\end{pmatrix}.
\]

Let \( A \) be partitioned in the form

\[
A = \begin{pmatrix}
A^{[k]} & B_k \\
C_k & D_k
\end{pmatrix}.
\]

Show that the equations \( L_k U_k = A^{[k]} \), \( L_k V_k = B_k \), \( M_{k+1} U_{k+1} = C_{k+1} \), uniquely determine \( L_k, U_k, V_k, \) and \( M_{k+1} \) provided \( A^{[1]}, A^{[2]}, \ldots, A^{[k]} \) are nonsingular. Use this result to give a formal justification of interchanges in Algorithm 3.4.

4. Prove that if \( A \) is positive definite, then \( A^{-1} \) is positive definite.

5. Prove that if \( A \) is positive definite, then \( A \) can be written uniquely in the form \( A = L^T L \), where \( L \) is a lower triangular matrix with positive diagonal elements. [Hint: Apply Theorem 3.8 to \( A^{-1} \).]

6. Show that if \( A \) is symmetric, diagonally dominant, and has positive diagonal elements, then \( A \) is positive definite [Hint: By Exercise 2.11, Gaussian elimination may be performed on \( A \) without pivoting. Show that the pivot elements are positive and hence conclude \( A = LL^T \), where \( L \) is lower triangular with positive diagonal elements. For an easier way see Exercise 6.4.10.]

7. Prove that if \( A \) is positive definite, then Algorithm 2.4 (Gaussian elimination without pivoting) will not fail.

8. Show that the element of a positive definite matrix that is largest in absolute value lies on the diagonal.

9. Give an efficient INFL program to calculate the Cholesky decomposition of a positive definite tridiagonal matrix.

10. Give an INFL program to overwrite a positive definite matrix \( A \) with its inverse. Work only with the lower half of the array \( A \). [Hint:
Use Algorithm 3.9 to compute $L$ of the Cholesky decomposition. Then compute $L^{-1}$ and $L^{-T}L^{-1}$.

11. Let $A$ be positive definite. If one step of Gaussian elimination is performed on $A$, the result is a matrix of the form

$$
\begin{pmatrix}
\alpha & a^T \\
0 & A'
\end{pmatrix}.
$$

Show that $A'$ is positive definite.

12. Devise a variant of Algorithm 3.4 that, in the spirit of Exercise 2.20, avoids interchanges.

NOTES AND REFERENCES

In view of the close connections between Gaussian elimination and methods of triangular decomposition, it is not surprising that many of the "new" methods that have appeared in the literature from time to time are simply rearrangements of Gaussian elimination. The term "$LDU$ decomposition" is due to Turing (1948), who related it to Gaussian elimination. A complete discussion is given by Householder (1964). See also the book "Numerical Methods of Linear Algebra" by Faddeev and Faddeeva (1960, 1963), which although it is somewhat out of date, contains much other interesting material.

Programs implementing the Crout reduction have been published by Bowdler, Martin, Peters, and Wilkinson (1966, HACLA/I/7). The Cholesky decomposition has been implemented by Martin, Peters, and Wilkinson (1965, HACLA/I/1) for full positive definite matrices by Martin and Wilkinson (1965, HACLA/I/4) for positive definite band matrices.

4. THE SOLUTION OF LINEAR SYSTEMS

In the introductory material to Section 2 we indicated how the method of Gaussian elimination could be used to solve the linear system

$$Ax = b,$$

(4.1)

where $A$ is of order $n$. In this section we shall describe in detail how the decompositions of Sections 2 and 3 can be used to solve (4.1). The essential feature of the resulting methods is that the reduction of $A$ and the solution of (4.1) can be separated: once $A$ has been reduced, the reduced form can be used to solve (4.1) for any number of right-hand sides $b$. 
We consider first the use of Gaussian elimination with complete pivoting. Algorithm 2.10 defines elementary permutations \( P_i, Q_i \) \((i = 1, 2, \ldots, n - 1)\), and elementary lower triangular matrices \( M_i \) of index \( i \) such that
\[
A_n = M_{n-1}P_{n-1}M_{n-2} \cdots M_1P_1AQ_1Q_2 \cdots Q_{n-1}
\]
is upper triangular. Equation (4.1) can thus be written in the form
\[
P_1M_1^{-1}P_2M_2^{-1} \cdots P_{n-1}M_{n-1}^{-1}A_nQ_{n-1}Q_{n-2} \cdots Q_1x = b,
\]
and the solution is given by
\[
x = Q_1Q_2 \cdots Q_{n-1}A_n^{-1}M_{n-1}P_{n-1}M_{n-2}P_{n-2} \cdots M_1P_1b.
\]
This suggests the following algorithm for solving (4.1).

**Algorithm 4.1.** Let \( P_i, Q_i, M_i \) \((i = 1, 2, \ldots, n - 1)\) and \( A_n \) be the matrices defined by Gaussian elimination with complete pivoting applied to \( A \). Given the vector \( b \), this algorithm computes the solution \( x \) of (4.1).

1) \( y_1 = b \)
2) For \( k = 1, 2, \ldots, n - 1 \)
   1) \( y_{k+1} = M_kP_ky_k \)
3) Solve the upper triangular system \( A_nz_n = y_n \)
4) For \( k = n - 1, n - 2, \ldots, 1 \)
   1) \( z_k = Q_kz_{k+1} \)
5) \( x = z_1 \)

The only place where the algorithm can break down is in statement 3. However, if \( A \) is nonsingular, then so is \( A_n \), and the solution of \( A_nz_n = y_n \) always exists. Hence, if \( A \) is nonsingular, Algorithm 4.1 can be carried to completion and yields the solution to (4.1).

It is instructive to consider the practical details of the implementation of Algorithm 4.1. In the first place, it is not necessary to store the \( y_i \) and \( z_i \) separately. Rather we can initially set \( x = b \), and store each new vector in \( x \) as it is generated. Second, if Algorithm 2.10 is used to accomplish the Gaussian reduction, the matrices \( P_k, Q_k, \) and \( M_k \) are given in terms of the numbers \( \varrho_k, \gamma_k, \) and \( \mu_{ik} \). Since these matrices are of very simple form, it
would be wasteful of time and storage to generate them in full and perform the matrix-vector multiplications indicated by the algorithm. Instead we may compute one vector directly from its predecessor. For example, statement 2.1 can be accomplished by the sequence of computations

\[ 1) \quad \eta_k^{(k)} \rightarrow \eta_k^{(0)} \]

\[ 2) \quad \eta_i^{(k+1)} = \eta_i^{(k)} - \mu_{ik} \eta_k^{(k)} \quad (i = k + 1, k + 2, \ldots, n) \]

Finally, statement 3 can be accomplished by Algorithm 1.3.

We sum up these considerations by rewriting Algorithm 4.1. In the sequel, we shall describe the use of the other decompositions in the spirit of Algorithm 4.1, leaving the detailed recastings as exercises.

**Algorithm 4.2.** Given the output from Algorithm 2.10 and the \( n \)-vector \( b \), this algorithm computes the solution of (4.1).

1) \( x = b \)

2) For \( k = 1, 2, \ldots, n - 1 \)
   1) \( \xi_k \leftarrow \xi_k \)
   2) \( \xi_i \leftarrow \xi_i - \mu_{ik} \xi_k \quad (i = k + 1, k + 2, \ldots, n) \)

3) \( x \leftarrow A^{-1} x \)

4) For \( k = n - 1, n - 2, \ldots, 1 \)
   1) \( \xi_k \leftarrow \xi_k \)

If Algorithm 1.3 is used to accomplish step 3, Algorithm 4.2 requires about \( n^2 \) multiplications for its execution. This should be compared with the \( n^3/3 \) multiplications required to reduce \( A \) to triangular form. In other words, if we wish to solve a single linear system, the bulk of the work will be concentrated in the initial reduction. Thereafter, additional systems with the same matrix but different right-hand sides can be solved at relatively little additional cost.

Turning now to the use of Gaussian elimination with partial pivoting, we note that the reduction with partial pivoting differs from the reduction with complete pivoting only in the absence of column interchanges. Hence, having performed Gaussian elimination with partial pivoting on \( A \), we may obtain an algorithm for solving (4.1) by deleting step 4 from Algorithm 4.1.
**Algorithm 4.3.** Let $P_i$ and $M_i$ ($i = 1, 2, \ldots, n - 1$), and $A_n$ be the matrices obtained by applying Gaussian elimination with partial pivoting to $A$. Given the $n$-vector $b$, this algorithm computes the solution of (4.1).

1) $x = b$
2) For $k = 1, 2, \ldots, n - 1$
   1) $x \leftarrow M_k P_k x$
3) $x \leftarrow A_n^{-1} x$

The Crout reduction of $A$ with pivoting yields elementary permutations $P_i$ ($i = 1, 2, \ldots, n - 1$) such that

$$P_{n-1} P_{n-2} \cdots P_1 A = LU,$$

where $L$ is lower triangular and $U$ is unit upper triangular. Hence the solution of (4.1) is given by

$$x = U^{-1} L^{-1} P_{n-1} P_{n-2} \cdots P_1 b.$$

This leads to the following algorithm for solving (4.1).

**Algorithm 4.4.** Let $LU$ be the Crout decomposition of $P_{n-1} P_{n-2} \cdots P_1 A$, and let $b$ be given. This algorithm computes the solution of (4.1).

1) $x = b$
2) For $k = 1, 2, \ldots, n - 1$
   1) $x \leftarrow P_k x$
3) $x \leftarrow L^{-1} x$
4) $x \leftarrow U^{-1} x$

The algorithm can be carried to completion if $A$ is nonsingular and requires about $n^2$ multiplications. In statement 4 a minor saving can be effected by taking advantage of the fact that $U$ is unit upper triangular.

When $A$ is symmetric and positive definite, it has a Cholesky decomposition in the form $A = LL^T$, where $L$ is lower triangular. Thus the solution of (4.1) is given by

$$x = L^{-1} L^{-1} b$$

and we obtain the following algorithm.
ALGORITHM 4.5. Let $LL^T$ be the Cholesky decomposition of the positive definite matrix $A$, and let $b$ be given. This algorithm computes the solution of (4.1).

1) $x = L^{-1}b$
2) $x \leftarrow L^{-T}x$

Any of the above decompositions can be used to find the inverse of $A$, either by inverting and multiplying the matrices in the decomposition, or by solving the $n$ linear equations

$$Ax_i = e_i$$

for the columns of the inverse. The details are left as exercises. Again it must be stressed that in applications the inverse of a matrix is seldom required (cf. Section 1).

EXERCISES

1. Devise an efficient algorithm for solving upper-Hessenberg systems (cf. Exercise 2.6).
2. Devise an efficient algorithm for solving tridiagonal systems (cf. Exercise 2.7).
3. Write an INFL program that takes the output of Algorithm 3.4 and overwrites $A$ with $A^{-1}$ by forming the product $U^{-1}L^{-1}$. Be careful of the interchanges.
4. Give detailed INFL code that uses the output of the algorithms of Exercises 2.19, 1.20, 2.21, and 3.12 to solve linear systems.

5. THE EFFECTS OF ROUNDING ERROR

In this section we shall discuss the effects of rounding errors on the algorithms described in this chapter. For example, we shall show that if the algorithms of Section 4 are used to compute a solution of the equation

$$Ax = b,$$  \hspace{1cm} (5.1)

the computed solution $\bar{x}$ satisfies

$$(A + H)\bar{x} = b,$$  \hspace{1cm} (5.2)
and we shall give rigorous bounds on the sizes of the elements of $H$. If $H$
can be shown to be small, then the algorithm is stable in the sense of Section 2.1.

Such an error analysis has two important limitations. In the first place, the error bounds are often far larger than the observed error. There are two reasons for this. First, in order to obtain reasonably simple error bounds one must use estimates that are obviously not sharp. Second, no rigorous upper bound on the error, however sharp, can satisfactorily account for the statistical nature of rounding error. It should not be concluded from this that the error analyses are useless. Even if the upper bound on $H$ in (5.2) is an overestimate, it none the less guarantees the stability of the algorithm, provided it is reasonably small. In addition, an error analysis can suggest how to arrange the details of an algorithm for greater stability. For example, the bound on $H$ in (5.2) contains factors that depend on the pivoting strategy used in the decomposition of $A$ and thereby provides a rationale for choosing a pivoting strategy.

The second limitation is that a stability result such as (5.2) cannot insure the accuracy of the solution, unless the problem is well conditioned, for if the system (5.1) is ill conditioned, even a small random $H$ will correspond to a large deviation in $\mathbf{x}$. However, it may happen that the matrix $H$ that results from rounding error has elements that are so correlated that $\mathbf{x}$ is accurate, even when (5.1) is ill conditioned. Our theorems will say nothing about this phenomenon, since they only bound the size of the elements of $H$. Such correlated errors actually occur in solving triangular systems, whose solutions are usually computed to high accuracy.

A rigorous rounding-error analysis proceeds by repeated applications of the rounding-error bounds of Section 2.1. Although the analyses are usually conceptually straightforward, they are fussy in detail. For this reason we shall only state the results of the rounding-error analyses in this section (however, some of the analyses are given in Appendix 3). We shall assume that all calculations have been performed in $t$-digit floating-point arithmetic satisfying the bounds of Section 2.1. We further assume that underflow and overflow have not occurred in the calculation. Finally, we assume that the order of the problem considered, say $n$, is so restricted that $n \mu \cdot 10^{-t} < .1$. This restriction, which in practice is always satisfied, is necessary to obtain bounds in a reasonably simple form (cf. Exercises 2.1.4 and 2.1.5).

We turn first to the solution of triangular systems by Algorithm 1.3.
THEOREM 5.1. Let $T \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$. Let $\bar{x}$ denote the computed solution of the equation $Tx = b$ obtained from Algorithm 1.3. Then $\bar{x}$ satisfies the equation

$$(T + E)\bar{x} = b,$$

where

$$|\varepsilon_{ij}| \leq (n + 1)\pi |\tau_{ij}| 10^{-t} \quad (i, j = 1, 2, \ldots, n). \quad (5.3)$$

Here $\pi$ is a constant of order unity that depends on the details of the arithmetic.

This theorem is in many respects quite satisfactory. It says that the computed solution is an exact solution of a problem in which $T$ has been perturbed slightly. In fact the elements of $T + E$ differ from the corresponding elements of $T$ by small relative errors. For moderate $n$, these relative errors are comparable to the errors made in rounding $T$ itself, and if the elements of $T$ are themselves computed, the errors $\varepsilon_{ij}$ may be considerably smaller than the uncertainty in $T$. If inner products are accumulated in double precision, the factor $n$ may be dropped from (5.3), so that the error matrix $E$ is quite comparable with the error made in rounding $T$.

None the less, the above results illustrate the limitations mentioned at the first of this section. In the first place, the details of the analysis show that

$$|\varepsilon_{ij}| \leq (j - i + 2)\pi |\tau_{ij}| 10^{-t},$$

which may be considerably smaller than (5.3). More important, though, is the fact that the solutions of triangular systems are usually computed to high accuracy. This fact, which is not a consequence of Theorem 5.1, cannot be proved in general, for counter examples exist. However, it is true of many special kinds of triangular matrices and the phenomenon has been observed in many others. The practical consequences of this fact cannot be over-emphasized.

The question of whether Algorithm 1.4 for inverting an upper triangular matrix is stable is open. Since each column $s_i$ of the inverse was formed by solving the system $Ts_i = e_i$, it follows from Theorem 5.1 that $s_i$ is the $i$th column of the inverse of a slightly perturbed matrix $T + E_i$. However, since the perturbation is different for each $i$, it does not follow that $S = (s_1, s_2, \ldots, s_n)$ is the inverse of some perturbed matrix $T + E$ or even that $S$ is near the inverse of such a matrix.
None the less, in practice the computed inverses of triangular matrices are usually found to be quite accurate. This is to be expected, since the inverse is obtained by solving a set of triangular systems, and we have already observed that the solutions of triangular systems are usually computed with high accuracy. Moreover, it can be shown that for some special classes of matrices the computed inverse must be accurate, although nothing can be proved in general.

We shall now consider the error analyses of Gaussian elimination and the algorithms for triangular decomposition. We have already seen that the strategy for selecting the pivots in these algorithms can have a marked effect on their numerical properties, and we should expect the error analysis to reflect the pivoting strategy. It turns out that the choice of pivots affects the outcome by limiting the growth of elements computed in the course of the reduction.

This point is most clearly illustrated by Gaussian elimination. By Theorem 2.9 we may assume that any interchanges required by the pivoting strategy have already been performed. Thus let the algorithm for Gaussian elimination without pivoting be performed in floating-point arithmetic on the matrix $A = A_1$ yielding matrices $M_1, M_2, \ldots, M_{n-1}$, and $A_1, A_2, \ldots, A_n$. Let

$$\beta_k = \max \{|\alpha_{ij}^{(k)}|\} \quad (k = 1, 2, \ldots, n)$$

and

$$\gamma = \max \{\beta_k : k = 1, 2, \ldots, n\} / \beta_1.$$

Thus $\gamma$, the ratio of the largest element of $A_1, A_2, \ldots, A_n$ to the largest element of $A_1$, is a measure of the growth of the matrices generated by the algorithm. With these definitions, we have the following result.

