Optional reading about gaussian elimination, LU factorization, and the condition number of linear systems

This is an excerpt from Chapter 2 of

Scientific Computing: An Introductory Survey (2nd edition)
by Michael T. Heath

Chapter 2

Systems of Linear Equations

2.1 Linear Systems

Systems of linear algebraic equations arise in almost every aspect of applied mathematics and scientific computation. Such systems often occur naturally, but they are also frequently the result of approximating nonlinear equations by linear equations or differential equations by algebraic equations. We will see many examples of such approximations throughout this book. For these reasons, the efficient and accurate solution of linear systems forms the cornerstone of many numerical methods for solving a wide variety of practical computational problems.

In matrix-vector notation, a system of linear algebraic equations has the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},$$

where $\boldsymbol{A}$ is an $m \times n$ matrix, $\boldsymbol{b}$ is a given m-vector, and $\boldsymbol{x}$ is the unknown solution n-vector to be determined. Such a system of equations asks the question, "Can the vector $\boldsymbol{b}$ be expressed as a linear combination of the columns of the matrix $\boldsymbol{A}$?" If so, the coefficients of this linear combination are given by the components of the solution vector $\boldsymbol{x}$. There may or may not be a solution; and if there is a solution, it may or may not be unique. In this chapter we will consider only *square* systems, which means that $m = n$, i.e., the matrix has the same number of rows and columns. In later chapters we will consider systems where $m \neq n$.

2.1.1 Singularity and Nonsingularity

An $n \times n$ matrix $\boldsymbol{A}$ is said to be *singular* if it has any one of the following equivalent properties:

1. $\boldsymbol{A}$ has no inverse (i.e, there is no matrix $\boldsymbol{M}$ such that $\boldsymbol{A}\boldsymbol{M} = \boldsymbol{M}\boldsymbol{A} = \boldsymbol{I}$, the identity matrix).
2. $\det(\boldsymbol{A}) = 0$.

3. $\text{rank}(\boldsymbol{A}) < n$ (the *rank* of a matrix is the maximum number of linearly independent rows or columns it contains).
4. $\boldsymbol{Az} = \boldsymbol{o}$ for some vector $\boldsymbol{z} \neq \boldsymbol{o}$.

Otherwise, the matrix is *nonsingular*. The solvability of a system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$ is determined by whether the matrix $\boldsymbol{A}$ is singular or nonsingular. If the matrix $\boldsymbol{A}$ is nonsingular, then its inverse, denoted by $\boldsymbol{A}^{-1}$, exists, and the system $\boldsymbol{Ax} = \boldsymbol{b}$ always has a unique solution $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ regardless of the value for $\boldsymbol{b}$. If, on the other hand, the matrix $\boldsymbol{A}$ is singular, then the *number* of solutions is determined by the right-hand-side vector $\boldsymbol{b}$: for a given value of $\boldsymbol{b}$ there may be no solution, but if there is a solution $\boldsymbol{x}$, so that $\boldsymbol{Ax} = \boldsymbol{b}$, then we also have $\boldsymbol{A}(\boldsymbol{x} + \gamma\boldsymbol{z}) = \boldsymbol{b}$ for any scalar γ, where the vector $\boldsymbol{z}$ is as in the foregoing definition. Thus, if a singular system has a solution, then the solution cannot be unique. To summarize the possibilities, for a given matrix $\boldsymbol{A}$ and right-hand-side vector $\boldsymbol{b}$, the system may have

One solution:	nonsingular
No solution:	singular
Infinitely many solutions:	singular

In two dimensions, each linear equation determines a straight line in the plane. The solution of the system is the intersection point of the two lines. If the two straight lines are not parallel, then they have a unique intersection point (the nonsingular case). If the two straight lines are parallel, then either they do not intersect at all (there is no solution) or the two lines are the same (any point along the line is a solution). In higher dimensions, each equation determines a hyperplane. In the nonsingular case, the unique solution is the intersection point of all of the hyperplanes.

Example 2.1 Singularity and Nonsingularity. The 2×2 system

$$\begin{aligned} 2x_1 + 3x_2 &= b_1, \\ 5x_1 + 4x_2 &= b_2, \end{aligned}$$

or in matrix-vector notation

$$\begin{bmatrix} 2 & 3 \\ 5 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

is nonsingular regardless of the value of b. If $\boldsymbol{b} = [\,8 \quad 13\,]^T$, for example, then the unique solution is $\boldsymbol{x} = [\,1 \quad 2\,]^T$.

The 2×2 system

$$\begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

is singular regardless of the value of $\boldsymbol{b}$. With $\boldsymbol{b} = [\,4 \quad 7\,]^T$, there is no solution. With $\boldsymbol{b} = [\,4 \quad 8\,]^T$, then

$$\boldsymbol{x} = \begin{bmatrix} \gamma \\ (4 - 2\gamma)/3 \end{bmatrix}$$

is a solution for any real number γ.