**THEOREM 5.2.** The matrices $M_1, M_2, \ldots, M_{n-1}, A_n$ computed by Algorithm 2.4 satisfy

$$M_1^{-1}M_2^{-1} \cdots M_{n-1}^{-1}A_n = A + E,$$

(5.4)

where

$$|\varepsilon_{ij}| \leq n\pi\beta_1\gamma 10^{-t}$$

(5.5)

for some constant $\pi$ of order unity that is independent of $A$. 
By the theorem, the computed triangular decomposition obtained from Gaussian elimination is the exact triangular decomposition of a perturbed matrix. The perturbation will be small compared to $\beta_1$ (that is, compared to the largest element of $A$) provided the growth factor $\gamma$ is small. Thus it is important to establish bounds on the growth of the elements in the course of the reduction. For partial and complete pivoting upper bounds can be given for $\gamma$ that do not depend on $A$. The bound for complete pivoting is

$$\gamma \leq \{n \cdot 2^{1/2} \cdot 3^{1/2} \cdot 4^{1/2} \cdots n^{1/(n-1)}\}^{1/2}.$$  

This bound increases rather slowly with $n$, and moreover the proof that establishes it shows that it cannot be attained. In fact, no matrix with real elements is known for which $\gamma$ is greater than $n$. Thus Gaussian elimination with complete pivoting is a stable algorithm.

The bound for partial pivoting is

$$\gamma \leq 2^{n-1}. \quad (5.6)$$

This is a rather fast growing function, and its use in the bound (5.5) suggests that the order of those matrices that can be safely decomposed by Gaussian elimination with partial pivoting is severely limited. For example, on a four-digit machine, if $A$ is of order 12, the elements $E$ may be of the same size as the elements of $A$. Unfortunately matrices are known for which this bound on $\gamma$ is attained, so that we cannot assert that Gaussian elimination with partial pivoting is unconditionally stable.

In practice, however, the bound (5.6) is seldom attained. Usually the elements of the reduced matrices $A_1, A_2, \ldots, A_n$ remain of the same order of magnitude or even show a progressive decrease in size. Moreover, for many commonly occurring kinds of matrices, such as Hessenberg matrices, the growth is much more restricted. Thus in practice Gaussian elimination with partial pivoting must be considered a stable algorithm.

The error analysis of the Crout reduction yields similar results. The computed $L$ and $U$ satisfy

$$LU = A + E$$

where the elements of $E$ satisfy (5.5). The growth factor $\gamma$ in the bounds for the Crout reduction is the same as the growth factor for Gaussian elimination. We have already noted that with partial pivoting $\gamma$ can increase swiftly with the order of the matrix. Since partial pivoting is the only
convenient strategy for the Crout reduction, we cannot claim that this reduction is stable in general. However, the observation that large growth factors are almost never encountered applies here, and for practical purposes the Crout reduction can be considered a stable algorithm.

We have mentioned earlier that an important reason for preferring the Crout reduction to Gaussian elimination is that the Crout reduction permits the accumulation of inner products in double precision. As far as the rounding-error analysis is concerned, the effect of this is to remove the factor \( n \) from the bound (5.5). Thus if \( \gamma \) is also of order unity, the elements of the error matrix \( E \) are of the same size as the errors made in rounding \( A \), and in many cases they will be considerably smaller. This is about as much as can be expected in the way of stability.

The error analysis of the Cholesky algorithm for positive definite matrices differs from the others in that there is no growth factor (this is also true of Gaussian elimination and Crout reduction applied to positive definite matrices; see Exercise 5.2). In fact, we can prove that the algorithm computes a lower triangular matrix \( L \) satisfying

\[
LL^T = A + E,
\]

where the elements of \( E \) satisfy (5.5) with \( \gamma = 1 \). When inner products are accumulated in double precision, the factor \( n \) may be removed from the bounds. Thus the Cholesky algorithm for positive definite matrices is unconditionally stable.

It should be stressed that the results quoted here for Gaussian elimination and the algorithms for triangular decomposition are the final, simplified results of rather detailed error analyses. An examination of the details of the analyses shows that the bounds on the elements of \( E \) are, in many frequently occurring cases, severe overestimates. Nonetheless, the above results indicate the major features that determine the stability of the algorithms.

For example, these bounds have something to say about the delicate question of choosing between Gaussian elimination with complete pivoting, Gaussian elimination with partial pivoting, and the Crout reduction with partial pivoting. When inner products cannot be accumulated in double precision the Crout reduction and Gaussian elimination have the same numerical properties. When inner products can be accumulated in double precision, the Crout reduction is superior. Thus there seems to be little reason for using Gaussian elimination with partial pivoting. This conclusion
is reinforced by the fact that a Crout reduction program that does not accumulate inner products in double precision can be converted into one that does with very little trouble.

The choice is then between Gaussian elimination with complete pivoting and the Crout reduction. A reason customarily advanced against the former is that searching the entire matrix for the largest element is inordinately time consuming. This will generally be true if the algorithm is coded in one of the standard languages such as FORTRAN. However, if the algorithm is coded in machine language, the search can be performed as the elements are computed, usually at little additional cost. Since the stability of Gaussian elimination with complete pivoting is guaranteed, a person of conservative temperament who is willing to put up with a machine dependent subroutine might prefer it over the Crout reduction. On the other hand, practically speaking the Crout reduction has the same stability as Gaussian elimination with complete pivoting and can be coded efficiently in, say, FORTRAN. If inner products can be accumulated in double precision, the Crout reduction is superior, except in the unlikely cases where there is a significant growth in the elements.

Bounds for the solution of the linear system \( Ax = b \) may be obtained by combining Theorem 5.1 and 5.2. For definiteness we shall suppose that the matrix \( A \) has been decomposed by Gaussian elimination with partial pivoting into the product

\[
A = (M_1^{-1}M_2^{-1} \cdots M_n^{-1})A_n = LU.
\]

The elements of \( L \) are the multipliers \( \mu_{ij} \) of Algorithm 2.4, which are less than unity in magnitude. Hence

\[
| \lambda_{ij} | \leq 1.
\]

The elements of \( U \) are taken from the matrices \( A_1, A_2, \ldots, A_n \) of the reduction and hence satisfy

\[
| v_{ij} | \leq \gamma \beta_1,
\]

where \( \beta_1 \) is the magnitude of the largest element of \( A_1 \) and \( \gamma \) is the growth factor. Finally from Theorem 5.2 we know that

\[
LU = A + E,
\]

where the elements of \( E \) satisfy (5.5).
Now suppose that Algorithm 4.3 is used to solve the system $Ax = b$. Statement 2 of the algorithm is numerically equivalent to solving the lower triangular system

$$Ly = b,$$  \hspace{1cm} (5.8)

while statement 3 amounts to solving the upper triangular system

$$Ux = y.$$

We have observed that triangular systems are usually solved with high accuracy. Hence it is not surprising that the computed solution $\tilde{x}$ is often very near the true solution $x$ of the system

$$(A + E)x = b.$$  \hspace{1cm} (5.9)

In other words, the computed solution $\tilde{x}$ of a system of linear equations is often near a solution $x$ of (5.9), where the error matrix $E$ is independent of the right-hand side $b$. However, this is not a rigorous result and counter examples can be constructed. The following theorem shows that if we drop the requirement that the error matrix be independent of the right-hand side, we can obtain a rigorous stability result.

**THEOREM 5.3.** Let $\bar{x}$ denote the solution of the system $Ax = b$ computed by Algorithm 4.3. Then $\bar{x}$ satisfies

$$(A + H)\bar{x} = b,$$  \hspace{1cm} (5.10)

where

$$|\eta_{ij}| \leq (n\pi + 2n^2\varrho + n^3\varrho^3)10^{-t}\gamma\beta_i10^{-t},$$  \hspace{1cm} (5.11)

with $\pi$ and $\varrho$ constants of order unity.

**PROOF.** By Theorem 5.1, the computed solution $\bar{y}$ of (5.8) satisfies

$$(L + F)\bar{y} = b,$$  \hspace{1cm} (5.12)

where

$$|\phi_{ij}| \leq n\varrho |\lambda_{ij}| 10^{-t} \leq n\varrho 10^{-t}$$

with $\varrho$ a constant of order unity. The last inequality follows from the fact that partial pivoting insures that $|\lambda_{ij}| \leq 1$. In statement 3 of Algorithm
4.3 we solve the system $Ux = \bar{y}$, yielding a computed solution $\bar{x}$ that satisfies

$$(U + G)\bar{x} = \bar{y}$$

(5.13)

where

$$|\gamma_{ij}| \leq n\eta |v_{ij}| 10^{-t} \leq n\eta \beta \gamma 10^{-t}.$$ 

Combining (5.7), (5.12), and (5.13), we find that $\bar{x}$ satisfies (5.10) where

$$H = E + LG + FU + FG.$$ 

Now the largest element of the product of two matrices of order $n$ is bounded by $n$ times the product of the largest elements of each matrix. Hence, using the bounds for the elements of $E$, $F$, $G$, $L$, and $U$, we obtain (5.11).

The error matrix $H$ of course depends on the vector $b$, since the matrices $F$ and $G$ do. The bound (5.11) on its elements depends critically on the growth factor $\gamma$. If $\gamma$ is of order unity and $n$ is not unreasonably large, the elements of $H$ will be not too much larger than the errors made in rounding $A$; often they will be a good deal smaller. The term containing $n^2$ comes from the errors made in solving triangular systems. Since, as we have noted, these errors are often negligible, the true error may be more accurately reflected by the term $n\pi$.

Analogs of Theorem 5.3 hold for the Crout and Cholesky algorithms. If inner products are accumulated in double precision throughout the calculations, the first factor in (5.11) may be replaced by

$$\pi + 2n\eta + n\eta^2 10^{-t}.$$ 

In many cases the term $2n\eta$, which accounts for the errors introduced by solving the triangular systems, may be omitted, so that the bound becomes effectively independent of the size of the matrix.

Since no general theorems on the computed inverses of triangular equations have been established, it is not surprising that little can be said about the computed inverses of general matrices. Suppose, for example, that the columns of the inverse $X$ are computed as the solutions of the systems $Ax_i = e_i$ $(i = 1, 2, \ldots, n)$. Then from Theorem 5.3 we know that each computed column $\bar{x_i}$ satisfies $(A + H_i)\bar{x_i} = e_i$; however, the error matrices $H_i$ may be different for each column, and we cannot conclude that there is a single small matrix $H$ such that $(A + H)\bar{X} = I$. 


None the less, it follows from the remarks preceding Theorem 5.3 that each \( \bar{x}_i \) will often be near the solution \( \bar{x}_i \) of the equation \((A + E)\bar{x}_i = e_i\), where \( E \) is the error matrix from the \(LU\) decomposition of \( A\). Since \( E \) is independent of the right-hand side \( e_i \), it follows that \( \bar{x} \) is near an inverse of \( A + E \). This means that in many cases the error in the computed inverse of \( A \) will be almost entirely due to the errors made in computing the \(LU\) decomposition of \( A\).

All of the rigorous results established in this section concern the stability of algorithms. The only results on the accuracy of computed solutions were in the form of informal observations. In Section 5.4 we will apply the theory of norms to obtain rigorous bounds on the errors in computed solutions.

We conclude this section with a word of caution. The factor \( \beta_1 \), which appears in the error bound of Theorem 5.2, serves the purpose of a normalizing factor. In effect the bound says that the error will be small compared to \( \beta_1 \), that is compared to the largest element of \( A \). If all the elements of \( A \) are of about the same size, then this is a perfectly satisfactory result. On the other hand, if there is a wide disparity in the sizes of the elements of \( A \), there is a danger that the smaller elements will be overwhelmed by the errors. If these small elements are critical to the reduction, the results will be inaccurate.

For example, consider the matrix

\[
A = \begin{pmatrix}
3.000 & 6000. & 9000. \\
-1.000 & 3.712 & 4.623 \\
-2.000 & 1.072 & 5.643
\end{pmatrix},
\]

(5.14)

which was obtained from the matrix of Example 2.8 by multiplying its first row by 3000. If Gaussian elimination with partial pivoting is performed in four-digit arithmetic on \( A \), the \((1, 1)\)-element is accepted as the first pivot, and the calculation proceeds exactly as in Example 2.8 with a disastrous cancellation occurring in the computation of the \((3, 3)\)-element of the reduced matrix. Our error analysis guarantees that the computed decomposition is the exact decomposition of \( A + E \), where the elements of \( E \) are of the order \( \beta_1 \cdot 10^{-4} \); however, this is small consolation, for \( \beta_1 = 9000 \) and the elements of \( E \) can be, and are, as large as unity.

Since the disparity in the sizes of the elements of \( A \) is responsible for the problem, it is natural to attempt to scale the rows and columns of \( A \) so that the matrix is balanced. A frequently suggested method is to scale \( A \) so that the largest element in each row and column lies between, say, 1 and 10.
For example, the original matrix of Example 2.8, which can be reduced safely by Gaussian elimination with partial pivoting, satisfies this criterion. However, the matrix

\[
A = \begin{pmatrix}
3.000 & 1.000 & 1.000 \\
-1.000 & 6.187 \times 10^{-4} & 5.137 \times 10^{-4} \\
-2.000 & 1.787 \times 10^{-4} & 6.270 \times 10^{-4}
\end{pmatrix},
\]

obtained from (5.14) by dividing the second column by 6000 and the third column by 9000 also satisfies the criterion. It is easily verified that, in four-digit arithmetic, even complete pivoting will not yield an accurate reduction. The error matrix associated with the reduction has elements of order 10^{-4}, but this is no help since four of the elements of \( A \) are also of order 10^{-4}.

The difficulties involved in scaling can be overstressed. For many problems the criterion suggested above is quite satisfactory and for many others the correct scaling is perfectly evident. However, the above example shows that pat scaling strategies are suspect. In spite of intensive theoretical investigation, there is no satisfactory algorithm for scaling a general matrix.

**EXERCISES**

1. Compute the last column of the inverse of

\[
\begin{pmatrix}
10^{-4} & .9 & -.4 \\
0 & .9 & -.4 \\
0 & 0 & 10^{-4}
\end{pmatrix}
\]

in four-digit arithmetic. Interpret your results.

2. Show that if Gaussian elimination is applied to a positive definite matrix, then the growth factor \( \gamma \) of Theorem 5.2 is equal to unity. *[Hint: Show that the diagonal elements must decrease and then apply Exercises 3.8 and 3.11.]*

3. Show that if \( A \) is upper Hessenberg of order \( n \), then the growth factor \( \gamma \) for Gaussian elimination with partial pivoting is bounded by \( n \).

4. Show that if \( A \) is tridiagonal, then the growth factor \( \gamma \) for Gaussian elimination with partial pivoting is bounded by two.

5. Show that if \( A^T \) is diagonally dominant, then the growth factor \( \gamma \) for Gaussian elimination with partial pivoting is bounded by two.
NOTES AND REFERENCES

All the material in this section is contained in works by Wilkinson (1961; 1963; AEP, Chapter IV). The 1961 paper contains a wealth of detail: in particular, a derivation of bounds for the growth factors for matrices of special form and a discussion of the high accuracy of the computed solutions of triangular systems.

The conjecture that the growth factor for complete pivoting is bounded by the order of the matrix has neither been proved nor disproved. Cryer (1968) proves the conjecture for matrices of order four and surveys other attempts.

For more on the problem of scaling or *equilibration*, as it is sometimes called, see the works by Bauer (1963), Forsythe and Strauss (1955), and Wilkinson (1961; AEP, Chapter IV).
With few exceptions we have, in the first three chapters of this book, confined ourselves to the algebraic properties of vectors and matrices; that is we have developed those areas that could be described in terms of algebraic operations without requiring the notion of limit. In this chapter we shall introduce the ideas of norm and limit and apply them to the problems of assessing and improving the accuracy of approximate solutions of linear systems.

A vector norm on \( \mathbb{R}^n \) is a function from \( \mathbb{R}^n \) into the nonnegative real numbers whose value in some sense measures the size of a vector in \( \mathbb{R}^n \). Examples of norms are the absolute value in \( \mathbb{R} \) and the Euclidean length of a vector in \( \mathbb{R}^3 \). The idea of norm is closely related to the idea of limit, as is suggested by the equivalence

\[
\lim_{k \to \infty} \xi_k = \xi \quad \iff \quad \lim_{k \to \infty} |\xi - \xi_k| = 0.
\]

In Section 1 we shall introduce the idea of a vector norm on \( \mathbb{R}^n \) and explore its relation to limits in \( \mathbb{R}^n \). In Section 2 we shall extend the concept of norm to matrices. In Section 3.5, we discussed the effects of rounding
error on some algorithms for solving linear systems. However, our formal results gave no indication of the accuracy of the solution; rather they implied that the errors in the solution would be proportional to the degree of ill conditioning of the system. In Sections 3 and 4 we shall use the theory of norms to define a condition number for a linear system whose size measures the effects of perturbations in the matrix on the final solution. Finally in Section 5, we shall analyze a method for refining an approximate solution of a linear system to produce a more accurate one.

1. NORMS AND LIMITS

The idea of a vector norm arises from the attempt to generalize the idea of the length of a vector in \( \mathbb{R}^1 \) or \( \mathbb{R}^2 \). The simplest example of a norm is the absolute value function, which measures the distance of a scalar from the origin. The absolute value \( | \xi | \) of a number \( \xi \) may be defined by the equation

\[
| \xi | = \sqrt{\xi^2}
\]

and has the three well-known properties

1. \( \xi \neq 0 \implies | \xi | > 0 \),
2. \( | \alpha \xi | = | \alpha | | \xi | \),
3. \( | \xi + \eta | \leq | \xi | + | \eta | \).