2.2 Solving Linear Systems

To solve a linear system, the general strategy outlined in Section 1.1.1 suggests that we should transform the system into one whose solution is the same as that of the original system but is easier to compute. What type of transformation of a linear system leaves the solution unchanged? The answer is that we can premultiply (i.e., multiply from the left) both sides of the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by any nonsingular matrix $\boldsymbol{M}$ without affecting the solution. To see why this is so, note that the solution to the linear system $\boldsymbol{M}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}\boldsymbol{b}$ is given by

$$\boldsymbol{x} = (\boldsymbol{M}\boldsymbol{A})^{-1}\boldsymbol{M}\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{M}^{-1}\boldsymbol{M}\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{b}.$$

Example 2.2 Permutations. An important example of such a transformation is the fact that the rows of $\boldsymbol{A}$ and corresponding entries of $\boldsymbol{b}$ can be reordered without changing the solution $\boldsymbol{x}$. This is intuitively obvious: all of the equations in the system must be satisfied simultaneously in any case, so the order in which they happen to be written down is irrelevant; they may as well have been drawn randomly from a hat. Formally, such a reordering of the rows is accomplished by premultiplying both sides of the equation by a *permutation* matrix $\boldsymbol{P}$, which is a square matrix having exactly one 1 in each row and column and zeros elsewhere (i.e., an identity matrix with its rows and columns permuted). For example,

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_3 \\ b_1 \\ b_2 \end{bmatrix}.$$

A permutation matrix is always nonsingular; in fact, its inverse is simply its transpose, $\boldsymbol{P}^{-1} = \boldsymbol{P}^T$ (the *transpose* of a matrix $\boldsymbol{M}$, denoted by $\boldsymbol{M}^T$, is a matrix whose columns are the rows of $\boldsymbol{M}$, that is, if $\boldsymbol{N} = \boldsymbol{M}^T$, then $n_{ij} = m_{ji}$). Thus, the reordered system can be written $\boldsymbol{P}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{P}\boldsymbol{b}$, and the solution $\boldsymbol{x}$ is unchanged.

Postmultiplying (i.e., multiplying from the right) by a permutation matrix reorders the columns of the matrix instead of the rows. Such a transformation *does* change the solution, but only in that the components of the solution are permuted. To see this, observe that the solution to the system $\boldsymbol{A}\boldsymbol{P}\boldsymbol{x} = \boldsymbol{b}$ is given by

$$\boldsymbol{x} = (\boldsymbol{A}\boldsymbol{P})^{-1}\boldsymbol{b} = \boldsymbol{P}^{-1}\boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{P}^T(\boldsymbol{A}^{-1}\boldsymbol{b}).$$

Example 2.3 Diagonal Scaling. Another simple but important type of transformation is *diagonal scaling*. Recall that a matrix $\boldsymbol{D}$ is *diagonal* if $d_{ij} = 0$ for all $i \neq j$, that is, the only nonzero entries are $d_{ii}, i = 1, \ldots, n$, on the *main diagonal*. Premultiplying both sides of a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by a nonsingular diagonal matrix $\boldsymbol{D}$ multiplies each row of the matrix and right-hand side by the corresponding diagonal entry of $\boldsymbol{D}$, and hence is called *row scaling*. In principle, row scaling does not change the solution to the linear system, but in practice it can affect the numerical solution process and the accuracy that can be attained for a given problem, as we will see.

Column scaling—postmultiplying the matrix of a linear system by a nonsingular diagonal matrix $\boldsymbol{D}$—multiplies each column of the matrix by the corresponding diagonal entry of $\boldsymbol{D}$. Such a transformation does alter the solution, in effect changing the units in which the components of the solution are measured. The solution to the scaled system $\boldsymbol{ADx} = \boldsymbol{b}$ is given by

$$\boldsymbol{x} = (\boldsymbol{AD})^{-1}\boldsymbol{b} = \boldsymbol{D}^{-1}\boldsymbol{A}^{-1}\boldsymbol{b},$$

and hence the solution to the original system is given by $\boldsymbol{D}$.

2.2.1 Triangular Linear Systems

The next question is what type of linear system is easy to solve. Suppose there is an equation in the system $\boldsymbol{Ax} = \boldsymbol{b}$ that involves only one of the unknown solution components (i.e., only one entry in that row of $\boldsymbol{A}$ is nonzero). Then that equation can easily be solved (by division) for that unknown. Now suppose there is another equation in the system that involves only two unknowns, one of which is the one already determined. By substituting the one solution component already determined into this second equation, we can then easily solve for its other unknown. If this pattern continues, with only one new unknown component arising per equation, then all of the solution components can be computed in succession. A matrix with this special property is called *triangular*, for reasons that will soon become apparent. Because triangular linear systems are easily solved by this successive substitution process, they are a suitable target in transforming a general linear system.

Although the general triangular form just described is all that is required to enable the system to be solved by successive substitution, it is convenient to define two specific triangular forms for computational purposes. A matrix $\boldsymbol{A}$ is *upper triangular* if all of its entries below the main diagonal are zero (i.e., if $a_{ij} = 0$ for $i > j$). Similarly, a matrix is *lower triangular* if all of its entries above the main diagonal are zero (i.e., if $a_{ij} = 0$ for $i < j$). For an upper triangular system $\boldsymbol{Ax} = \boldsymbol{b}$, the successive substitution process is called *back-substitution* and can be expressed as follows:

$$x_n = b_n/a_{nn},$$

$$x_i = \left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j \right) /a_{ii}, \quad i = n-1, \ldots, 1.$$

Similarly, for a lower triangular system $\boldsymbol{Ax} = \boldsymbol{b}$, the successive substitution process is called *forward-substitution* and can be expressed as follows:

$$x_1 = b_1/a_{11},$$

$$x_i = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j \right) /a_{ii}, \quad i = 2, \ldots, n.$$

A matrix that is triangular in the more general sense defined earlier can be permuted into upper or lower triangular form by a suitable permutation of its rows or columns.

Example 2.4 Triangular Linear System. Consider the upper triangular linear system

$$\begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix}.$$

The last equation, $4x_3 = 8$, can be solved directly for $x_3 = 2$. This value can then be substituted into the second equation to obtain $x_2 = 2$, and finally both x_3 and x_2 are substituted into the first equation to obtain $x_1 = -1$.