Because of (1.1.1), the absolute value is said to be positive definite, or simply definite. Because of (1.1.2), it is said to be homogeneous. The third condition is called the triangle inequality for reasons that will become clear in a moment.

![Fig. 1](image-url)
A less trivial example of a norm is the Euclidean length of a vector in $\mathbb{R}^2$. For any $x \in \mathbb{R}^2$ it is denoted by $\| x \|_2$ and defined by

$$\| x \|_2 = \sqrt{x_1^2 + x_2^2}. \quad (1.2)$$

By the Pythagorean theorem, $\| x \|_2$ is the distance from $x$ to the origin (Fig. 1).

The Euclidean norm $\| \cdot \|_2$ satisfies three conditions that correspond to the properties (1.1) satisfied by the absolute value. Specifically,

1. $x \neq 0 \Rightarrow \| x \|_2 > 0$,
2. $\| \alpha x \|_2 = |\alpha| \| x \|_2$,
3. $\| x + y \|_2 \leq \| x \|_2 + \| y \|_2$.

The first two conditions are clear from the definition of $\| \cdot \|_2$. The third condition may be interpreted geometrically as saying that the length of a side of a triangle is not greater than the sum of the lengths of the other two sides, whence the name triangle inequality (Fig. 2).

![Fig. 2](image)

One natural way of extending these ideas to $\mathbb{R}^n$ is to generalize formula (1.2) in the obvious way to obtain a Euclidean length of a vector in $\mathbb{R}^n$. However, this is somewhat restrictive, for there are many other functions that measure the size of a vector. For example, the number $v(x)$ defined for $x \in \mathbb{R}^n$ by

$$v(x) = \max \{ |x_1|, |x_2| \} \quad (1.3)$$

is also a reasonable measure of the size of $x$ and is computationally more convenient than (1.2). It turns out that in many applications, all that is required of a measure of size is that it be definite, homogeneous, and satisfy the triangle inequality. This motivates the following definition.
DEFINITION 1.1. A vector norm (or simply a norm) on \( \mathbb{R}^n \) is a function \( \nu : \mathbb{R}^n \to \mathbb{R} \) that satisfies the following conditions:

1. \( x \neq 0 \Rightarrow \nu(x) > 0, \)
2. \( \nu(\alpha x) = |\alpha| \nu(x), \)
3. \( \nu(x + y) \leq \nu(x) + \nu(y). \)

The absolute value function is a norm on \( \mathbb{R}^1 \), and the function \( \| \cdot \|_2 \) defined by (1.2) is a norm on \( \mathbb{R}^2 \). The function \( \nu \) defined by (1.3) is also a norm on \( \mathbb{R}^2 \).

The definition of norm has some elementary consequences. By property 1.1.1, the function \( \nu \) is positive except at the zero vector. By property 1.1.2,

\[
\nu(0) = \nu(0 \cdot x) = 0 \nu(x) = 0,
\]

so that the norm of the zero vector is zero. Also from 1.1.2, it follows that

\[
\nu(-x) = \nu(-1 \cdot x) = |-1| \nu(x) = \nu(x).
\]

An important inequality is the following.

\[
|\nu(x) - \nu(y)| \leq \nu(x - y), \tag{1.4}
\]

which in terms of the Euclidean norm says that the length of a side of a triangle is not less than the difference of the lengths of the other two sides (Fig. 3). To establish it, note that

\[
\nu(x) = \nu[(x - y) + y] \leq \nu(x - y) + \nu(y).
\]

Hence

\[
\nu(x) - \nu(y) \leq \nu(x - y). \tag{1.5}
\]
If $x$ and $y$ are interchanged in (1.5), we obtain
\[ v(y) - v(x) \leq v(y - x) = v(x - y). \]  
\hspace{1cm} (1.6)

Inequalities (1.5) and (1.6) together imply (1.4).

If $y$ is replaced by $-y$ in (1.4), there results the related inequality
\[ |v(x) - v(y)| \leq v(x + y). \]

We turn now to the construction of specific vector norms. We begin with three norms on $\mathbb{R}^n$ that are frequently used in analyzing matrix processes. They are the 1-, 2-, and $\infty$-norms defined by
\[ \|x\|_1 = \sum_{i=1}^{n} |\xi_i|, \]
\[ \|x\|_2 = \sqrt{\sum_{i=1}^{n} \xi_i^2} = \sqrt{x^Tx}, \]
and
\[ \|x\|_\infty = \max \{ |\xi_i| : i = 1, 2, \ldots, n \}. \]

The 2-norm is the natural generalization of the Euclidean length of a 2- or 3-vector and is also called the Euclidean norm. The $\infty$-norm is sometimes called the maximum norm (max-norm) or the Chebyshev norm.

All three functions $\| \cdot \|_1$, $\| \cdot \|_2$, and $\| \cdot \|_\infty$ are obviously definite and homogeneous. Moreover, it is easy to show that $\| \cdot \|_1$ and $\| \cdot \|_\infty$ satisfy the triangle inequality. For example,
\[ \|x + y\|_1 = \sum_{i=1}^{n} |\xi_i + \eta_i| \leq \sum_{i=1}^{n} (|\xi_i| + |\eta_i|) \]
\[ = \sum_{i=1}^{n} |\xi_i| + \sum_{i=1}^{n} |\eta_i| = \|x\|_1 + \|y\|_1. \]

To establish the triangle inequality for the 2-norm we require an inequality which is important in its own right. It is the well-known Cauchy inequality (which, incidentally, is also associated with the names Schwarz and Bunyakovski).
1. NORMS AND LIMITS

THEOREM 1.2. For all $x, y \in \mathbb{R}^n$,

$$|x^T y| \leq \|x\|_2 \|y\|_2,$$

with equality if and only if $x$ and $y$ are linearly dependent.

PROOF. If either $x$ or $y$ is zero, the theorem is trivially true. Hence suppose that both $x$ and $y$ are nonzero. Then $\|x\|_2$ and $\|y\|_2$ are both nonzero. Let $x' = x/\|x\|_2$ and $y' = y/\|y\|_2$. Then $\|x'\|_2 = \|y'\|_2 = 1$, and the inequality (1.7) is equivalent to the inequality

$$|x'^T y'| \leq 1.$$ 

Now suppose $x'^T y' \geq 0$. Then

$$0 \leq \|x' - y'\|_2^2 = (x' - y')^T (x' - y')$$
$$= x'^T y' + y'^T y' - 2x'^T y'$$
$$= \|x'\|_2^2 + \|y'\|_2^2 - 2x'^T y'$$
$$= 2 - 2x'^T y'.$$

Hence $x'^T y' \leq 1$. If $x'^T y' < 0$, the inequality $-x'^T y' \leq 1$ follows similarly, with $x' + y'$ replacing $x' - y'$ in (1.8).

Now suppose $|x^T y| = \|x\|_2 \|y\|_2$ with say $x^T y \geq 0$. This can only happen when the inequality in (1.8) is an equality. Hence $\|x' - y'\|_2 = 0$, and $x' = y'$. This implies that

$$x = \frac{\|x\|_2}{\|y\|_2} y$$

and hence that $x$ and $y$ are linearly dependent. The case $x^T y < 0$ is treated similarly.

Finally, suppose that $x$ and $y$ are linearly dependent. Since $y \neq 0$, $x = \alpha y$ for some scalar $\alpha$. Then $\|x\|_2 = |\alpha| \|y\|_2$, and

$$|x^T y| = |\alpha| y^T y = |\alpha| \|y\|_2 \|y\|_2 = \|x\|_2 \|y\|_2.$$

The above proof makes heavy use of the identity $\|x\|_2^2 = x^T x$. This characterization is also used in establishing that $\|\cdot\|_2$ is a norm.

THEOREM 1.3. The function $\|\cdot\|_2$ satisfies the triangle inequality.
4. NORMS, LIMITS, AND CONDITION NUMBERS

PROOF. The proof is purely computational.

\[
\| x + y \|_2^2 = (x + y)^T(x + y) \\
= x^T x + 2 x^T y + y^T y \\
\leq \| x \|_2^2 + 2 \| x \|_2 \| y \|_2 + \| y \|_2^2 \\
= (\| x \|_2 + \| y \|_2)^2,
\]

where the inequality follows from the Cauchy inequality. ■

We have used the same symbol \( \| \cdot \|_2 \) to denote the 2-norm on \( \mathbb{R}^1, \mathbb{R}^2, \mathbb{R}^3, \ldots \). While this is not a serious abuse of notation, it can be avoided by introducing some terminology which will prove useful later in connection with matrix norms.

**Definition 1.4.** Let \( v : \bigcup_{n=1}^{\infty} \mathbb{R}^n \rightarrow \mathbb{R} \). Then \( v \) is a family of vector norms if for each \( n = 1, 2, 3, \ldots \) the restriction of \( v \) to \( \mathbb{R}^n \) is a norm.

Thus each of the functions \( \| \cdot \|_1, \| \cdot \|_2, \| \cdot \|_{\infty} \) is a family of vector norms. Incidentally, these norms are special cases of the Hölder norms or \( p \)-norms defined by

\[
\| x \|_p = \left( \sum_{i=1}^{n} | x_i |^p \right)^{1/p}, \quad 1 \leq p < \infty.
\]

(\( \| x \|_{\infty} \) is \( \lim_{p \rightarrow \infty} \| x \|_p \).) A proof that these are indeed norms will be found in the exercises.

A useful technique for constructing new norms from old is contained in the following theorem.

**Theorem 1.5.** Let \( v \) be a norm on \( \mathbb{R}^m \) and \( A \in \mathbb{R}^{m \times n} \) have linearly independent columns. Then the function \( \mu : \mathbb{R}^n \rightarrow \mathbb{R} \) defined by

\[
\mu(x) = v(Ax)
\]

is a norm on \( \mathbb{R}^n \).

**Proof.** Because \( A \) has linearly independent columns and \( v \) is a norm,

\[
x \neq 0 \Rightarrow Ax \neq 0 \Rightarrow v(Ax) > 0 \Rightarrow \mu(x) > 0.
\]
Hence \( \mu \) is definite. Since

\[
\mu(\alpha x) = v(\alpha Ax) = |\alpha| v(Ax) = |\alpha| \mu(x),
\]

\( \mu \) is homogeneous. Finally,

\[
\mu(x + y) = v[A(x + y)] = v(Ax + Ay) \leq v(Ax) + v(Ay) = \mu(x) + \mu(y),
\]

so that \( \mu \) satisfies the triangle inequality. ■

Theorem 1.5 may be applied to give an important class of norms.

**THEOREM 1.6.** Let \( A \in \mathbb{R}^{n \times n} \) be positive definite. Then the function \( v : \mathbb{R}^n \rightarrow \mathbb{R} \) defined by

\[
v(x) = \sqrt{x^T Ax}
\]

is a norm on \( \mathbb{R}^n \).

**PROOF.** By Theorem 3.3.8, \( A \) can be written in the form \( A = LL^T \), where \( L \) is a nonsingular lower triangular matrix. Then

\[
v(x) = \sqrt{(L^T x)^T (L^T x)} = \|L^T x\|_2.
\]

Hence by Theorem 1.5, \( v \) is a norm. ■

Note that when \( A = I \), the function \( v \) of Theorem 1.6 reduces to the 2-norm.

The notions of norm and limit are closely connected, and in more abstract settings it is often convenient to define the notion of limit in terms of a norm. However, in \( \mathbb{R}^n \) there is a very natural definition of limit in terms of limits of real numbers. We shall start with this definition and explore its connection with vector norms later (Theorem 1.12). We begin with a review of the properties of limits of real numbers.

We shall denote a sequence \( \alpha_1, \alpha_2, \alpha_3, \ldots \) by \( \langle \alpha_k \rangle \). The \( \epsilon - N \) definition of the limit of a sequence of real numbers may be found in almost any calculus text. For our purposes, we need only the elementary properties of the limit. Specifically, the sequence \( \langle \alpha_k \rangle \) can have at most one limit. If \( \langle \alpha_k \rangle \) has the limit \( \alpha \), we write

\[
\alpha = \lim_{k \to \infty} \alpha_k,
\]
or, when it will cause no confusion,

\[ \alpha = \lim \alpha_k. \]

If \( \alpha = \lim \alpha_k \) and \( \beta = \lim \beta_k \), then

\[ \lim (\alpha_k \pm \beta_k) = \alpha \pm \beta \]

(1.9)

and

\[ \lim \alpha_k \beta_k = \alpha \beta. \]

Moreover, if \( \alpha \neq 0 \), then, for all sufficiently large \( k \), \( \alpha_k \neq 0 \) and

\[ \lim \frac{\beta_k}{\alpha_k} = \frac{\beta}{\alpha}. \]

Finally we have the characterization

\[ \lim \alpha_k = \alpha \iff \lim |\alpha - \alpha_k| = 0. \]

(1.10)

In discussing limits of vectors, we shall need to refer to the components of a sequence of vectors. To do this we shall adopt the following convention. If \( \langle x_k \rangle \) is a sequence of \( n \)-vectors, we shall denote the \( i \)th component of \( x_k \) by \( x_{ik}^{(k)} \). We have already used this convention in Section 3.2 in connection with the matrices \( A_k \) generated by Gaussian elimination.

In \( \mathbb{R}^n \) it is natural to say that a sequence of vectors converges to a limit of each if its components converges to a limit. For example, in \( \mathbb{R}^2 \) we should say that the sequence of vectors

\[
\begin{pmatrix} 2.1 \\ 0.9 \end{pmatrix}, \quad \begin{pmatrix} 2.01 \\ 0.99 \end{pmatrix}, \quad \begin{pmatrix} 2.001 \\ 0.999 \end{pmatrix}, \quad \ldots, \quad \begin{pmatrix} 2 \pm 10^{-k} \\ 1 \mp 10^{-k} \end{pmatrix}, \quad \ldots
\]

is converging to the vector \( (2, 1)^T \). On the other hand, we should say that the sequence

\[
\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \quad \ldots, \quad \begin{pmatrix} 1 \\ (-1)^{k+2} \end{pmatrix}, \quad \ldots
\]

has no limit, since the second component never settles down. This suggests the following definition.
DEFINITION 1.7. Let \( \langle x_k \rangle \) be a sequence of \( n \)-vectors and let \( x \in \mathbb{R}^n \). Then \( x \) is a \textit{limit} of the sequence \( \langle x_k \rangle \) (written \( x = \lim_{k \to \infty} x_k \)) if
\[
\lim_{k \to \infty} x_i^{(k)} = x_i \quad (i = 1, 2, \ldots, n).
\]

Since the limit of a sequence of real numbers is unique, Definition 1.7 implies that the limit of a sequence of vectors must be unique. Moreover, the vector limit has some of the algebraic properties of the scalar limit, as the following theorem shows.

THEOREM 1.8. Let the sequences \( \langle a_k \rangle, \langle b_k \rangle, \) and \( \langle \lambda_k \rangle \) have limits \( a, b, \) and \( \lambda \). Then
\[
\lim (a_k \pm b_k) = a \pm b \quad (1.11)
\]
and
\[
\lim \lambda_k a_k = \lambda a. \quad (1.12)
\]

PROOF. We prove (1.11), leaving (1.12) as an exercise. By Definition 1.7, we know that \( \lim a_i^{(k)} = a_i \) and \( \lim b_i^{(k)} = b_i \), and we must show that
\[
\lim_{k \to \infty} (a_i^{(k)} \pm b_i^{(k)}) = a_i \pm b_i \quad (i = 1, 2, \ldots, n).
\]
However, for each \( i \), this is simply (1.9).

The relation between limits and norms is suggested by the equivalence (1.10). Namely, if \( v \) is a norm on \( \mathbb{R}^n \) and \( \langle x_k \rangle \) is a sequence of \( n \)-vectors, then \( x_k = x \) if and only if \( \lim v(x - x_k) = 0 \). However, the proof of this fact requires a fairly deep result on the equivalence of norms. To illustrate its statement, consider the following scalar inequalities:
\[
\max \{ |x_i| : i = 1, 2, \ldots, n \} \leq \sum_{i=1}^{n} |x_i| \leq n \max \{ |x_i| : i = 1, 2, \ldots, n \}.
\]

In terms of norms, the inequalities state that in \( \mathbb{R}^n \)
\[
1 \cdot \| x \|_{\infty} \leq \| x \|_1 \leq n \| x \|_{\infty}.
\]
Thus whenever \( \| \cdot \|_1 \) appears in an expression as an upper bound, we may replace it by \( n \cdot \| \cdot \|_{\infty} \). Whenever \( \| \cdot \|_1 \) appears as a lower bound we may replace it with \( 1 \cdot \| \cdot \|_{\infty} \). In other words, by multiplying the \( \infty \)-norm by
suitable constants, we can make it equivalent to the $l$-norm, at least for the purpose of stating upper and lower bounds. The following theorem, which we give without proof, states that such an equivalence obtains for any two norms.

**Theorem 1.9.** Let $\mu$ and $\nu$ be norms on $\mathbb{R}^n$. Then there are positive constants $\sigma_{\mu\nu}$ and $\tau_{\mu\nu}$ such that for all $x \in \mathbb{R}^n$

$$\sigma_{\mu\nu}(x) \leq \nu(x) \leq \tau_{\mu\nu}(x).$$

Moreover, $\sigma_{\mu\nu}$ and $\tau_{\mu\nu}$ may be chosen so that for some nonzero $y \in \mathbb{R}^n$ we have $\sigma_{\mu\nu}(y) = \nu(y)$ and for some nonzero $z \in \mathbb{R}^n$ we have $\tau_{\mu\nu}(z) = \nu(z)$.

**Example 1.10.** The following tables give the constants $\tau$ and $\sigma$ for the $p$-norms ($p = 1, 2, \infty$) or $\mathbb{R}^n$.

<table>
<thead>
<tr>
<th></th>
<th>$\sigma_{pp}$</th>
<th>$\tau_{pp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>$q$</td>
<td>$1$</td>
</tr>
<tr>
<td>$1$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$2$</td>
<td>$\frac{n^{-1/2}}{n}$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\infty$</td>
<td>$\frac{n^{-1}}{n}$</td>
<td>$\frac{n^{-1/2}}{n}$</td>
</tr>
</tbody>
</table>

Note that in Example 1.10, $\sigma_{pp} = \tau_{pp}^{-1}$. This relation holds for any two norms $\mu$ and $\nu$. In fact $\sigma_{\mu\nu}$ is the largest constant for which

$$\sigma_{\mu\nu}(x) \leq \nu(x), \quad x \in \mathbb{R}^n,$$

and $\tau_{\mu\nu}$ is the smallest constant for which

$$\mu(x) \leq \tau_{\mu\nu}(x), \quad x \in \mathbb{R}^n.$$

Hence

$$\sigma_{\mu\nu} = \tau_{\mu\nu}^{-1}.$$

We are now in a position to establish the relation between norms and limits. We first establish it for the $\infty$-norm.
**LEMMA 1.1.** Let \( \langle x_k \rangle \) be a sequence of \( n \)-vectors and \( x \in \mathbb{R}^n \). Then
\[
\lim x_k = x \iff \lim \| x - x_k \|_\infty = 0.
\]

**PROOF.** From Definition 1.7 and the equivalence (1.10) we have the following equivalences.
\[
\begin{align*}
\lim x_k = x & \iff \lim \xi_i^{(k)} = \xi_i \quad (i = 1, 2, \ldots, n) \\
& \iff \lim |\xi_i - \xi_i^{(k)}| = 0 \quad (i = 1, 2, \ldots, n) \\
& \iff \lim \max \{|\xi_i - \xi_i^{(k)}| : i = 1, 2, \ldots, n\} = 0 \\
& \iff \lim \| x - x_k \|_\infty = 0.
\end{align*}
\]

**THEOREM 1.12.** Let \( v \) be a norm on \( \mathbb{R}^n \) and \( x, x_1, x_2, x_3, \ldots \in \mathbb{R}^n \). Then
\[
\lim_{k \to \infty} x_k = x \iff \lim_{k \to \infty} v(x - x_k) = 0. \tag{1.13}
\]

**PROOF.** If \( \lim x_k = x \), by Lemma 1.11 we have \( \lim \| x - x_k \|_\infty = 0 \). However, by Theorem 1.10, \( v(x - x_k) \leq \tau_{\infty} \| x - x_k \|_\infty \). Hence \( \lim v(x - x_k) = 0 \). Conversely, let \( \lim v(x - x_k) = 0 \). Then by Theorem 1.10, \( \| x - x_k \|_\infty \leq \tau_{\infty} v(x - x_k) \), and hence \( \lim \| x - x_k \|_\infty = 0 \). By Lemma 1.11, we have \( \lim x_k = x \). □

The importance of Theorem 1.12 is that it allows us to use the right-hand side of (1.13) as an alternate definition of limit. Since the norm \( v \) is completely arbitrary, it may be chosen, say, to make it easy to show that \( \lim v(x - x_k) = 0 \). However it is chosen, though, we are assured that if \( \lim v(x - x_k) = 0 \), then \( \lim x_k = x \) in the sense of Definition 1.7.

**EXERCISES**

1. Show that if \( v : \mathbb{R}^1 \to \mathbb{R} \) is a norm, then \( v([\xi]) = \lambda |\xi| \), for some \( \lambda > 0 \).

2. Show that the function \( v \) defined by Equation (1.3) is a vector norm.

3. Prove the *polarization identity*
\[
x^T y = \frac{\| x + y \|^2 - \| x - y \|^2}{4}.
\]
4. Show that \( \| \cdot \|_\infty \) satisfies the triangle inequality.

5. Show by example that the condition that \( A \) have linearly independent columns cannot be removed from Theorem 1.5.

6. Let \( \alpha_1, \alpha_2, \ldots, \alpha_n \) be positive. Show that the function \( v : \mathbb{R}^n \to \mathbb{R} \) defined by \( v(x) = (\sum_{i=1}^{n} \alpha_i x_i^2)^{1/2} \) is a norm.

7. Find linearly independent vectors \( x \) and \( y \) such that \( \| x + y \|_\infty = \| x \|_\infty + \| y \|_\infty \).

8. Show that \( \| x + y \|_2 = \| x \|_2 + \| y \|_2 \) if and only if \( x \) and \( y \) are linearly dependent and \( x^T y \geq 0 \).

9. Let \( v \) be a norm on \( \mathbb{R}^n \) and let \( \mathcal{B}_v = \{ x : v(x) \leq 1 \} \). Prove that \( \mathcal{B}_v \) satisfies the following conditions.
   1. There is an \( \epsilon > 0 \) such that \( \{ x : \| x \|_2 \leq \epsilon \} \subset \mathcal{B}_v \).
   2. (Convexity) If \( x, y \in \mathcal{B}_v \) and \( \alpha, \beta \) are nonnegative scalars with \( \alpha + \beta = 1 \), then \( \alpha x + \beta y \in \mathcal{B}_v \).
   3. (Equilibration) If \( x \in \mathcal{B}_v \) and \( | \alpha | \leq 1 \), then \( \alpha x \in \mathcal{B}_v \).

The set \( \mathcal{B}_v \) is called the "unit v-ball."

10. Describe the unit 1-, 2-, and \( \infty \)-balls in \( \mathbb{R}^2 \); in \( \mathbb{R}^3 \).

11. Establish the bounds of Example 1.10. For each \( \sigma \) and \( \tau \) give the vectors \( y \) and \( z \) promised in Theorem 1.9.

12. Prove that \( \| x \|_\infty = \lim_{p \to \infty} \| x \|_p \).

13. Show that if \( p, q > 1 \) with \( p^{-1} + q^{-1} = 1 \), then \( | \alpha \beta | \leq | \alpha |^{p/p} + | \beta |^{q/q} \), for all scalars \( \alpha \) and \( \beta \). [Hint: Show that the function \( \phi(\tau) = \tau^{p/p} + \tau^{-q/q} \) satisfies \( \phi(\tau) \geq 1 \) for all positive \( \tau \). Let \( \tau = | \alpha |^{1/q} | \beta |^{-1/p} \).]

14. (Hölder inequality) Prove that if \( p, q > 1 \) with \( p^{-1} + q^{-1} = 1 \) and \( x, y \in \mathbb{R}^n \), then \( | x^T y | \leq \| x \|_p \| y \|_q \). [Hint: Assume \( \| x \|_p = \| y \|_q = 1 \), and apply Exercise 3 to \( | \xi_i | \| \eta_i | \). Then sum.]

15. (Minkowski inequality) Let \( x, y \in \mathbb{R}^n \). Show that for \( 1 < p < \infty \) that \( \| x + y \|_p \leq \| x \|_p + \| y \|_p \). [Hint: \( \| x + y \|_p \leq (\| x \|_p + \| y \|_p)(\sum_{i=1}^{n} | \xi_i + \eta_i |^{q(p-1)})^{1/q} \), where \( p^{-1} + q^{-1} = 1 \).]
2. MATRIX NORMS

In this section we shall consider the problem of extending the idea of a vector norm to matrices. At first glance this does not seem to be a very difficult problem. We have seen (page 32) that the set of matrices $\mathbb{R}^{m \times n}$ is a vector space which is essentially identical with $\mathbb{R}^{mn}$. Consequently any vector norm on $\mathbb{R}^{mn}$ induces a definite, homogeneous function on $\mathbb{R}^{m \times n}$ that satisfies the triangle inequality, and it is natural to call such a function a matrix norm. For example, the 2-norm on $\mathbb{R}^{mn}$ induces the Frobenius norm $\| \cdot \|_F$ on $\mathbb{R}^{m \times n}$ defined by

$$
\| A \|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}.
$$

(2.1)

The same proof that $\| \cdot \|_2$ is a vector norm shows that $\| \cdot \|_F : \mathbb{R}^{m \times n} \to \mathbb{R}$ is a definite, homogeneous function that satisfies the triangle inequality

$$
\| A + B \|_F \leq \| A \|_F + \| B \|_F.
$$

We shall adopt this general definition of matrix norm. However, most matrix norms defined in this way are not very useful. The reason is that this definition of matrix norm gives no relation between the norms of two matrices and the norm of their product. Consequently we shall restrict our attention to matrix norms satisfying a consistency condition, which for the Frobenius norm reads

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

(2.2)

We begin our development with the general definition of matrix norm.

**DEFINITION 2.1.** A function $\nu : \mathbb{R}^{m \times n} \to \mathbb{R}$ is a matrix norm on $\mathbb{R}^{m \times n}$ if

1. $A \neq 0 \Rightarrow \nu(A) > 0, \quad A \in \mathbb{R}^{m \times n},$
2. $\nu(\alpha A) = |\alpha| \nu(A), \quad A \in \mathbb{R}^{m \times n}, \quad \alpha \in \mathbb{R},$
3. $\nu(A + B) \leq \nu(A) + \nu(B), \quad A, B \in \mathbb{R}^{m \times n}.$

This definition has some immediate consequences. Since a matrix norm on $\mathbb{R}^{m \times n}$ is essentially a vector norm on $\mathbb{R}^{mn}$, it follows that a matrix norm enjoys all the properties of a vector norm. In particular, any two matrix norms on $\mathbb{R}^{m \times n}$ are equivalent in the sense of Theorem 1.9. This means that if we define convergence in $\mathbb{R}^{m \times n}$, in analogy with Definition 1.7, as
elementwise convergence, then \( \lim A_k = A \) if and only if \( \lim \nu(A - A_k) = 0 \) for any matrix norm \( \nu \). We shall return to the subject of limits of matrices at the end of this section.

Since we have agreed to identify \( n \times 1 \) matrices with \( n \)-vectors, any matrix norm on \( \mathbb{R}^{n \times 1} \) is also a vector norm on \( \mathbb{R}^n \), and conversely. As with vector norms, we may speak of families of matrix norms.

**Definition 2.2.** The function \( \nu : \bigcup_{m,n=1}^{\infty} \mathbb{R}^{m \times n} \to \mathbb{R} \) is a family of matrix norms if for each \( m, n \geq 1 \) the restriction of \( \nu \) to \( \mathbb{R}^{m \times n} \) is a matrix norm.

It follows from this definition and the preceding comments that if \( \nu \) is a family of matrix norms, then the restriction of \( \nu \) to \( \bigcup_{n=1}^{\infty} \mathbb{R}^{n \times 1} \) is a family of vector norms.

**Example 2.3.** The Frobenius norms defined for any \( A \in \mathbb{R}^{m \times n} \) by

\[
\nu(A) = \left( \sum_{i,j} |a_{ij}|^2 \right)^{1/2} \quad (2.1)
\]

form a family of matrix norms. The restriction of \( \| \cdot \|_F \) to \( \bigcup_{n=1}^{\infty} \mathbb{R}^{n \times 1} \) is the family of vector 2-norms; that is, \( \| x \|_2 = \| x \|_2 \) for all vectors \( x \).

The Frobenius norm is a natural generalization of the Euclidean vector norm. Since it is relatively easy to compute and satisfies the inequality (2.2), it is used rather frequently in matrix computations. However, all matrix norms satisfying Definition 2.1 need not satisfy an inequality such as (2.2). For example, a natural generalization of the \( \infty \)-norm is the function \( \nu \) defined for \( A \in \mathbb{R}^{m \times n} \) by

\[
\nu(A) = \max \{ |a_{ij}| : i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, n \}. \quad (2.3)
\]

It is easily verified that \( \nu \) is a matrix norm in the sense of Definition 2.1. However, if

\[
A = B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix},
\]

then \( \nu(A)\nu(B) = 1 \) while \( \nu(AB) = 2 \), so that it is not true that \( \nu(AB) \leq \nu(A)\nu(B) \). For this reason the matrix norm (2.3) is seldom encountered in the literature.

These considerations suggest that we consider the properties of matrix norms that satisfy a relation like (2.2). Since \( A, B, \) and \( AB \) will in general have different dimensions, we may use a different norm for each member of (2.2).
DEFINITION 2.4. Let \( \mu : \mathbb{R}^{l \times m} \to \mathbb{R} \), \( \nu : \mathbb{R}^{m \times n} \to \mathbb{R} \), and \( \varrho : \mathbb{R}^{l \times n} \to \mathbb{R} \) be matrix norms. Then \( \mu \), \( \nu \), and \( \varrho \) are consistent if

\[
\varrho(AB) \leq \mu(A)\nu(B)
\]

for all \( A \in \mathbb{R}^{l \times m} \) and \( B \in \mathbb{R}^{m \times n} \). A matrix norm on \( \mathbb{R}^{n \times n} \) is consistent if it is consistent with itself. A family of matrix norms \( \nu \) is consistent if

\[
\nu(AB) \leq \nu(A)\nu(B)
\]

whenever the product \( AB \) is defined.

As an example of a family of consistent matrix norms we show that the Frobenius norms of Example 2.3 are consistent.

THEOREM 2.5. The family \( \| \cdot \|_F \) is consistent.

PROOF. We first show that

\[
\| Ax \|_2 \leq \| A \|_F \| x \|_2,
\]

(2.4)

for any \( A \in \mathbb{R}^{l \times m} \) and \( x \in \mathbb{R}^{m} \). Let \( A \) be partitioned by rows: \( A^T = (a_1, a_2, \ldots, a_l) \). Then

\[
Ax = \begin{pmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_l^T x \end{pmatrix}.
\]

Hence

\[
\| Ax \|_2^2 = \sum_{i=1}^{l} |a_i^T x|^2.
\]

By the Cauchy inequality \( |a_i^T x| \leq \| a_i \|_2 \| x \|_2 \). Hence

\[
\| Ax \|_2^2 \leq \| x \|_2^2 \sum_{i=1}^{l} \| a_i \|_2^2.
\]

However, it is easily verified that \( \| A \|_F^2 = \sum_{i=1}^{l} \| a_i \|_2^2 \). Hence (2.4) is satisfied.
Now let $C = AB$, where $A \in \mathbb{R}^{l \times m}$ and $B \in \mathbb{R}^{m \times n}$. If $B = (b_1, b_2, \ldots, b_n)$ is partitioned by columns, then

$$
\| C \|_F^2 = \| AB \|_F^2 = \| (Ab_1, Ab_2, \ldots, Ab_n) \|_F^2
$$

$$
= \sum_{j=1}^n \| Ab_j \|_2^2 \leq \| A \|_F^2 \sum_{j=1}^n \| b_j \|_2^2 = \| A \|_F^2 \| B \|_F^2,
$$

which is the consistency relation.

By specialization, Definition 2.4 includes the notion of consistency of a vector norm and a matrix norm. For example, if $\| \cdot \| : \mathbb{R}^{n \times n} \to \mathbb{R}$ is a matrix norm and $v : \mathbb{R}^n \to \mathbb{R}$ is a vector norm, then $v$ and $\| \cdot \|$ are consistent if

$$
v(Ax) \leq \| A \| v(x).
$$

The inequality (2.4) in the proof of Theorem 2.5 shows that the Frobenius matrix norm is consistent with the Euclidean vector norm.

It sometimes happens that we are given a consistent matrix norm on $\mathbb{R}^{n \times n}$ and require a consistent vector norm. The proof of the following theorem shows how to construct such a norm.

**THEOREM 2.6.** Let $\| \cdot \| : \mathbb{R}^{n \times n} \to \mathbb{R}$ be a consistent matrix norm. Then there is a norm on $\mathbb{R}^n$ that is consistent with $\| \cdot \|$.

**PROOF.** Let $a \neq 0$ be an $n$-vector. Define the function $v : \mathbb{R}^n \to \mathbb{R}$ by

$$
v(x) = \| xa^T \|, \quad x \in \mathbb{R}^n.
$$

It is easily verified that $v$ is a vector norm. Moreover,

$$
v(Ax) = \| (Ax)a^T \| = \| A(xa^T) \| \leq \| A \| \| xa^T \| = \| A \| v(x),
$$

so that $v$ is consistent with $\| \cdot \|$. ■

We turn now to an important process for constructing matrix norms consistent with given vector norms. The idea of the construction is the following. Let $A \in \mathbb{R}^{m \times n}$ and let $v : \mathbb{R}^n \to \mathbb{R}$ and $\mu : \mathbb{R}^m \to \mathbb{R}$ be norms. If $v(x) = 1$ and $y = \alpha x$, then $v(y) = | \alpha |$ and

$$
\mu(Ay) = \mu(\alpha Ax) = | \alpha | \mu(Ax) = v(y)\mu(Ax).
$$
In other words, the number $\mu(Ax)$ measures how much the linear transformation $A$ magnifies (or diminishes) any vector that is a multiple of $x$. If there is a largest such number $\mu(Ax)$, this largest magnification constant is a natural candidate for a norm of $A$. In order to insure the existence of a vector $x$ for which $\mu(Ax)$ is maximal, we need the following theorem, which we state without proof.