2.2.2 Elementary Elimination Matrices

Our strategy then is to devise a nonsingular linear transformation that transforms a given general linear system into a triangular linear system that we can then solve easily by successive substitution. Thus, we need a transformation that replaces selected nonzero entries of the given matrix with zeros. This can be accomplished by taking appropriate linear combinations of the rows of the matrix, as we will now show.

Consider the 2-vector $a = [\, a_1 \quad a_2 \,]^T$. If $a_1 \neq 0$, then

$$\begin{bmatrix} 1 & 0 \\ -a_2/a_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} a_1 \\ 0 \end{bmatrix}.$$

More generally, given an n-vector a, we can annihilate *all* of its entries below the kth position, provided that $a_k \neq 0$, by the following transformation:

$$\boldsymbol{M}_k \boldsymbol{a} = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -m_{k+1} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -m_n & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ a_{k+1} \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $m_i = a_i/a_k$, $i = k+1, \ldots, n$. The divisor a_k is called the *pivot*. A matrix of this form is sometimes called an *elementary elimination matrix* or *Gauss transformation*, and its effect on a vector is to add a multiple of row k to each subsequent row, with the multipliers m_i chosen so that the result in each case is zero. Note the following about these elementary elimination matrices:

1. $\boldsymbol{M}_k$ is a lower triangular matrix with unit main diagonal, and hence it must be nonsingular.
2. $\boldsymbol{M}_k = \boldsymbol{I} - \boldsymbol{m}\boldsymbol{e}_k^T$, where $\boldsymbol{m} = [0, \ldots, 0, m_{k+1}, \ldots, m_n]^T$ and $\boldsymbol{e}_k$ is the kth column of the identity matrix.
3. $\boldsymbol{M}_k^{-1} = \boldsymbol{I} + \boldsymbol{m}\boldsymbol{e}_k^T$, which means that $\boldsymbol{M}_k^{-1}$, which we will denote by $\boldsymbol{L}_k$, is the same as $\boldsymbol{M}_k$ except that the signs of the multipliers are reversed.

4. If $\boldsymbol{M}_j$, $j > k$, is another elementary elimination matrix, with vector of multipliers $\boldsymbol{t}$, then
$$\boldsymbol{M}_k\boldsymbol{M}_j = \boldsymbol{I} - \boldsymbol{m}\boldsymbol{e}_k^T - \boldsymbol{t}\boldsymbol{e}_j^T + \boldsymbol{m}\boldsymbol{e}_k^T\boldsymbol{t}\boldsymbol{e}_j^T = \boldsymbol{I} - \boldsymbol{m}\boldsymbol{e}_k^T - \boldsymbol{t}\boldsymbol{e}_j^T,$$
since $\boldsymbol{e}_k^T\boldsymbol{t} = \boldsymbol{o}$. Thus, their product is essentially their "union." Because they have the same form, a similar result holds for the product of their inverses, $\boldsymbol{L}_k\boldsymbol{L}_j$. Note that the order of multiplication is significant; these results do not hold for the reverse product.

Example 2.5 Elementary Elimination Matrices. If $\boldsymbol{a} = [\,2 \quad 4 \quad -2\,]^T$, then
$$\boldsymbol{M}_1\boldsymbol{a} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \quad \text{and} \quad \boldsymbol{M}_2\boldsymbol{a} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix}.$$
We also note that
$$\boldsymbol{L}_1 = \boldsymbol{M}_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{L}_2 = \boldsymbol{M}_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix},$$
and
$$\boldsymbol{M}_1\boldsymbol{M}_2 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & \frac{1}{2} & 1 \end{bmatrix}, \quad \boldsymbol{L}_1\boldsymbol{L}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -\frac{1}{2} & 1 \end{bmatrix}.$$

2.2.3 Gaussian Elimination and LU Factorization

With elementary elimination matrices, it is a fairly simple matter to reduce a general linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ to upper triangular form. We first choose an elementary elimination matrix $\boldsymbol{M}_1$ according to the recipe given in Section 2.2.2, with the first diagonal entry a_{11} as pivot, so that, when premultiplied by $\boldsymbol{M}_1$, the first column of $\boldsymbol{A}$ becomes zero below the first row. Of course, all of the remaining columns of $\boldsymbol{A}$, as well as the right-hand-side vector $\boldsymbol{b}$, are also multiplied by $\boldsymbol{M}_1$, so the new system becomes $\boldsymbol{M}_1\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}_1\boldsymbol{b}$, but by our previous discussion the solution is unchanged.

Next we use the second diagonal entry as pivot to determine a second elementary elimination matrix $\boldsymbol{M}_2$ that annihilates all of the entries of the second column of the new matrix, $\boldsymbol{M}_1\boldsymbol{A}$, below the second row. Again, $\boldsymbol{M}_2$ must be applied to the entire matrix and right-hand-side vector, so that we obtain the further modified linear system $\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b}$. Note that the first column of the matrix $\boldsymbol{M}_1\boldsymbol{A}$ is not affected by $\boldsymbol{M}_2$ because all of its entries are zero in the relevant rows. This process is continued for each successive column until all of the subdiagonal entries of the matrix have been annihilated, so that the linear system
$$\boldsymbol{M}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{b} = \boldsymbol{M}\boldsymbol{b}$$
is upper triangular and can be solved by back-substitution to obtain the solution to the original linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

The process we have just described is known as *Gaussian elimination*. It is also known as *LU factorization* or *LU decomposition* because it decomposes the matrix $\boldsymbol{A}$ into a product of a unit lower triangular matrix, $\boldsymbol{L}$, and an upper triangular matrix, $\boldsymbol{U}$. To see this, recall that the product $\boldsymbol{L}_k\boldsymbol{L}_j$ is unit lower triangular if $k < j$, so that

$$\boldsymbol{L} = \boldsymbol{M}^{-1} = (\boldsymbol{M}_{n-1} \cdots \boldsymbol{M}_1)^{-1} = \boldsymbol{M}_1^{-1} \cdots \boldsymbol{M}_{n-1}^{-1} = \boldsymbol{L}_1 \cdots \boldsymbol{L}_{n-1}$$

is unit lower triangular. We have already seen that, by design, the matrix $\boldsymbol{U} = \boldsymbol{MA}$ is upper triangular. Therefore, we have expressed $\boldsymbol{A}$ as a product

$$\boldsymbol{A} = \boldsymbol{LU},$$

where $\boldsymbol{L}$ is unit lower triangular and $\boldsymbol{U}$ is upper triangular. Given such a factorization, the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ can then be written as $\boldsymbol{LUx} = \boldsymbol{b}$ and hence can be solved by first solving the lower triangular system $\boldsymbol{Ly} = \boldsymbol{b}$ by forward-substitution, then the upper triangular system $\boldsymbol{Ux} = \boldsymbol{y}$ by back-substitution. Note that the intermediate solution $\boldsymbol{y}$ is the same as the transformed right-hand-side vector, $\boldsymbol{Mb}$, in the previous formulation. Thus, Gaussian elimination and LU factorization are simply two ways of expressing the same solution process.

Example 2.6 Gaussian Elimination. We illustrate Gaussian elimination by solving the linear system

$$\begin{aligned} 2x_1 + 4x_2 - 2x_3 &= 2, \\ 4x_1 + 9x_2 - 3x_3 &= 8, \\ -2x_1 - 3x_2 + 7x_3 &= 10, \end{aligned}$$

or in matrix notation

$$\boldsymbol{Ax} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix} = \boldsymbol{b}.$$

To annihilate the subdiagonal entries of the first column of $\boldsymbol{A}$, we subtract two times the first row from the second row, and add the first row to the third row:

$$\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix},$$

$$\boldsymbol{M}_1\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}.$$

Now to annihilate the subdiagonal entry of the second column of M_1A, we subtract the second row from the third row:

$$\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix},$$

$$\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix}.$$

We have therefore reduced the original system to the equivalent upper triangular system

$$\begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix},$$

which can now be solved by back-substitution (as in Example 2.4) to obtain $\boldsymbol{x} = [\,-1 \quad 2 \quad 2\,]^T$. To write out the LU factorization explicitly, we have

$$\boldsymbol{L} = \boldsymbol{L}_1\boldsymbol{L}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix},$$

so that

$$\boldsymbol{A} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} = \boldsymbol{L}\boldsymbol{U}.$$

2.2.4 Pivoting

There is one obvious problem with the Gaussian elimination process as we have described it, as well as another, somewhat more subtle, problem. The obvious potential difficulty is that the process breaks down if the leading diagonal entry of the remaining unreduced portion of the matrix is zero at any stage, as computing the multipliers m_i for a given column requires division by the diagonal entry in that column. The solution to this problem is almost equally obvious: if the diagonal entry is zero at stage k, then interchange row k of the system with some subsequent row whose entry in column k is nonzero. As we know from Example 2.2, such an interchange does not alter the solution to the system. With a nonzero diagonal entry as pivot, the process can then proceed as usual.

But what if there is no nonzero entry on or below the diagonal in column k? Then there is nothing to do at this stage, since all the entries to be annihilated are already zero, and we can simply move on to the next column (i.e., $\boldsymbol{M}_k = \boldsymbol{I}$). Note that this step leaves a zero on the diagonal, and hence the resulting upper triangular matrix $\boldsymbol{U}$ is singular, but the LU factorization can still be completed. It does mean, however, that the subsequent back-substitution process will fail, since it requires a division by each diagonal entry of $\boldsymbol{U}$, but this is not surprising because the original matrix must have been singular anyway. A more insidious problem is that in floating-point arithmetic we may not get an exact zero, but only a very small diagonal entry, which brings us to the more subtle point.

In principle, any nonzero value will do as the pivot for computing the multipliers, but in practice the choice should be made with some care to minimize error. When the remaining portion of the matrix is multiplied by the resulting elementary elimination matrix, we should try to limit the growth of the entries of the transformed matrix in order not to

amplify rounding errors. For this reason, it is desirable for the multipliers not to exceed 1 in magnitude. This requirement can be met by choosing the entry of largest magnitude on or below the diagonal as pivot. Such a policy is called *partial pivoting*, and it is essential in practice for a numerically stable implementation of Gaussian elimination for general linear systems.

The row interchanges required by partial pivoting slightly complicate the formal description of LU factorization given earlier. In particular, each elementary elimination matrix $\boldsymbol{M}_k$ is preceded by a permutation matrix $\boldsymbol{P}_k$ that interchanges rows to bring the entry of largest magnitude into the diagonal pivot position. We still have $\boldsymbol{MA} = \boldsymbol{U}$, where $\boldsymbol{U}$ is upper triangular, but now

$$\boldsymbol{M} = \boldsymbol{M}_{n-1}\boldsymbol{P}_{n-1} \cdots \boldsymbol{M}_1\boldsymbol{P}_1.$$

$\boldsymbol{M}^{-1}$ is still triangular in the general sense defined earlier, but because of the permutations, $\boldsymbol{M}^{-1}$ is not necessarily *lower* triangular, though we still denote it by $\boldsymbol{L}$. Thus, "LU" factorization no longer literally means "lower times upper" triangular, but it is still equally useful for solving linear systems by successive substitution.