THEOREM 2.7. Let $\mu : \mathbb{R}^m \to \mathbb{R}$ and $\nu : \mathbb{R}^n \to \mathbb{R}$ be vector norms, and let $A \in \mathbb{R}^{m \times n}$. Then there are vectors $y, z \in \mathbb{R}^n$ such that $\nu(y) = \nu(z) = 1$ and for all $x \in \mathbb{R}^n$

$$\nu(x) = 1 \implies \mu(Az) \leq \mu(Ax) \leq \mu(Ay).$$

Otherwise put, Theorem 2.7 says that the numbers

$$\max_{\nu(z) = 1} \mu(Ax) \quad \text{and} \quad \min_{\nu(z) = 1} \mu(Ax)$$

are well defined. Moreover the maximum and minimum values can be attained for certain vectors $y$ and $z$.

Now suppose we define the function $\| \cdot \| : \mathbb{R}^{m \times n} \to \mathbb{R}$ by

$$\| A \| = \max_{\nu(z) = 1} \mu(Ax). \quad (2.6)$$

Then it follows that

$$\mu(Ax) \leq \| A \| \nu(x), \quad (2.7)$$

so that the function $\| \cdot \|$ is consistent with the vector norms $\mu$ and $\nu$. We can also prove that $\| \cdot \|$ is a matrix norm. However, we can prove much more, as the following theorem shows.

THEOREM 2.8. Let $\nu$ be a family of vector norms. For $A \in \mathbb{R}^{m \times n}$ define

$$\| A \|_{\nu} = \max_{\nu(z) = 1} \nu(Ax).$$

Then $\| \cdot \|_{\nu} : \bigcup_{m,n>1}^{\infty} \mathbb{R}^{m \times n} \to \mathbb{R}$ is a consistent family of matrix norms. Moreover, if on $\mathbb{R}^1$ we have

$$\nu[(\xi)] = | \xi |, \quad (2.8)$$

then

$$\| x \|_{\nu} = \nu(x), \quad x \in \mathbb{R}^n.$$
PROOF. By Theorem 2.7, the function \( \| \cdot \|_\nu \) is well defined. By (2.7) it satisfies

\[
v(Ax) \leq \| A \|_\nu v(x).
\]

To show that \( \| \cdot \|_\nu \) is a family of matrix norms, we must show that for \( m, n \geq 1 \), the restriction of \( \| \cdot \|_\nu \) to \( \mathbb{R}^{m \times n} \) is a matrix norm.

1. **Definiteness:** Let \( A \neq 0 \). Then, say, the \( i \)th column of \( A \) is nonzero, so that \( Ae_i \neq 0 \). Then \( 0 < v(Ae_i) \leq \| A \|_\nu v(e_i) \). Since \( v(e_i) \neq 0 \), it follows that \( \| A \|_\nu > 0 \).

2. **Homogeneity:**

\[
\| \alpha A \|_\nu = \max \{ v(\alpha Ax) : v(x) = 1 \} = \alpha \| A \|_\nu,
\]

3. **Triangle inequality:** Let \( x \) be such that \( v(x) = 1 \) and \( v[(A + B)x] = \| A + B \|_\nu \). Then

\[
\| A + B \|_\nu = v[(A + B)x] \leq v(Ax) + v(Bx) \\
\leq \| A \|_\nu v(x) + \| B \|_\nu v(x) = \| A \|_\nu + \| B \|_\nu.
\]

To show that \( \| \cdot \|_\nu \) is consistent, let \( \| AB \|_\nu = v(ABx) \), where \( v(x) = 1 \). Then

\[
\| AB \|_\nu = v[(ABx)] \leq \| A \|_\nu v(Bx) \leq \| A \|_\nu \| B \|_\nu v(x) \\
= \| A \|_\nu \| B \|_\nu.
\]

Finally if \( v[(\xi)] = | \xi | \), it follows that for \( y \in \mathbb{R}^n \)

\[
\| y \|_\nu = \max \{ v(yx) : v(x) = 1 \} = \max \{ | \xi | v(y) : | \xi | = 1 \} v(y) = v(y).
\]

The family of matrix norms \( \| \cdot \|_\nu \) is said to be **subordinate** to the family of vector norms \( v \). The same proof shows that the function \( \| \cdot \| \) defined on \( \mathbb{R}^{m \times n} \) by (2.6) is a matrix norm. Such a norm is sometimes called an **operator norm** (subordinate to \( \mu \) and \( v \)). The most frequently occurring special case is when \( m = n \) and \( \mu = v \), in which case (2.6) defines a consistent norm on \( \mathbb{R}^{n \times n} \). It should be noted that unless \( \mu = v \), we cannot guarantee that the resulting norm is consistent.

The most natural family of vector norms to use in Theorem 2.8 is one of the \( p \)-norms (\( p = 1, 2, \infty \)). Since the \( p \)-norms satisfy (2.8), the subordinate matrix norms defined by the theorem give the same values as the
$p$-norms when applied to vectors. Hence there is no possibility of confusion in using the same symbol $\| \cdot \|_p$ for the matrix norm and the vector norm.

The 1-norm and the $\infty$-norm may be very easily computed.

**Theorem 2.9.** Let $A \in \mathbb{R}^{m \times n}$. Then

$$\| A \|_1 = \max \left( \sum_{j=1}^{n} |a_{ij}| : j = 1, 2, \ldots, n \right)$$ (2.9)

and

$$\| A \|_\infty = \max \left( \sum_{i=1}^{m} |a_{ij}| : i = 1, 2, \ldots, m \right).$$ (2.10)

**Proof.** The strategy of the proof is the same in both cases. Letting $\lambda$ be the right-hand side of (2.9) or (2.10), as the case may be, we first show that for any vector $x$ we have $\| Ax \|_p \leq \lambda \| x \|_p$ ($p = 1, \infty$). This implies that

$$\| A \|_p \leq \lambda.$$

Next we find a particular vector $x$ with $\| x \|_p = 1$ such that $\| Ax \|_p = \lambda$. This shows that

$$\| A \|_p \geq \lambda.$$

The two inequalities together imply that $\| A \|_p = \lambda$.

Specifically, for the 1-norm let $A = (a_1, a_2, \ldots, a_n)$ be partitioned by columns. Then $\lambda = \max \{ \| a_j \|_1 : j = 1, 2, \ldots, n \}$ and

$$\| Ax \|_1 = \| \xi_1 a_1 + \xi_2 a_2 + \cdots + \xi_n a_n \|_1 \leq | \xi_1 | \| a_1 \|_1 + | \xi_2 | \| a_2 \|_1 + \cdots + | \xi_n | \| a_n \|_1 \leq (| \xi_1 | + | \xi_2 | + \cdots + | \xi_n |) \max \{ \| a_j \|_1 \} = \lambda \| x \|_1.$$

On the other hand if $\lambda = \| a_k \|_1$, then $\| e_k \|_1 = 1$ and

$$\| Ae_k \|_1 = \| a_k \|_1 = \lambda.$$

This establishes (2.9).

For the $\infty$-norm, $\lambda = \max \{ \sum_{j=1}^{n} |a_{ij}| : i = 1, 2, \ldots, n \}$. For any $x$ we have

$$\| Ax \|_\infty = \max_i \left\{ \left| \sum_j a_{ij} \xi_j \right| \right\} \leq \max_i \left\{ \sum_j \left| a_{ij} \right| \left| \xi_j \right| \right\} \leq \max_i \left\{ \sum_j \left| a_{ij} \right| \right\} \max_j \left\{ \left| \xi_j \right| \right\} = \lambda \| x \|_\infty.$$
On the other hand if $\lambda = \sum_{j=1}^{n} |\alpha_{kj}|$ and $x = (\text{sign}(\alpha_{k1}), \text{sign}(\alpha_{k2}), \ldots, \text{sign}(\alpha_{kn}))^T$, then $\|x\|_{\infty} = 1$ and $\|Ax\|_{\infty} = \lambda$. \hfill $\blacksquare$

Thus the 1-norm of a matrix is equal to the maximum of the 1-norms of its columns. For this reason the matrix 1-norm is often called the column sum norm. Similarly the $\infty$-norm is called the row sum norm. Because these norms are so easy to compute they, along with the Frobenius norm, are often used in matrix algorithms.

There is no computationally convenient characterization of the matrix 2-norm (to anticipate a little, $\|A\|_2^2$ is the largest eigenvalue of $A^TA$). However, the 2-norm does have a number of nice properties that make it useful for theoretical purposes. Some of these properties are contained in the following theorem.

**Theorem 2.10.** Let $A \in \mathbb{R}^{m \times n}$. Then

1. $\|A\|_2 = \max_{\|x\|_2 = 1} |y^TAx|$,  
2. $\|AT\|_2 = \|A\|_2$,  
3. $\|ATA\|_2 = \|A\|_2^2$.

**Proof.** For part 1, let $\|x\|_2 = \|y\|_2 = 1$. Then by the Cauchy inequality,

$$|y^TAx| \leq \|y\|_2 \|Ax\|_2 \leq \|y\|_2 \|x\|_2 \|A\|_2 = \|A\|_2.$$ 

On the other hand let $\|x\|_2 = 1$ and $\|Ax\|_2 = \|A\|_2$. Set $y = Ax/\|Ax\|_2$. Then $\|y\|_2 = 1$ and

$$|y^TAx| = \frac{x^TA^TAx}{\|Ax\|_2} = \frac{\|Ax\|_2^2}{\|Ax\|_2} = \|A\|_2^2 = \|A\|_2,$$

so that for this choice of $x$ and $y$ equality is attained.

For part 2, note that

$$\|AT\|_2 = \max_{\|x\|_2 = 1} |y^TA^TAx| = \max_{\|y\|_2 = 1} |x^TAy| = \|A\|_2.$$ 

Finally, for part 3 we have

$$\|ATA\|_2 \leq \|AT\|_2 \|A\|_2 \leq \|A\|_2^2,$$

where the first inequality follows from the consistency of $\|\cdot\|_2$ and the se-
cond from part 2 of this theorem. For the reverse inequality, let \( \| x \|_2 = 1 \) and \( \| Ax \|_2 = \| A \|_2 \). Then by part 1 with \( y = x \),

\[
\| A^T A \|_2 \geq \| x^T A^T A x \| = \| Ax \|_2^2 = \| A \|_2^2. \]

We conclude this section with a brief discussion of matrix limits. The results are analogous to the corresponding results for vectors. However, because matrices have a product defined among them, there are some additional results. We begin with the definition of the limit of a sequence of matrices.

**Definition 2.11.** Let \( A, A_2, A_3, \ldots \in \mathbb{R}^{m \times n} \). Then \( \lim_{k \to \infty} A_k = A \) if

\[
\lim_{k \to \infty} a_{ij}^{(k)} = a_{ij} \quad (i = 1, 2, \ldots, m; \; j = 1, 2, \ldots, n).\]

As was mentioned at the beginning of this section, the notion of limit of a sequence of matrices can be characterized in terms of any matrix norm \( v \). Namely

\[
\lim_{k \to \infty} A_k = A \iff \lim_{k \to \infty} v(A - A_k) = 0.\]

Many of the algebraic properties of limits of scalars carry over to limits of matrices. A few are given in the following theorem, in which it is assumed that the dimensions of the matrices in the statements are consistent with the operators involved.

**Theorem 2.12.** Let \( \lim A_k = A \) and \( \lim B_k = B \). Then

1. \( \lim (A_k + B_k) = A + B \),
2. \( \lim A_k B_k = AB \),
3. if \( A \) is nonsingular, then, for all sufficiently large \( k \), \( A_k \) is nonsingular and \( \lim A_k^{-1} = A^{-1} \).

The proofs of parts 1 and 2 are immediate from the definition of limit and are left as exercises. The proof of part 3 must be deferred to the end of the next section.

It should be noted that other properties of the limit are immediate consequences of this theorem. For example, since \( \lambda A = \text{diag}(\lambda, \lambda, \ldots, \lambda) A \), it follows from 2.12.2 that \( \lim \lambda A_k = \lambda \lim A_k \). Other properties of the limit are given in the exercises.
EXERCISES

1. Verify that the function $v$ defined by (2.5) is a vector norm.

2. Establish the inequality (2.7) from the definition (2.6).

3. Prove that if $A = (a_1, a_2, \ldots, a_n)$ is partitioned by columns then $\| A \|_F^2 = \| a_1 \|_2^2 + \| a_2 \|_2^2 + \cdots + \| a_n \|_2^2$.

4. Show that $\| AB \|_F \leq \| A \|_2 \| B \|_F$ and $\| AB \|_F \leq \| A \|_F \| B \|_2$.

5. Show that $\| \cdot \|_2$ and $\| \cdot \|_F$ are not the same by calculating $\| I_n \|_2$ and $\| I_n \|_F$ for $n > 1$.

6. Show directly that for $x \in \mathbb{R}^n$, $\| x \|_2 = \| x^T \|_2$.

7. Let $\| \cdot \| : \mathbb{R}^{m \times n} \to \mathbb{R}$ be the operator norm subordinate to the vector norm $v : \mathbb{R}^n \to \mathbb{R}$. Show that $\| I \|_v = 1$.

8. Let $v : \mathbb{R}^{m \times n} \to \mathbb{R}$ be defined by

$$v(A) = n \{ \max |\alpha_{ij}| : i, j = 1, 2, \ldots, n \}.$$  

Show that $v$ is a consistent matrix norm.

9. Show that the function $v : \bigcup_{m,n=1}^{\infty} \mathbb{R}^{m \times n} \to \mathbb{R}$ defined by

$$v(A) = \sum_{i=1}^{m} \sum_{j=1}^{n} |\alpha_{ij}|, \quad A \in \mathbb{R}^{m \times n},$$

is a consistent family of matrix norms.

10. Let $v : \mathbb{R}^{m \times n} \to \mathbb{R}$ be a matrix norm. Let $B \in \mathbb{R}^{m \times m}$ and $C \in \mathbb{R}^{n \times n}$ be nonsingular. Show that the function $\mu : \mathbb{R}^{m \times n} \to \mathbb{R}$ defined by $\mu(A) = v(BAC)$ is a matrix norm.

11. Let $v : \mathbb{R}^n \to \mathbb{R}$ be a norm and let $\| \cdot \|_v$ be the operator norm on $\mathbb{R}^{n \times n}$ subordinate to $v$. Show that if $A$ is nonsingular, then

$$\| A^{-1} \|_v^{-1} = \min_{v(x)=1} v(Ax).$$

12. Let $v$ be a vector norm on $\mathbb{R}^n$, let $B \in \mathbb{R}^{n \times n}$ be nonsingular, and let $\mu$ the norm defined by $\mu(x) = v(Bx)$. Let $\| \cdot \|_v$ and $\| \cdot \|_\mu$ denote the operator norms subordinate to $v$ and $\mu$. Show that $\| A \|_\mu = \| BAB^{-1} \|_v$. 
13. For $p, q = 1, 2, \infty, F$ establish the following table of constants $\tau_{pq}$ such that for all $A \in \mathbb{R}^{n \times n}$, \[ \| A \|_p \leq \tau_{pq} \| A \|_q. \]

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>1</th>
<th>2</th>
<th>$\infty$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>$n^{1/2}$</td>
<td>$n$</td>
<td>$n^{1/2}$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>$n^{1/2}$</td>
<td>1</td>
<td>$n^{1/2}$</td>
<td>1</td>
</tr>
<tr>
<td>$\infty$</td>
<td></td>
<td>$n$</td>
<td>$n^{1/2}$</td>
<td>$n^{1/2}$</td>
<td>1</td>
</tr>
<tr>
<td>$F$</td>
<td></td>
<td>$n^{1/2}$</td>
<td>$n^{1/2}$</td>
<td>$n^{1/2}$</td>
<td>1</td>
</tr>
</tbody>
</table>

14. Let $\nu : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ be a matrix norm. Show that there is a constant $\lambda$ such that the function $\mu$ defined by $\mu(A) = \lambda \nu(A)$ is a consistent matrix norm.

15. Let $A \in \mathbb{R}^{n \times n}$ and $\mu$ and $\nu$ be norms on $\mathbb{R}^n$. Show that there exists a vector $x$ with $\nu(x) = 1$ such that for all $y$ with $\nu(y) = 1$

\[ \mu(Ax) \leq \mu(Ay). \] 

16. Prove that if $\lim \lambda_k = \lambda$ and $\lim A_k = A$, then $\lim \lambda_k A_k = \lambda A$.

17. Prove that if $A$ is nonsingular and $\lim AB_k = C$, then $\lim B_k$ exists and $A \lim B_k = C$. Show by example that the hypothesis of nonsingularity of $A$ cannot be removed.

18. The sequence $\langle A_k \rangle$ is a Cauchy sequence with respect to the matrix norm $\nu$ if for every $\varepsilon > 0$ there is an integer $N$ such that if $m, n \geq N$, then $\nu(A_m - A_n) < \varepsilon$. Show that if $\mu$ and $\nu$ are matrix norms and $\langle A_k \rangle$ is a Cauchy sequence with respect to $\nu$, then $\langle A_k \rangle$ is a Cauchy sequence with respect to $\mu$. This justifies speaking simply of Cauchy sequences of matrices.