We note that the permutation matrix

$$\boldsymbol{P} = \boldsymbol{P}_{n-1} \cdots \boldsymbol{P}_1$$

permutes the rows of $\boldsymbol{A}$ into the order determined by partial pivoting. An alternative interpretation, therefore, is to think of partial pivoting as a way of determining a row ordering for the system under which no pivoting would be required for numerical stability. Thus, we obtain the factorization

$$\boldsymbol{PA} = \boldsymbol{LU},$$

where now $\boldsymbol{L}$ really is lower triangular. To solve the linear system $\boldsymbol{Ax} = \boldsymbol{b}$, we first solve the lower triangular system $\boldsymbol{Ly} = \boldsymbol{Pb}$ by forward-substitution, then the upper triangular system $\boldsymbol{Ux} = \boldsymbol{y}$ by back-substitution.

The name "partial" pivoting comes from the fact that only the current column is searched for a suitable pivot. A more exhaustive pivoting strategy is *complete pivoting*, in which the entire remaining unreduced submatrix is searched for the largest entry, which is then permuted into the diagonal pivot position. Note that this requires interchanging columns as well as rows, and hence it leads to a factorization of the form

$$\boldsymbol{PAQ} = \boldsymbol{LU},$$

where $\boldsymbol{L}$ is unit lower triangular, $\boldsymbol{U}$ is upper triangular, and $\boldsymbol{P}$ and $\boldsymbol{Q}$ are permutation matrices that reorder the rows and columns, respectively, of $\boldsymbol{A}$. To solve the linear system $\boldsymbol{Ax} = \boldsymbol{b}$, we first solve the lower triangular system $\boldsymbol{Ly} = \boldsymbol{Pb}$ by forward-substitution, then the upper triangular system $\boldsymbol{Uz} = \boldsymbol{y}$ by back-substitution, and finally we permute the solution components to obtain $\boldsymbol{x} = \boldsymbol{Qz}$. Although the numerical stability of complete pivoting is theoretically superior, it requires a much more expensive pivot search than partial pivoting. Because the numerical stability of partial pivoting is more than adequate in practice, it is almost universally used in solving linear systems by Gaussian elimination.

Since pivot selection depends on the magnitudes of individual matrix entries, the particular choice obviously depends on the scaling of the matrix. A diagonal scaling of the

matrix (recall Example 2.3) may result in a different sequence of pivots. For example, any nonzero entry in a given column can be made the largest in magnitude simply by giving that row a sufficiently heavy weighting. This does not mean that an arbitrary pivot sequence is acceptable, however: a badly skewed scaling can result in an inherently sensitive system and a correspondingly inaccurate solution. A well-formulated problem should have appropriately commensurate units for measuring the unknown variables (column scaling), and a weighting of the individual equations that properly reflects their relative importance (row scaling). It should also account for the relative accuracy of the input data. Under these circumstances, the pivoting procedure will usually produce a solution that is as accurate as the problem warrants (see Section 2.4).

Example 2.7 Pivoting. Here are some examples to illustrate the necessity of pivoting, both in theory and practice, for a stable implementation of Gaussian elimination. We first observe that the need for pivoting has nothing to do with whether the matrix is singular or nearly singular. For example, the matrix

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

is nonsingular yet has no LU factorization unless we interchange rows, whereas the singular matrix

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

does have an LU factorization.

In practice, using finite-precision arithmetic, we must avoid not only zero pivots but also *small* pivots in order to prevent unacceptable error growth, as shown in the following example. Let

$$\boldsymbol{A} = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix},$$

where ϵ is a positive number smaller than the unit roundoff ϵ_{mach} in a given floating-point system. If we do not interchange rows, then the pivot is ϵ and the resulting multiplier is $-1/\epsilon$, so that we get the elimination matrix

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 \\ -1/\epsilon & 1 \end{bmatrix},$$

and hence

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{U} = \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - 1/\epsilon \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix}$$

in floating-point arithmetic. But then

$$\boldsymbol{LU} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix} \neq \boldsymbol{A}.$$

Using a small pivot, and a correspondingly large multiplier, has caused an unrecoverable loss of information in the transformed matrix. If we interchange rows, on the other hand, then the pivot is 1 and the resulting multiplier is $-\epsilon$, so that we get the elimination matrix

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 \\ -\epsilon & 1 \end{bmatrix},$$

and hence

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{U} = \begin{bmatrix} 1 & 1 \\ 0 & 1-\epsilon \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

in floating-point arithmetic. We therefore have

$$\boldsymbol{LU} = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix},$$

which is the correct result after permutation.

Although the foregoing example is rather extreme, the principle holds in general that larger pivots produce smaller multipliers and hence smaller errors. In particular, if the largest entry on or below the diagonal in each column is used as pivot (partial pivoting), then the multipliers are bounded in magnitude by 1. In Example 2.6, we did not use row interchanges, and some of the multipliers were greater than 1. For illustration, we now repeat that example, this time using partial pivoting. The system in Example 2.6 is

$$\begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix}.$$

The largest entry in the first column is 4, so we interchange the first two rows using the permutation matrix

$$\boldsymbol{P}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

obtaining the permuted system

$$\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 4 & 9 & -3 \\ 2 & 4 & -2 \\ -2 & -3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \\ 10 \end{bmatrix} = \boldsymbol{P}_1\boldsymbol{b}.$$

To annihilate the subdiagonal entries of the first column, we use the elimination matrix

$$\boldsymbol{M}_1 = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{bmatrix},$$

obtaining the transformed system

$$\boldsymbol{M}_1\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 4 & 9 & -3 \\ 0 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{3}{2} & \frac{11}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -2 \\ 14 \end{bmatrix} = \boldsymbol{M}_1\boldsymbol{P}_1\boldsymbol{b}.$$

The largest entry in the second column on or below the diagonal is $\frac{3}{2}$, so we interchange the last two rows using the permutation matrix

$$\boldsymbol{P}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

obtaining the permuted system

$$\boldsymbol{P}_2\boldsymbol{M}_1\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 4 & 9 & -3 \\ 0 & \frac{3}{2} & \frac{11}{2} \\ 0 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 14 \\ -2 \end{bmatrix} = \boldsymbol{P}_2\boldsymbol{M}_1\boldsymbol{P}_1\boldsymbol{b}.$$

To annihilate the subdiagonal entry of the second column, we use the elimination matrix

$$\boldsymbol{M}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{3} & 1 \end{bmatrix},$$

obtaining the transformed system

$$\boldsymbol{M}_2\boldsymbol{P}_2\boldsymbol{M}_1\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 4 & 9 & -3 \\ 0 & \frac{3}{2} & \frac{11}{2} \\ 0 & 0 & \frac{4}{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 14 \\ \frac{8}{3} \end{bmatrix} = \boldsymbol{M}_2\boldsymbol{P}_2\boldsymbol{M}_1\boldsymbol{P}_1\boldsymbol{b}.$$

We have therefore reduced the original system to an equivalent upper triangular system, which can now be solved by back-substitution to obtain the same answer as before.

To write out the LU factorization explicitly, we have

$$\boldsymbol{L} = \boldsymbol{M}^{-1} = (\boldsymbol{M}_2\boldsymbol{P}_2\boldsymbol{M}_1\boldsymbol{P}_1)^{-1} = \boldsymbol{P}_1^T\boldsymbol{L}_1\boldsymbol{P}_2^T\boldsymbol{L}_2 =$$

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

and hence

$$\boldsymbol{A} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{3} & 1 \\ 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 9 & -3 \\ 0 & \frac{3}{2} & \frac{11}{2} \\ 0 & 0 & \frac{4}{3} \end{bmatrix} = \boldsymbol{L}\boldsymbol{U}.$$

Note that $\boldsymbol{L}$ is not lower triangular, but it is triangular in the more general sense (it is a permutation of a lower triangular matrix). Alternatively, we can take

$$\boldsymbol{P} = \boldsymbol{P}_2\boldsymbol{P}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix},$$

and

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

so that

$$\boldsymbol{P}\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & -\frac{1}{3} & 1 \end{bmatrix} \begin{bmatrix} 4 & 9 & -3 \\ 0 & \frac{3}{2} & \frac{11}{2} \\ 0 & 0 & \frac{4}{3} \end{bmatrix} = \boldsymbol{L}\boldsymbol{U},$$

where $\boldsymbol{L}$ now really is lower triangular but $\boldsymbol{A}$ is permuted.

As we have just seen, pivoting is generally required for Gaussian elimination to be stable. There are some classes of matrices, however, for which Gaussian elimination is stable *without* pivoting. For example, if the matrix $\boldsymbol{A}$ is *diagonally dominant* by columns, which means that each diagonal entry is larger in magnitude than the sum of the magnitudes of the other entries in its column,

$$\sum_{i=1,\, i\neq j}^{n} |a_{ij}| < |a_{jj}|, \quad j = 1, \ldots, n,$$

then pivoting is not required in computing its LU factorization by Gaussian elimination. If partial pivoting is used on such a matrix, then no row interchanges will actually occur. Another important class for which pivoting is not required is matrices that are symmetric and positive definite, which will be defined in Section 2.5. Avoiding an unnecessary pivot search can save a significant amount of time in computing the factorization.

2.2.5 Implementation of Gaussian Elimination

Gaussian elimination, or LU factorization, has the general form of a triple-nested loop,

```
for _____
    for _____
        for _____
            a_ij = a_ij − (a_ik/a_kk)a_kj
        end
    end
end
```

where the indices i, j, and k of the **for** loops can be taken in any order, for a total of $3! = 6$ different ways of arranging the loops. Some of the indicated arithmetic operations can be moved outside the innermost loop for greater efficiency, depending on the specific indices involved, and additional reorderings of the operations that do not have strictly nested loops are also possible. These variations of the basic algorithm have different memory access patterns (e.g., accessing memory row-wise or column-wise), and also differ in their ability to take advantage of the architectural features of a given computer (e.g., cache, paging, vectorization, multiple processors). Thus, their performance may vary widely on a given computer or across different computers, and no single arrangement may be uniformly superior.

Numerous implementation details of the algorithm are subject to variation in this way. For example, the partial pivoting procedure we described searches along columns and interchanges rows, but alternatively, one could search along rows and interchange columns. We have also taken $\boldsymbol{L}$ to have unit diagonal, but one could instead arrange for $\boldsymbol{U}$ to have unit diagonal. Some of these variations of Gaussian elimination are of sufficient importance to have been given names, such as the Crout and Doolittle methods.