19. Show that $\langle A_k \rangle$ is a Cauchy sequence if and only if for each $i, j$ the sequence $\langle a_{ij}^{(k)} \rangle$ is a Cauchy sequence. Conclude from the fact that Cauchy sequences of scalars converge, that Cauchy sequences of matrices converge.

20. The infinite series $\sum_{k=0}^{\infty} A_k$ of matrices converges if the sequence of partial sums $S_k = \sum_{i=1}^{k} A_i$ converges. Prove that if $\nu$ is a matrix
norm and $\sum_{k=0}^{\infty} \alpha_k$ is a convergent series of scalars such that $v(A_k) \leq \alpha_k$ 
$(k = 0, 1, 2, \ldots)$, then $\sum_{k=0}^{\infty} A_k$ converges.

21. Let $\phi(x) = \sum_{k=0}^{\infty} \gamma_k x^k$ be a power series with radius of convergence $\rho$. 
Let $A \in \mathbb{R}^{n \times n}$. Show that if, for any consistent matrix norm $v$, 
$v(A) < \rho$, then the series $\sum_{k=0}^{\infty} \gamma_k A^k$ converges. [The limit of the series 
is written $\phi(A).$]

22. Let $A \in \mathbb{R}^{n \times n}$ and define $e^A = \sum_{k=0}^{\infty} A^k/k!$. Prove that if $AB = BA$, 
then $e^{A+B} = e^A e^B$.

NOTES AND REFERENCES

The concept of norm has long been used in functional analysis; its application to matrix theory is more recent. Details and further references are given by Householder (1964, Chapter II), who has done much to popularize the use of norms in analyzing matrix process. Mention should also be made of Bauer's "Theory of Norms" (1967), which unfortunately is available only as a report.

In many papers matrix norms are used only for square matrices of a fixed order. Since the product is always defined among such matrices, it is customary to add consistency as a property of a matrix norm, a usage which conflicts with our definition.

The notion of a family of norms appears to be new. It is merely a terminological device for saying in a precise way what everyone knows anyway.

Theorems 1.9 and 2.7 require compactness arguments for their proofs, but are not otherwise very difficult. A proof of Theorem 1.9 may be found in the book of Issacson and Keller (1966).

3. INVERSES OF PERTURBED MATRICES

In this section we shall consider the following problem. Let $A \in \mathbb{R}^{n \times n}$ be nonsingular and let $E$ be an $n \times n$ matrix that is presumed to be small. 
How small must $E$ be so that the perturbed matrix $A + E$ is also nonsingular, and by how much does $(A + E)^{-1}$ differ from $A^{-1}$? In answering these questions we shall use the theory of norms developed in the last two sections to make precise the phrases "how small" and "how much."

One application of our results is to the computation of matrix inverses. In Section 3.5 we indicated that the computed inverse of a matrix $A$ would often be near the exact inverse of a slightly perturbed matrix.
A + E. However, this result does not guarantee the accuracy of the computed inverse, for $A^{-1}$ and $(A + E)^{-1}$ may differ greatly. In the terminology of Section 2.3 such a matrix would be called *ill conditioned with respect to inversion*. Our analysis not only gives conditions under which $A$ is ill conditioned, but it also associates with $A$ a *condition number* that measures the degree of its ill-conditioning.

In order to speak rigorously about the sizes of errors involving vectors and matrices, we introduce the following generalizations of absolute and relative error.

**Definition 3.1.** Let $A, B \in \mathbb{R}^{m \times n}$ with $B$ regarded as an approximation to $A$. The *residual* of $B$ is the matrix

$$ A - B. $$

If $\nu : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ is a norm, the *error in $B$ with respect to $\nu$* is the number

$$ \nu(A - B). $$

If $A \neq 0$, the *relative error in $B$ with respect to $\nu$* is the number

$$ \frac{\nu(A - B)}{\nu(A)}. $$

Where it will cause no confusion, we drop the phrase "with respect to $\nu" and refer simply to the error or the relative error. The notions of residual, error, and relative error are of course defined for $n$-vectors regarded as $n \times 1$ matrices.

**Example 3.2.** On a given computer suppose that the rounded value of a number $\beta$ is $\text{fl}(\beta) = \beta(1 + \epsilon)$, where $|\epsilon| \leq 10^{-t}$. Let $A \in \mathbb{R}^{m \times n}$ and let $\alpha$ be the magnitude of the largest element of $A$. Let $B$ be obtained from $A$ by rounding the elements of $A$. Then $B = A + E$, where

$$ \sum_{j=1}^{n} |\epsilon_{ij}| \leq \sum_{j=1}^{n} |q_{ij}| |\alpha_{ij}| \leq 10^{-t} \| A \|_{\infty}. $$

Hence

$$ \| E \|_{\infty} \leq \| A \|_{\infty} 10^{-t}, $$

and

$$ \frac{\| A - B \|_{\infty}}{\| A \|_{\infty}} \leq 10^{-t}. $$
that is, the relative error with respect to the \( \infty \)-norm in a matrix rounded to \( t \) figures is not greater than \( 10^{-t} \).

The definition of error and relative error uses the notion of a norm to combine information about a set of numbers into a single number. It is not surprising that we should lose information in the process. For example, a small component of a vector with a low relative error may have a high relative error. Moreover, a change of norm may change the relative error markedly.

**EXAMPLE 3.3.** Let

\[
A = \begin{pmatrix} 1.000 & 0.0050 \\ 0.0050 & 0.0001 \end{pmatrix},
\]

\[
B = \begin{pmatrix} 1.0001 & 0.0051 \\ 0.0051 & 0.0002 \end{pmatrix},
\]

and

\[
C = \begin{pmatrix} 1.0001 & 0.0050001 \\ 0.0050001 & 0.00010001 \end{pmatrix}.
\]

Obviously \( C \) is a fairly good approximation to \( A \), while \( B \) has a relative error of unity in its \((2, 2)\)-element. In spite of this both matrices have a relative error of about \( 10^{-4} \) with respect to the \( \infty \)-norm. On the other hand, if we introduce the norm \( v \) defined by

\[
v(A) = \| \text{diag}(1, 10^2) A \text{diag}(1, 10^2) \|_\infty,
\]

then

\[
\frac{v(A - B)}{v(A)} \approx 1
\]

while

\[
\frac{v(A - C)}{v(A)} \approx 10^{-4}.
\]

Thus the norm \( v \) exposes the inaccuracy in the \((2, 2)\)-element of \( B \).

Example 3.3 indicates the necessity of choosing a norm to fit the problem, or alternatively, of scaling the problem to fit the norm. In applications to linear systems, the problem of choosing a norm is closely related to the scaling problems we discussed at the end of Section 3.5. Fortunately,
in many cases the choice of norm is perfectly evident, as it was in Example 3.3.
We now turn to the question of the accuracy of perturbed inverses that was raised at the beginning of this section. We shall answer the question by exhibiting a bound on the relative error in \((A + E)^{-1}\). For the rest of this and the next section all matrices will be assumed square of order \(n\). The symbol \(\| \cdot \|\) will denote both a consistent matrix norm on \(\mathbb{R}^{n \times n}\) satisfying
\[
\| I \| = 1
\] (3.1)
and a vector norm on \(\mathbb{R}^n\) that is consistent with the matrix norm \(\| \cdot \|\); that is, \(\| Ax \| \leq \| A \| \| x \|\). It should be noted that the results of this section depend critically on (3.1). In particular, they hold for the 1-, 2-, and \(\infty\)-norms, but not for the Frobenius norm.
We begin by considering perturbations of the identity matrix.

**Theorem 3.4.** If \(\| P \| < 1\), then \(I - P\) is nonsingular, and
\[
\| (I - P)^{-1} \| \leq (1 - \| P \|)^{-1}. \tag{3.2}
\]

**Proof.** Let \(x \neq 0\). Then
\[
\| (I - P)x \| = \| x - Px \| \geq \| x \| - \| Px \| \geq \| x \| - \| P \| \| x \|
\geq (1 - \| P \|) \| x \| > 0,
\]
since \(1 - \| P \| > 0\). Hence if \(x \neq 0\), then \((I - P)x \neq 0\), and \(I - P\) is nonsingular.
Now from the equation
\[
(I - P)(I - P)^{-1} = I,
\]
it follows that
\[
(I - P)^{-1} = I + P(I - P)^{-1}. \tag{3.3}
\]
Hence
\[
\| (I - P)^{-1} \| \leq \| I \| + \| P \| \| (I - P)^{-1} \|. \tag{3.4}
\]
Since \(\| I \| = 1\), the inequalities (3.4) and (3.2) are equivalent. \(\blacksquare\)

For the identity matrix, Theorem 3.4 answers the first question raised at the beginning of this section; namely how small must \(P\) be so that \(I - P\)
is nonsingular? The answers to the second question of the proximity of \((I - P)^{-1}\) to \(I^{-1} = I\) is contained in the following corollary.

**COROLLARY 3.5.** If \(\| P \| < 1\), then
\[
\| I - (I - P)^{-1} \| \leq \frac{\| P \|}{1 - \| P \|}. \tag{3.5}
\]

**PROOF.** From (3.3) we have
\[
I - (I - P)^{-1} = -P(I - P)^{-1}.
\]

Hence
\[
\| I - (I - P)^{-1} \| \leq \| P \| \| (I - P)^{-1} \| \leq \| P \| (1 - \| P \|^{-1}).
\]

For very small \(P\), the term \((1 - \| P \|^{-1})\) in (3.5) does not differ significantly from unity and the right-hand side of (3.5) becomes effectively \(\| P \|\). In other words, the error in the inverse of a slightly perturbed identity matrix is roughly of the same order as the perturbation.

Corollary 3.5 can be used to estimate the error in the inverse of a perturbed matrix. Actually, a slightly more elaborate result will prove useful later.

**THEOREM 3.6.** Let \(A\) be nonsingular and let \(\| A^{-1}E \| < 1\). Then \(A + E\) is nonsingular and \((A + E)^{-1}\) can be written in the form
\[
(A + E)^{-1} = (I + F)A^{-1}, \tag{3.6}
\]

where
\[
\| F \| \leq \frac{\| A^{-1}E \|}{1 - \| A^{-1}E \|}. \tag{3.7}
\]

Moreover
\[
\frac{\| A^{-1} - (A + E)^{-1} \|}{\| A^{-1} \|} \leq \frac{\| A^{-1}E \|}{1 - \| A^{-1}E \|}. \tag{3.8}
\]

**PROOF.** Since
\[
A + E = A(I + A^{-1}E)
\]
and \(\| A^{-1}E \| < 1\), it follows from Corollary 3.4 that \((I + A^{-1}E)\), and hence \(A + E\), is nonsingular. Moreover
\[
(A + E)^{-1} = (I + A^{-1}E)^{-1}A^{-1}.
\]
If we define $F$ by $I + F = (I + A^{-1}E)^{-1}$, then this equation is equivalent to (3.6), and by Theorem 3.5 (with $P = -A^{-1}E$) we have
\[ \| F \| = \| I - (I + A^{-1}E)^{-1} \| \leq \frac{\| A^{-1}E \|}{1 - \| A^{-1}E \|}. \]

Finally, from (3.6),
\[ A^{-1} - (A + E)^{-1} = -FA^{-1}, \]
so that
\[ \| A^{-1} - (A + E)^{-1} \| \leq \| F \| \| A^{-1} \| \leq \frac{\| A^{-1} \| \| A^{-1}E \|}{1 - \| A^{-1}E \|}, \]
which is equivalent to (3.8). ♣

Recall that if the approximation $\beta$ to $\alpha$ has a relative residual $\varrho$, then $\beta = (1 - \varrho)\alpha$. Equation (3.6) is quite analogous, and it is not out of place to call the matrix $-F$ the relative residual of $(A + E)^{-1}$ as an approximation to $A^{-1}$. Accordingly, we should expect that $\| F \|$ would be the relative error in $(A + F)^{-1}$; and indeed, comparing the inequalities (3.7) and (3.8), we see that $\| F \|$ and $\| A^{-1} - (A + E)^{-1} \|/\| A^{-1} \|$, which is the relative error in $(A + E)^{-1}$, have the same upper bounds.

If $\| A^{-1}E \|$ is significantly less than unity, then the term $(1 - \| A^{-1}E \|)^{-1}$ in (3.8) is essentially unity and the relative error in $(A + E)^{-1}$ is approximately bounded by $\| A^{-1}E \|$. This suggests that if $A^{-1}$ is very large, the error $E$ in $A$ may be magnified considerably in the inverse of $A + E$. The following trivial corollary of Theorem 3.6 makes these considerations precise.

**COROLLARY 3.7.** In Theorem 3.6, let
\[ \kappa(A) = \| A \| \| A^{-1} \|. \] (3.9)

If $\| A^{-1} \| \| E \| < 1$, then
\[ \| F \| \leq \frac{\kappa(A) \| E \|}{\| A \|} \] (3.10)
and
\[ \frac{\| A^{-1} - (A + E)^{-1} \|}{\| A^{-1} \|} \leq \frac{\kappa(A) \| E \|}{1 - \kappa(A) \| E \|}. \] (3.11)
PROOF. We have
\[ \| A^{-1}E \| \leq \| A^{-1} \| \| E \| = \| A \| \| A^{-1} \| \ \frac{\| E \|}{\| A \|} = \kappa(A) \frac{\| E \|}{\| A \|}. \]

If this upper bound is substituted for \( \| A^{-1}E \| \) in (3.7) and (3.8), there result the inequalities (3.10) and (3.11).

The left-hand side of (3.11) is the relative error in \( (A + E)^{-1} \). If \( E \) is sufficiently small, the right-hand side is effectively \( \kappa(A) \| E \|/\| A \| \). Since \( \| E \|/\| A \| \) is the relative error in \( A + E \), the inequality (3.11) states that the relative error in \( A + E \) may be magnified by as much as \( \kappa(A) \) in passing to \( (A + E)^{-1} \). For this reason, \( \kappa(A) \) is called the condition number of \( A \) with respect to inversion (and with respect to the norm \( \| \cdot \| \)). If \( \kappa(A) \) is large, then the inverse of \( A \) is sensitive to small perturbations in \( A \), and the problem of computing the inverse is ill conditioned. Incidentally, note that \( \kappa(A) \) is indeed a magnification constant, for

\[ 1 \leq \| I \| = \| AA^{-1} \| \leq \| A \| \| A^{-1} \| = \kappa(A). \]

EXAMPLE 3.8. Suppose that we are given a \( t \)-digit approximation \( B \) to \( A \), in the sense that \( \| A - B \| /\| A \| \approx 10^{-t} \). If \( \kappa(A) = 10^p \), then the relative error in \( B^{-1} \) will be approximately \( 10^{p-t} \), and if the elements of \( A^{-1} \) are about equal in magnitude, they will be accurate to about \( t - p \) digits. In other words, if \( \kappa(A) = 10^p \), we may expect to lose roughly \( p \) significant figures in inverting an approximation to \( A \).

It should be stressed that Corollary 3.7 is weaker than Theorem 3.6, since it is based on the inequality
\[ \| A^{-1}E \| \leq \| A^{-1} \| \| E \|. \quad (3.12) \]

For random perturbations \( E \), the inequality (3.12) will usually be almost an equality. However, if \( E \) has special properties, then Theorem 3.6 may give much sharper bounds than Corollary 3.7. For example if \( E = \varepsilon A \), where \( |\varepsilon| < 1 \), then Theorem 3.6 gives
\[ \| F \| \leq \frac{\varepsilon}{1 - \varepsilon} \quad (3.13) \]
while Corollary 3.7 gives
\[ \| F \| \leq \frac{\nu(A)\varepsilon}{1 - \nu(A)\varepsilon}. \]

Obviously (3.13) is the better bound.

We conclude this section by completing the proof of Theorem 2.12.

**Theorem 2.12.3.** If \( A \) is nonsingular and \( \lim A_k = A \), then, for all sufficiently large \( k \), \( A_k \) is nonsingular and \( \lim A_k^{-1} = A^{-1} \).

**Proof.** Let \( E_k = A_k - A \). Then \( \lim \| E_k \| = 0 \). Hence \( \lim \| A^{-1}E_k \| = 0 \). Thus for all sufficiently large \( k \), \( \| A^{-1}E_k \| < 1 \), and by Theorem 3.6, \( A_k = A + E_k \) is nonsingular. Moreover, \( A_k^{-1} = (I + F_k)A^{-1} \), where \( \| F_k \| \leq \| A^{-1}E_k \|/(1 - \| A^{-1}E_k \|) \). It follows that \( \lim \| F_k \| = 0 \) and hence
\[ \lim A_k^{-1} = \lim(I + F_k)A^{-1} = A^{-1}. \]

**Exercises**

1. Let \( A \) and \( B \) be of order \( n \) with \( A \) nonsingular, and let \( B \) be regarded as an approximation to \( A \). Define the left relative residual of \( B \) as the matrix \( F = (A - B)A^{-1} \) and the right relative residual of \( B \) as the matrix \( G = A^{-1}(A - B) \). Show that \( B = (I - F)A \) and \( B = A(I - G) \).

2. Let \( A, E \in R^{m \times n} \) with \( \| EA^{-1} \| < 1 \). Prove that \( A + E \) is nonsingular and that there is a matrix \( G \) satisfying \( \| G \| \leq \| EA^{-1} \|/(1 - \| EA^{-1} \|) \) such that \( (A + E)^{-1} = A^{-1}(I + G) \).

3. Let \( A, B \in R^{m \times n} \) and let \( A \) have linearly independent columns. Show that it is possible to define a left relative residual \( F \) of \( B \) as an approximation to \( A \) so that \( B = (I - F)A \). [Hint: \( A \) has a left inverse \( (A^TA)^{-1}A^T \) (cf. Exercise 1.6.8).] What can be said if \( A \) has linearly independent rows?

4. Prove that if \( \| P \| < 1 \), then the Neumann series \( I + P + P^2 + \cdots \) converges to \( (I - P)^{-1} \). Use this to give an alternate proof of Theorem 3.4 and Corollary 3.5.

5. Show that if \( P \) is sufficiently small, then \( (I - P)^{-1} \simeq (I + P) \) in the sense that \( \| (I + P) - (I - P)^{-1} \| \leq 2 \| P \|^2 \). Conclude that the bound (3.5) is sharp.
6. Prove that the matrix $F$ of Theorem 3.6 is given by the infinite series

$$F = -A^{-1}E[I - A^{-1}E + (A^{-1}E)^2 - \cdots].$$

Hence derive the bound (3.7) and conclude that it is sharp.

7. Let

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1.01 \end{pmatrix}.$$  

For

$$E = \begin{pmatrix} .0001 & -.0001 \\ -.0001 & .0001 \end{pmatrix}$$

compute the relative error in $(A + E)^{-1}$. Also compute the error predicted by (3.8) and (3.11). Do the same for

$$E = \begin{pmatrix} .0001 & .0001 \\ .0001 & .0001 \end{pmatrix}.$$  

Explain any disparity in the results.

8. Explain under what conditions the inequality (3.12) may be expected to be nearly an equality when the elements of $E$ are random.

NOTES AND REFERENCES

The perturbation theory of this section is classical and goes through for bounded operators in a Banach space. The theorems are usually established by means of the Neumann series

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

The condition number $\| A \|_2 \| A^{-1} \|_2$ first appears in the rounding-error analysis of von Neumann and Goldstein (1947) in a somewhat disguised form. Turing (1948) seems to have been the first to use the term condition number, which he defines in terms of matrix norms as is done in the text.

4. THE ACCURACY OF SOLUTIONS OF LINEAR SYSTEMS

In Section 3.5, we saw that the computed solution $\tilde{x}$ of the nonsingular system of linear equations

$$Ax = b$$  

(4.1)
satisfies the equation

\[(A + E)x = b, \tag{4.2}\]

where \(E\) is in some sense small. However, the computed solution \(\tilde{x}\) will be accurate only if the problem of solving (4.1) is well conditioned in the sense of Section 2.1. In this section we shall derive bounds for the error in \(\tilde{x}\) as a function of \(E\). As in the last section, the bounds will depend on the condition number \(\kappa(A)\) defined by (3.9).

Since bounds on the matrix \(E\) in (4.2) are known, it is possible to combine them with the bounds of this section to bound the error in \(\tilde{x}\). Given a condition number for \(A\), this bound can be computed without computing \(\tilde{x}\). Such an error bound is called an \textit{a priori} bound. It is not unreasonable to suppose that if we compute \(\tilde{x}\), the additional information obtained in the course of the computations can be used to obtain a better bound. Such a bound that depends on the approximate solution itself is called an \textit{a posteriori} bound. We shall discuss \textit{a posteriori} bounds for the errors in approximate solutions of (4.1); however, the results are, in general, disappointing.

The first problem we shall consider is that of assessing the effects of a perturbation in the vector \(b\) on the solution of (4.1). So far as rounding-error analysis is concerned, this is not a very important problem, since the effects of rounding error can be accounted for by perturbations in \(A\) alone. In practice, however, the components of \(b\) may be contaminated with errors—perhaps they themselves have been computed with rounding error—and it is important to have a bound for the error induced in the solution.

Specifically suppose that \(\tilde{b}\) is an approximation to \(b\) and \(\tilde{x}\) is the solution of the system

\[A\tilde{x} = \tilde{b}.\]

Then

\[x - \tilde{x} = A^{-1}(b - \tilde{b}),\]

and hence

\[\| x - \tilde{x} \| \leq \| A^{-1} \| \| b - \tilde{b} \|. \tag{4.3}\]

The inequality (4.3) already suggests that if \(\| A^{-1} \|\) is large, then the error in \(\tilde{x}\) may be significantly greater than the error in \(\tilde{b}\). We can make this more precise by computing the relative error in the solution. From (4.1) it follows that

\[\| x \| \geq \frac{\| b \|}{\| A \|}. \tag{4.4}\]
Hence
\[ \frac{\| x - \tilde{x} \|}{\| x \|} \leq \| A \| \| A^{-1} \| \frac{\| b - \tilde{b} \|}{\| b \|}. \]

If, as in Section 3, we define \( \kappa(A) = \| A \| \| A^{-1} \| \), then our results may be summarized in the following theorem.

**THEOREM 4.1.** Let \( A \) be nonsingular, \( Ax = b \neq 0 \), and \( A\tilde{x} = \tilde{b} \). Then
\[ \frac{\| x - \tilde{x} \|}{\| x \|} \leq \kappa(A) \frac{\| b - \tilde{b} \|}{\| b \|}, \]
where \( \kappa(A) = \| A \| \| A^{-1} \| \).

This theorem requires two comments. In the first place, the condition number \( \kappa(A) \) plays much the same role as it did in Corollary 3.7; the relative error in \( b \) may be magnified by as much as a factor of \( \kappa(A) \) in the solution. The second comment is that when \( A \) is ill conditioned the result is frequently unrealistic, since the lower bound (4.4) on \( \| x \| \) is often a severe underestimate. In fact for most right-hand sides \( b \),
\[ \| x \| = \| A^{-1}b \| = \gamma \| A^{-1} \| \| b \|, \]
where \( \gamma < 1 \) is a constant near unity. If this estimate is used in place of (4.4), the result is
\[ \frac{\| x - \tilde{x} \|}{\| x \|} \leq \gamma^{-1} \frac{\| b - \tilde{b} \|}{\| b \|}, \]
so that the relative error \( \tilde{x} \) is hardly greater than the relative error in \( \tilde{b} \). If \( x \) and \( b \) satisfy (4.5) with \( \gamma \) near unity, then they are said to reflect the condition of the matrix \( A \). Our observation is, then, that a problem that reflects the condition of \( A \) is insensitive to perturbations in \( b \), even if \( \kappa(A) \) is large.

Either Theorem 3.6 or Corollary 3.7 may be used to estimate the error in the solution of (4.2), with Corollary 3.7 giving a weaker result. We state both results.

**THEOREM 4.2.** Let \( A \) be nonsingular, let \( x \neq 0 \) satisfy (4.1), and let \( \tilde{x} \) satisfy (4.2). If \( \| A^{-1}E \| < 1 \), then
\[ \frac{\| x - \tilde{x} \|}{\| x \|} \leq \frac{\| A^{-1}E \|}{1 - \| A^{-1}E \|}. \]
If \( \| A^{-1} \| \| E \| < 1 \), then

\[
\frac{\| x - \bar{x} \|}{\| x \|} \leq \frac{\kappa(A) \| E \|}{1 - \kappa(A) \| E \|} \| A \|.
\]  

(4.7)

PROOF. Since \( \| A^{-1} E \| < 1 \), by Theorem 3.6 the matrix \( A + E \) is nonsingular and \( (A + E)^{-1} = (I + F)A^{-1} \), where \( F \) satisfies (3.7). Then

\[
\bar{x} = (A + E)^{-1}b = (I + F)A^{-1}b = (I + F)x,
\]

or

\[
x - \bar{x} = -Fx.
\]

Hence

\[
\frac{\| x - \bar{x} \|}{\| x \|} \leq \| F \|.
\]

If (3.7) is used to estimate \( F \), then (4.6) results. If (3.10) is used, then (4.7) results.

It is interesting to note that the bounds are independent of the right-hand side \( b \). The number \( \kappa(A) \) again serves as a condition number in (4.7), telling how much the relative error in \( A \) is magnified in the solution. Unless \( \| A^{-1} E \| \ll \| A^{-1} \| \| E \| \), which is unlikely for random perturbations, the bounds (4.6) and (4.7) will give comparable results. Moreover, these bounds will be sharp if \( \| Fx \| \approx \| F \| \| x \| \), which will usually be true. This contrasts with the bound in Theorem 4.1 which is likely to be an overestimate when \( A \) is ill conditioned.

These bounds may be combined with the results of Section 3.5 to yield rigorous bounds on the accuracy of computed solutions of linear equations. For example, provided there is no growth in the elimination process, Theorem 3.5.3 may be interpreted as saying that the computed solution \( \bar{x} \) of (4.1) satisfies

\[
(A + H)\bar{x} = b,
\]

(4.8)

where

\[
\| H \|_{\infty} \leq \phi(n) \| A \|_{\infty} 10^{-t}.
\]

(4.9)

Here \( \phi(n) \) is a function of \( n \) whose form depends on the arithmetic details.
of the computation, but which in any event is not very big. If
\[ \kappa(A) \phi(n) 10^{-t} < 1, \]
then the condition \( \| A^{-1} \|_\infty \| H \|_\infty < 1 \) is satisfied, and Theorem 4.2 applies to show that
\[ \frac{\| x - \bar{x} \|_\infty}{\| x \|_\infty} \leq \frac{\kappa(A) \phi(n) 10^{-t}}{1 - \kappa(A) \phi(n) 10^{-t}}. \]

Recalling Example 3.8, we see that if \( \phi(n) \) is not too large and \( \kappa(A) = 10^p \), then we may expect a solution computed in \( t \)-digit arithmetic to be accurate to about \( t - p \) significant figures, at least when there is no severe imbalance in the sizes of the elements.

We turn now to the problem of calculating a posteriori bounds for an approximate solution \( \bar{x} \) of (4.1). A natural check on \( \bar{x} \) is to substitute it into Equation (4.1) and see to what extent it fails to satisfy the equation. This amounts to looking at the size of the residual vector
\[ r = b - A\bar{x}. \]

If \( r \) is zero, then \( \bar{x} \) is an exact solution of (4.1). If \( r \) is small, it might be expected that \( \bar{x} \) is near a solution of (4.1). The following theorem states precisely in what sense this is true.

**THEOREM 4.3.** Let \( A \) be nonsingular and \( Ax = b \neq 0 \). Let \( \bar{x} \) be given and set \( r = b - A\bar{x} \). Then
\[ \frac{\| x - \bar{x} \|}{\| x \|} \leq \kappa(A) \frac{\| r \|}{\| b \|}. \quad (4.10) \]

**PROOF.** We have
\[ A^{-1}r = A^{-1}b - \bar{x} = x - \bar{x}. \]

Hence
\[ \| x - \bar{x} \| \leq \| A^{-1} \| \| r \|. \quad (4.11) \]

However,
\[ \| x \| \geq \frac{\| b \|}{\| A \|}. \quad (4.12) \]

If (4.11) is divided by (4.12), the result is (4.10).
As a practical means for assessing the accuracy of an approximate solution, Theorem 4.3 is somewhat disappointing; for it asserts that the accuracy of the solution depends not only on the size of the residual but also on the condition number of the matrix. If $A$ is ill conditioned, even a very small residual cannot guarantee an accurate solution. Worse yet, it is possible for an accurate solution to have a large residual. These points are illustrated in the following example.

**EXAMPLE 4.4.** Let

$$A = \begin{pmatrix} 1.000 & 1.001 \\ 1.000 & 1.000 \end{pmatrix}$$

and $b = (2.001, 2.000)^T$. Then the exact solution of the equation $Ax = b$ is $x = (1, 1)^T$. However, the vector $\tilde{x} = (2, 0)^T$, which is in no sense near $x$, has the very small residual vector $r = (10^{-3}, 0)^T$.

On the other hand, the solution of $Ax = b = (1, 0)^T$ is $x = (-1000, 1000)^T$, and the vector $\tilde{x} = (-1001, 1000)^T$ is very near $x$. However, the residual vector of $\tilde{x}$ is $r = (0, -1)$ which is as large as $b$.

Further insight into the nature of the residual may be gained by considering the residual of a rounded solution $\tilde{x}$ of (4.1). From Example 3.2 we know that $\tilde{x}$ may be written in the form $\tilde{x} = x + e$, where $\| e \|_\infty \leq \| x \|_\infty 10^{-t}$. Hence

$$r = b - A\tilde{x} = b - Ax - Ae = -Ae,$$

and $\| r \|_\infty \leq \| A \|_\infty \| e \|_\infty \leq \| A \|_\infty \| x \|_\infty 10^{-t}$. Thus

$$\frac{\| r \|_\infty}{\| A \|_\infty} \leq \| x \|_\infty 10^{-t}. \quad (4.13)$$

Now if $A$ is ill conditioned and $x$ reflects the ill-conditioning of $A$, so that $\| x \|_\infty$ is large, then $r$ will be large. In other words, the rounded solution of an equation may have a large residual. This phenomenon is illustrated by the second part of Example 4.4.

It is also instructive to compare the residual of the rounded solution $\tilde{x}$ of (4.1) with the residual of the solution $\tilde{x}$ computed by, say, Gaussian elimination. We know that $\tilde{x}$ satisfies (4.8), where $H$ satisfies (4.9). Hence $r = b - A\tilde{x} = H\tilde{x}$, or

$$\frac{\| r \|_\infty}{\| A \|_\infty} \leq \phi(n)10^{-t} \| \tilde{x} \|_\infty. \quad (4.14)$$
If \( \phi(n) \) is not too large, the bound (4.14) is comparable to (4.13). Thus the residual of the computed solution will be roughly of the same size as the residual of the exact solution rounded to \( t \) figures. Note that if \( A \) is ill-conditioned and \( \bar{x} \) does not reflect the ill-conditioning of \( A \), then \( \| \bar{x} \|_\infty \) will be small, say of order unity, and \( \| r \|_\infty \) will be of order \( 10^{-t} \). This can happen even when \( \bar{x} \) is inaccurate in all its significant figures.

So far all our bounds have the defect that they depend on the condition number of the matrix \( A \) and hence on \( A^{-1} \). Since the calculation of inverses is relatively expensive, it is natural to ask how one can detect ill-conditioning without computing \( A^{-1} \).

One sign of ill-conditioning is the emergence of a small pivot in the reduction of \( A \). For example, the Crout reduction decomposes \( A \) into the product \( LU \), where the diagonal elements of \( L \) are the pivots. If one of these diagonal elements is small, then \( L^{-1} \) will be large. In all probability \( A^{-1} = U^{-1}L^{-1} \) will also be large, and hence \( A \) will be ill conditioned. For most matrices this is a fairly reliable indicator of ill-conditioning, especially if complete pivoting is used. However, there do exist ill-conditioned matrices for which no small pivots emerge.

A second sign of ill-conditioning is the emergence of a large solution. Suppose, for example, that \( \| A \| = \| b \| = 1 \). Then if \( Ax = b \), we have

\[
\| x \| \leq \| A^{-1} \| \| b \| = \| A^{-1} \| = \kappa(A),
\]

so that if \( \| x \| \) is large, \( \kappa(A) \) must perforce be large. Unfortunately, an ill-conditioned system may have a small solution. Note that such a solution must also have a very small residual.

It would seem then that the only sure way of detecting ill-conditioning is to compute an approximation to \( A^{-1} \). Actually, for full matrices this is not entirely out of the question, for if an \( LU \) decomposition of \( A \) has already been computed at a cost of \( n^3/3 \) multiplications, the decomposition \( A^{-1} = U^{-1}L^{-1} \) may be computed for another \( n^3/3 \) multiplications. Thus for twice the work one can compute the upper bound \( \| U^{-1} \| \| L^{-1} \| \) on \( \| A^{-1} \| \). A cautious person might feel the additional work is not too high a price to pay for the security of knowing when his problems are ill conditioned. However, in the next section we shall consider a method for improving approximation solutions from which a fair estimate of the condition of the problem may be obtained.
EXERCISES

1. Let $A \in \mathbb{R}^{n \times n}$ be nonsingular and let $X$ be an approximation to $A^{-1}$. Define the residual matrix $R$ by $R = I - AX$. Show that

$$\frac{\| A^{-1} - X \|}{\| A^{-1} \|} \leq \| R \|.$$ 

Compare this result with the corresponding result for linear systems.

2. In Exercise 1 show that

$$\| I - XA \| \leq \kappa(A) \| R \|.$$ 

Hence conclude that a matrix $X$ can be a good approximate right inverse for $A$ but a bad approximate left inverse.

3. Give an example of an ill-conditioned matrix $A$ and a matrix $X$ for which the bound in Exercise 2 is nearly attained.

4. Discuss the economics of computing $A^{-1}$ to estimate the condition number of $A$ in the special case when $A$ is tridiagonal; upper Hessenberg.

NOTES AND REFERENCES

The material in this section is covered by Wilkinson (1963) and many others.