Although the many possible variations on Gaussian elimination may have a dramatic effect on performance, they all produce essentially the same factorization. Provided the row pivot sequence is the same, if we have two LU factorizations $\boldsymbol{PA} = \boldsymbol{LU} = \hat{\boldsymbol{L}}\hat{\boldsymbol{U}}$, then

this expression implies that $\hat{\boldsymbol{L}}^{-1}\boldsymbol{L} = \hat{\boldsymbol{U}}\boldsymbol{U}^{-1} = \boldsymbol{D}$ is both lower and upper triangular, and hence diagonal. If both $\boldsymbol{L}$ and $\hat{\boldsymbol{L}}$ are assumed to be unit lower triangular, then $\boldsymbol{D}$ must in fact be the identity matrix $\boldsymbol{I}$, and hence $\boldsymbol{L} = \hat{\boldsymbol{L}}$ and $\boldsymbol{U} = \hat{\boldsymbol{U}}$, so that the factorization is unique. Even without this assumption, however, we may still conclude that that the LU factorization is unique up to diagonal scaling of the factors. This uniqueness is made explicit in the LDU factorization $\boldsymbol{PA} = \boldsymbol{LDU}$, where $\boldsymbol{L}$ is unit lower triangular, $\boldsymbol{U}$ is unit upper triangular, and $\boldsymbol{D}$ is diagonal.

Storage management is another important implementation issue. The numerous matrices we considered—the elementary elimination matrices $\boldsymbol{M}_k$, their inverses $\boldsymbol{L}_k$, and the permutation matrices $\boldsymbol{P}_k$—merely describe the factorization process formally. They are not formed explicitly in an actual implementation. To conserve storage, the $\boldsymbol{L}$ and $\boldsymbol{U}$ factors overwrite the initial storage for the input matrix $\boldsymbol{A}$, with the transformed matrix $\boldsymbol{U}$ occupying the upper triangle of $\boldsymbol{A}$ (including the diagonal), and the multipliers that make up the strict lower triangle of $\boldsymbol{L}$ occupying the (now zero) strict lower triangle of $\boldsymbol{A}$. The unit diagonal of $\boldsymbol{L}$ need not be stored.

To minimize data movement, the row interchanges required by pivoting are not usually carried out explicitly. Instead, the rows remain in their original locations, and an auxiliary integer vector is used to keep track of the new row order. Note that a single such vector suffices, because the net effect of all of the interchanges is still just a permutation of the integers $1, \ldots, n$.

2.2.6 Complexity of Solving Linear Systems

The Gaussian elimination process for computing the LU factorization requires about $n^3/3$ floating-point multiplications and a similar number of additions. Solving the resulting triangular system for a single right-hand-side vector by forward- and back-substitution requires about n^2 multiplications and a similar number of additions. Thus, as the order n of the matrix grows, the LU factorization phase becomes increasingly dominant in the cost of solving linear systems.

We can also solve a linear system by explicitly inverting the matrix so that the solution is given by $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. But computing $\boldsymbol{A}^{-1}$ is tantamount to solving n linear systems: it requires an LU factorization of $\boldsymbol{A}$ followed by n forward- and back-substitutions, one for each column of the identity matrix. The total operation count is about n^3 multiplications and a similar number of additions (taking advantage of the zeros in the right-hand-side vectors for the forward-substitution).

Thus, explicit inversion is three times as expensive as LU factorization. The subsequent matrix-vector multiplication $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ to solve a linear system requires about n^2 multiplications and a similar number of additions, which is similar to the total cost of forward- and back-substitution. Hence, even for multiple right-hand-side vectors, matrix inversion is more costly than LU factorization for solving linear systems. In addition, explicit inversion gives a less accurate answer. As a simple example, if we solve the 1×1 linear system $3x = 18$ by division, we get $x = 18/3 = 6$, but explicit inversion would give $x = 3^{-1} \times 18 = 0.333 \times 18 = 5.99$ using three-digit arithmetic. In this small example, inversion requires an additional arithmetic operation and obtains a less accurate result. The disadvantages of inversion become worse as the size of the system grows.

Explicit matrix inverses often occur as a convenient notation in various formulas, but this practice does not mean that an explicit inverse is required to implement such a formula. One merely need solve a linear system with an appropriate right-hand side, which might itself be a matrix. Thus, for example, a product of the form $\boldsymbol{A}^{-1}\boldsymbol{B}$ should be computed by LU factorization of $\boldsymbol{A}$, followed by forward- and back-substitutions using each column of $\boldsymbol{B}$. It is extremely rare in practice that an explicit matrix inverse is actually needed, so whenever you see a matrix inverse in a formula, you should think "solve a system" rather than "invert a matrix."

Another method for solving linear systems that should be avoided is *Cramer's rule*, in which each component of the solution is computed as a ratio of determinants. Though often taught in elementary linear algebra courses, this method is astronomically expensive for full matrices of nontrivial size. Cramer's rule is useful mostly as a theoretical tool.

2.2.7 Gauss-Jordan Elimination

The motivation for Gaussian elimination is to reduce a general matrix to triangular form, because the resulting linear system is easy to solve. Diagonal linear systems are even easier to solve, however, so diagonal form would appear to be an even more desirable target. *Gauss-Jordan elimination* is a variation of standard Gaussian elimination in which the matrix is reduced to diagonal form rather than merely to triangular form. The same type of row combinations are used to eliminate matrix entries as in standard Gaussian elimination, but they are applied to annihilate entries above as well as below the diagonal. Thus, the elimination matrix used for a given column vector a is of the form

$$\begin{bmatrix} 1 & \cdots & 0 & -m_1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & -m_{k-1} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -m_{k+1} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & -m_n & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{k-1} \\ a_k \\ a_{k+1} \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_k \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $m_i = a_i/a_k$, $i = 1, \ldots, n$. This process requires about $n^3/2$ multiplications and a similar number of additions, which is 50 percent more expensive than standard Gaussian elimination.