Given an approximate inverse $X$ to $A$ one can define, in analogy with the residual vector associated with a linear system, a residual matrix

$$R = I - AX.$$ 

Unlike the residual vector for linear systems, $R$ does reflect the accuracy of the approximate inverse $X$; for

$$\| A^{-1} - X \| = \| A^{-1}R \| \leq \| A^{-1} \| \| R \|,$$

hence

$$\frac{\| A^{-1} - X \|}{\| A^{-1} \|} \leq \| R \|.$$ 

However, it should be noted that if $A$ is ill conditioned, its rounded inverse may have a large residual.
Although it is possible to construct systems of equations that do not reveal any obvious signs of ill-conditioning, such systems do not often occur in practice. Noble (1969) gives a chart of various tests for ill-conditioning.

5. ITERATIVE REFINEMENT OF APPROXIMATE SOLUTIONS
OF LINEAR SYSTEMS

In this section we shall analyze a method for improving an approximate solution of the equation

\[ Ax = b, \]  

(5.1)

where \( A \) is a matrix of order \( n \) which is not too ill conditioned, in a sense to be made precise later. The method starts with an approximate solution \( x_1 \) and produces a new approximate solution \( x_2 \) which is nearer \( x \) than \( x_1 \). Obviously one can again apply the method to the approximate solution \( x_2 \) to obtain an even better approximation \( x_3 \). Proceeding in this way, we obtain a sequence \( \langle x_k \rangle \) of approximate solutions that converges to the true solution \( x \). A process, such as we have just described, that generates successively a sequence of approximate solutions of a problem is called an iteration. The method to be described in this section is therefore called the method of iterative refinement of an approximate solution of (5.1).

The idea behind the method is simple. Let \( x_1 \) be an approximate solution of (5.1), presumably obtained by the techniques of Section 3.4. Let

\[ r_1 = b - Ax_1 \]

be the residual corresponding to \( x_1 \). Then if we solve the system

\[ Ad_1 = r_1 \]

(5.2)

and calculate

\[ x_2 = x_1 + d_1, \]

it follows that

\[ x_2 = x_1 + A^{-1}r_1 = x_1 + A^{-1}(b - Ax_1) = x_1 + A^{-1}b - x_1 = A^{-1}b = x, \]

so that \( x_2 \) is the exact solution. Note that this procedure is quite cheap; for if the matrix \( A \) has been decomposed to compute \( x_1 \), say by Crout reduction, then it need not be decomposed again to solve (5.2). Since the
computation of the residual requires only $n^2$ multiplications, the whole process can be carried out with $2n^2$ multiplications.

In practice the computations must be carried out with rounding error, so that $x_2$ will not be an exact solution. In fact, the errors made in solving (5.2) for $d_1$ are of the same kind made in solving initially for $x_1$, and it is not at all clear that $x_2$ will even be as good an approximate solution as $x_1$. It is therefore necessary to give a detailed analysis of the process to discover under what conditions $x_2$ is an improvement.

Since the analysis is lengthy, we begin by stating its final objective. We shall attempt to exhibit a constant $\eta > 0$, independent of $x_1$, such that

$$
\| x - x_2 \|_\infty \leq \eta \| x - x_1 \|_\infty. \tag{5.3}
$$

This being done, it follows by induction that

$$
\| x - x_k \|_\infty \leq \eta^{k-1} \| x - x_1 \|_\infty. \tag{5.4}
$$

If we can adjust the computations so that $\eta < 1$, then $\lim \eta^k = 0$, and it follows from (5.4) that $\lim x_k = x$. The result we shall actually obtain is a little weaker than this, but it will still suffice to establish the practical convergence of the method. For notational convenience we shall let the symbol $\| \cdot \|$ without a subscript denote the $\infty$-norm $\| \cdot \|_\infty$.

The process with rounding error may be described as follows.

1) $r_1 = \operatorname{fl}(b - Ax_1) \equiv b - Ax_1 + e_1$

2) $d_1 = \operatorname{fl}(A^{-1}r_1) \equiv (A + H_1)^{-1}r_1 \tag{5.5}$

3) $x_2 = \operatorname{fl}(x_1 + d_1) \equiv x_1 + d_1 + g_1$

Here the vectors $g_1$ and $e_1$ represent the difference of the true values from the computed values (n.b. $e_1$ is not the natural basis vector). The matrix $H_1$ is the error matrix of Theorem 3.5.3. We have seen (cf. page 195) that if computations are performed in $t$-digit arithmetic, then

$$
\| H \| \leq \phi(n) \| A \| 10^{-t}.
$$

It follows from Corollary 3.7 that if, say,

$$
\phi(n)\kappa(A)10^{-t} < \frac{1}{2}, \tag{5.6}
$$

then

$$
(A + H_1)^{-1} = (I + F_1)A^{-1}, \tag{5.7}
$$
where
\[
\| F_1 \| \leq \frac{\phi(n)\kappa(A)10^{-t}}{1 - \phi(n)\kappa(A)10^{-t}} < 1. \tag{5.8}
\]

The first step in the analysis is to obtain a bound for \( \| e_1 \| \) in terms of \( \| x - x_1 \| \). To do this let
\[
\varrho = \frac{\| e_1 \|}{\| r_1 \|}. \tag{5.9}
\]
Note that \( \varrho \) is the relative error in the residual. Now
\[
r_1 - e_1 = b - Ax_1 = A(x - x_1).
\]
Hence
\[
\| r_1 \| - \| e_1 \| \leq \| A \| \| x - x_1 \|.
\]
Substituting the value for \( \| r_1 \| \) obtained from (5.9), we obtain
\[
(\varrho^{-1} - 1) \| e_1 \| \leq \| A \| \| x - x_1 \|,
\]
or
\[
\| e_1 \| \leq \frac{\varrho \| A \| \| x - x_1 \|}{1 - \varrho}. \tag{5.10}
\]

Now from statements 2 and 3 of (5.5), we have
\[
x - x_2 = x - x_1 - (A + H_1)^{-1}r_1 - g_1.
\]
In view of (5.7) and the definition of \( r_1 \), we have
\[
x - x_2 = x - x_1 - (I + F_1)A^{-1}(b - Ax_1 + e_1) - g_1
\]
\[
= x - x_1 - (I + F_1)(x - x_1 + A^{-1}e_1) - g_1
\]
\[
= -F_1(x - x_1) - (I + F_1)A^{-1}e_1 - g_1. \tag{5.11}
\]

Taking norms on both sides of (5.11) and remembering that \( \| F_1 \| < 1 \), we obtain
\[
\| x - x_2 \| \leq \| F_1 \| \| x - x_1 \| + 2 \| A^{-1} \| \| e_1 \| + \| g_1 \|.
\]
Finally if the bounds (5.8) and (5.10) are substituted for \( \| F_1 \| \) and \( \| e_1 \| \), the result is
\[
\| x - x_2 \| \leq \kappa(A)\left[ \frac{\phi(n)10^{-t}}{1 - \phi(n)\kappa(A)10^{-t}} + \frac{2\varrho}{1 - \varrho} \right] \| x - x_1 \| + \| g_1 \|. \tag{5.12}
\]
If we define
\[ \eta = \frac{\kappa(A)\phi(n)10^{-t}}{1 - \kappa(A)\phi(n)10^{-t}} + \frac{2\kappa(A)\rho}{1 - \rho}, \]  
(5.13)
then (5.12) becomes
\[ \| x - x_2 \| \leq \eta \| x - x_1 \| + \| g_1 \|. \]  
(5.14)

The bound (5.14) is not quite as good as the bound (5.3) since it contains the additional term \( \| g_1 \| \). However, \( g_1 \) is the error made in adding the correction \( d_1 \) to \( x_1 \), and unless \( x_1 \) is very near the solution it will be negligible compared to \( x - x_1 \) and \( x - x_2 \). Thus the problem of convergence is still one of determining when \( \eta \) is less than unity.

The first term in the definition of \( \eta \) is easily disposed of. If \( \kappa(A) \) satisfies the restriction (5.6), then by (5.8) this first term is less than unity. Since the polynomial \( \phi(n) \) from the rounding-error analysis is usually an overestimate, the actual bound will generally be a good deal smaller than unity. Thus if \( A \) is not too ill conditioned in the sense that it satisfies (5.6), we do not have to worry about the first term.

The second term is another story. Recall that the number \( \rho \) is the relative error in the computed residual. Now we know from the discussion in Section that if \( x_1 \) was computed as an approximate solution of (5.1), then its residual satisfies
\[ \| r_1 \| \leq \phi(n)10^{-t} \| A \| \| x_1 \|. \]  
(5.15)
On the other hand, when the product \( Ax_1 \) is formed, one of the terms \( \alpha_{ij}\xi_j^{(1)} \) will usually be of the same order of magnitude as \( \| A \| \| x_1 \| \). Since by (5.15) the elements of \( r \) are almost \( 10^{-t} \) smaller than \( \alpha_{ij}\xi_j^{(1)} \) one must cancel almost \( t \) significant figures when computing \( r_1 \). This means that if \( r_1 \) is calculated in \( t \)-digit arithmetic, the results will be inaccurate in almost their first figure and \( \rho \) will be almost unity. Obviously, \( 2\kappa(A)\rho/(1 - \rho) \), which is the second term in (5.13) may be much greater than unity.

On the other hand if \( r_1 \) is calculated in double precision (2\( t \) digits), the cancellation of \( t \) figures will still leave \( t \) accurate figures. Thus \( \rho \) will be about \( 10^{-t} \), and if \( \kappa(A) \) satisfies (5.6), the term \( 2\kappa(A)\rho/(1 - \rho) \) will be less than unity. Notice that double precision should be sufficient for all subsequent iterations of the process, since even the rounded exact solution will not have a residual much smaller than the right-hand side of (5.15). If inner products can be accumulated in double precision, the computation of the residual in double precision will require only single precision multi-
plications. Once the residual has been computed in double precision, it may be rounded to single precision and used in statement (2) of (5.5).

Thus if \( A \) is not too ill conditioned (this may still be very ill conditioned indeed) and if the residual \( r_1 \) is calculated in double precision, then the error \( \| x - x_2 \| \) will be smaller than the error \( \| x - x_1 \| \). If the process is applied iteratively to generate the sequence of approximate solutions \( x_1, x_2, x_3, \ldots \), each \( x_k \) will have a smaller error than its predecessor \( x_{k-1} \). This decrease in error will continue to occur until the error is approximately equal to \( \| g_k \| \) in (5.14). Practically, since \( g_k \) is the error made in adding the correction \( d_k \) to \( x_k \), \( \| g_k \| \) is approximately equal to \( \| x_k \| \cdot 10^{-t} \). If \( \| d_k \| \) is smaller than this, then \( d_k \) cannot be added accurately to \( x_k \) and we may as well terminate the iteration. The resulting approximation \( x_k \) will be very near the exact solution rounded to \( t \) figures.

We sum up all these considerations in the following algorithm.

**Algorithm 5.1.** Let \( x \) be an approximate solution of the equation \( Ax = b \). If \( A \) is not too ill conditioned, this algorithm, when performed in \( t \)-digit arithmetic, returns in \( x \) an approximate solution that is nearly equal to the exact solution rounded to \( t \) figures.

1) Compute \( r = b - Ax \) in double precision and round to single precision
2) Compute in single precision the solution of the equation \( Ad = r \)
3) If \( \| d \|_\infty / \| x \|_\infty \leq 10^{-t} \), terminate the iteration
4) \( x \leftarrow x + d \)
5) Go to 1

Several comments should be made about this algorithm. In the first place, as it is constructed, it may loop indefinitely if \( A \) is too ill conditioned for the iteration to converge. In this case the corrections \( d_i \) will not show a steady decrease but will behave erratically, now decreasing—now increasing. Thus it is sufficient to terminate the algorithm with an error stop when \( \| d_i \|_\infty / \| x_{i+1} \| \geq \| d_{i-1} \|_\infty / \| x_i \| \).

Secondly, we have not yet commented on the speed of convergence. However, from the approximate inequality

\[
\frac{\| x - x_{k-1} \|}{\| x \|} \leq \eta \frac{\| x - x_k \|}{\| x \|}
\]

it follows that if \( \eta \approx 10^{-p} \), then the \( x_k \) will improve at the rate of about \( p \).
figures per iteration (cf. Example 2.1.2). This fact allows us to estimate the condition number of $A$, for if we are gaining $p$ figures per iteration and $x_1$ is accurate to about $q$ figures, then $x_1$ and $x_2$ will agree to $q$ figures and $x_2$ and $x_3$ will agree to about $q + p$ figures. Thus

$$\frac{\|x_3 - x_2\|}{\|x_2 - x_1\|} = \frac{\|d_2\|}{\|d_1\|} \approx 10^{-p} \approx \eta.$$  

We have already seen that $\eta \approx \kappa(A) 10^{-t}$. Hence $10^t \|d_2\|/\|d_1\|$ should be a fair estimate for $\kappa(A)$ when $A$ is very ill conditioned. Of course if the process converges in one iteration, as is likely to happen when $\kappa(A)$ is significantly less than $10^{-t/2}$, this estimate is useless.

Finally, we note that, with a little cleaning up, our development amounts to a rigorous proof that Algorithm 5.1 must converge to something near the exact solution rounded to $t$ digits, provided $A$ is not too ill conditioned. However, we have not shown rigorously that if $A$ is too ill conditioned, the iteration will diverge. This leaves open the possibility that, with a violently ill-conditioned matrix, the iteration may appear to converge to a false solution. It is generally conceded that the probability of this happening is negligible and that the results of Algorithm 5.1 can be taken at face value.

**EXAMPLE 5.2.** Let

$$A = \begin{pmatrix} 7.000 & 6.990 \\ 4.000 & 4.000 \end{pmatrix}$$

and $b = (34.97, 20.000)^T$, so that $x = (2, 3)^T$ is the solution of the equation $Ax = b$. The inverse of $A$ to four figures is

$$A^{-1} = \begin{pmatrix} 100.0 & -174.8 \\ -100.0 & 175.0 \end{pmatrix},$$

and hence $\kappa(A) = \|A\|_\infty \|A^{-1}\|_\infty \approx 3850$. The $LU$ decomposition of $A$, computed in four-digit arithmetic is

$$\begin{pmatrix} 1.000 & 0.000 \\ 0.5714 & 1.000 \end{pmatrix} \begin{pmatrix} 7.000 & 6.990 \\ 0.000 & 0.006 \end{pmatrix}.$$  

Let $x_1$ be the solution of $Ax = b$ computed using this decomposition. If Algorithm 5.1 in four-digit arithmetic is applied to refine $x_1$, there results the following table of values.
In the above example \( \kappa(A)10^{-4} \approx 4 \), so that the \( A \) is almost too ill conditioned for the algorithm to work. On the basis of this estimate we should expect to gain about half a significant figure per iteration. The actual gain is more like one figure per iteration. The reason for this is that the convergence factor \( n \) is approximately equal to \( \| F \| \) where \( (I + F)A^{-1} = (A + E)^{-1} \) and \( E \) is the error made in solving the linear systems. Now most of this error \( E \) comes from the error in the \( LU \) decomposition, and it is easily verified that \( \| A - LU \| \approx 3 \cdot 10^{-4} \). Thus \( \| F \| \approx \| A^{-1} \| \| E \| \approx .8 \cdot 10^{-1} \), which is in accordance with the observed rate of convergence. For the same reason \( 10^4 \| d_2 \|/\| d_1 \| \approx 625 \) is an underestimate for \( \kappa(A) \).

The behavior of the residuals is interesting. The computed solution \( x_1 \) is inaccurate in all its digits. Nonetheless, the residual is about as small as can be expected, as was predicted in the last section. It is mildly surprising that, although the refined solution \( x_2 \) is much more accurate, its residual is appreciably larger. This is typical of the process. Although the iterates become more accurate, the residuals do not necessarily decrease. This is to be expected since we have seen in Section 4 that the computed solution \( x_1 \) and the final iterate, which is essentially the rounded exact solution, should have residuals of about the same size. The only way in which the above example is atypical is that the residual corresponding to \( x_4 \) is zero. This is because \( x_4 \) is the exact solution whose components happen to be integers.

**EXERCISE**

1. The method of iterative refinement can be applied to calculate a solution to \( Ax = b \) to any desired accuracy. The approximate solutions \( x_i \)
must, of course, be accumulated in higher and higher precision, and the residual vector, which will eventually get progressively smaller, must be calculated so that it is accurate. However, there is no point in calculating the residual so accurately that the second term in (5.13) is much less than the first. Assume that an upper bound for $\kappa(A)$ is known and that the computer can operate in $t, 2t, 3t, \ldots$ digit arithmetic. Write an algorithm for computing a solution of $Ax = b$ accurate to $mt$ figures.

NOTES AND REFERENCES

The method of iterative refinement in the form described here is due to Wilkinson (1963), who analyzes the process for fixed-point arithmetic. Moler (1967) has given a detailed analysis for floating-point arithmetic, which essentially parallels the one given here.

The merits of iterative refinement are not unquestioned. Its critics point out that one must save the original matrix to compute the residual; and anyway it is not worthwhile to try to compute accurate solutions of ill-conditioned problems. In favor of the method it can be said that its slow convergence is a reliable test for ill-conditioning, and for many people that is sufficient justification.

Codes for the iterative refinement of positive definite systems have been published by Martin, Peters, and Wilkinson (1966, HACLA/I/2) and for general systems by Bowdler, Martin, Peters, and Wilkinson (1966, HACLA/I/7).