During the elimination phase, the same row operations are also applied to the right-hand-side vector (or vectors) of a system of linear equations. Once the elimination phase has been completed and the matrix is in diagonal form, then the components of the solution to the linear system can be computed simply by dividing each entry of the transformed right-hand side by the corresponding diagonal entry of the matrix. This computation requires a total of only n divisions, which is significantly cheaper than solving a triangular system, but not enough to make up for the more costly elimination phase. Gauss-Jordan elimination also has the numerical disadvantage that the multipliers can exceed 1 in magnitude even if pivoting is used.

Despite its higher overall cost, Gauss-Jordan elimination may be preferred in some situations because of the extreme simplicity of its final solution phase. For example, it is

occasionally advocated for implementation on parallel computers because it has a uniform workload throughout the factorization phase, and then all of the solution components can be computed simultaneously rather than one at a time as in ordinary back-substitution.

Gauss-Jordan elimination is also sometimes used to compute the inverse of a matrix explicitly, if desired. If the right-hand-side matrix is initialized to be the identity matrix $\boldsymbol{I}$ and the given matrix $\boldsymbol{A}$ is reduced to the identity matrix by Gauss-Jordan elimination, then the transformed right-hand-side matrix will be the inverse of $\boldsymbol{A}$. For computing the inverse, Gauss-Jordan elimination has about the same operation count as explicit inversion by Gaussian elimination followed by forward- and back-substitution.

Example 2.8 Gauss-Jordan Elimination. We illustrate Gauss-Jordan elimination by using it to compute the inverse of the matrix of Example 2.6. For simplicity, we omit pivoting. We begin with the matrix $\boldsymbol{A}$, augmented by the identity matrix $\boldsymbol{I}$ as right-hand side, and repeatedly apply elimination matrices to annihilate off-diagonal entries of $\boldsymbol{A}$ until we reach diagonal form, then scale by the remaining diagonal entries to produce the identity matrix on the left, and hence the inverse matrix on the right.

$$\left[\begin{array}{ccc} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right]\left[\begin{array}{ccc|ccc} 2 & 4 & -2 & 1 & 0 & 0 \\ 4 & 9 & -3 & 0 & 1 & 0 \\ -2 & -3 & 7 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} 2 & 4 & -2 & 1 & 0 & 0 \\ 0 & 1 & 1 & -2 & 1 & 0 \\ 0 & 1 & 5 & 1 & 0 & 1 \end{array}\right],$$

$$\left[\begin{array}{ccc} 1 & -4 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{array}\right]\left[\begin{array}{ccc|ccc} 2 & 4 & -2 & 1 & 0 & 0 \\ 0 & 1 & 1 & -2 & 1 & 0 \\ 0 & 1 & 5 & 1 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} 2 & 0 & -6 & 9 & -4 & 0 \\ 0 & 1 & 1 & -2 & 1 & 0 \\ 0 & 0 & 4 & 3 & -1 & 1 \end{array}\right],$$

$$\left[\begin{array}{ccc} 1 & 0 & \frac{3}{2} \\ 0 & 1 & -\frac{1}{4} \\ 0 & 0 & 1 \end{array}\right]\left[\begin{array}{ccc|ccc} 2 & 0 & -6 & 9 & -4 & 0 \\ 0 & 1 & 1 & -2 & 1 & 0 \\ 0 & 0 & 4 & 3 & -1 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} 2 & 0 & 0 & \frac{27}{2} & -\frac{11}{2} & \frac{3}{2} \\ 0 & 1 & 0 & -\frac{11}{4} & \frac{5}{4} & -\frac{1}{4} \\ 0 & 0 & 4 & 3 & -1 & 1 \end{array}\right],$$

$$\left[\begin{array}{ccc} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{4} \end{array}\right]\left[\begin{array}{ccc|ccc} 2 & 0 & 0 & \frac{27}{2} & -\frac{11}{2} & \frac{3}{2} \\ 0 & 1 & 0 & -\frac{11}{4} & \frac{5}{4} & -\frac{1}{4} \\ 0 & 0 & 4 & 3 & -1 & 1 \end{array}\right] =$$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{27}{4} & -\frac{11}{4} & \frac{3}{4} \\ 0 & 1 & 0 & -\frac{11}{4} & \frac{5}{4} & -\frac{1}{4} \\ 0 & 0 & 1 & \frac{3}{4} & -\frac{1}{4} & \frac{1}{4} \end{array}\right], \quad \text{so} \quad A^{-1} = \frac{1}{4}\left[\begin{array}{ccc} 27 & -11 & 3 \\ -11 & 5 & -1 \\ 3 & -1 & 1 \end{array}\right].$$

2.2.8 Solving Modified Problems

In many practical situations linear systems do not occur in isolation but as part of a sequence of related problems that change in some systematic way. For example, one may need to solve a sequence of linear systems $\boldsymbol{Ax} = \boldsymbol{b}$ having the same matrix $\boldsymbol{A}$ but different right-hand sides $\boldsymbol{b}$. After having solved the initial system by Gaussian elimination, then the $\boldsymbol{L}$ and $\boldsymbol{U}$ factors already computed can be used to solve the additional systems by forward- and back-substitution. The factorization phase need not be repeated in solving subsequent linear systems unless the matrix changes. This procedure represents a substantial savings

in work, since additional triangular solutions cost only $\mathcal{O}(n^2)$ work, in contrast to the $\mathcal{O}(n^3)$ cost of a factorization.

In fact, in some important special cases a new factorization can be avoided even when the matrix does change. One such case that arises frequently is the addition of a matrix that is an outer product $\boldsymbol{u}\boldsymbol{v}^T$ of two nonzero vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. This is called a *rank-one change* because the outer product matrix $\boldsymbol{u}\boldsymbol{v}^T$ has rank one (i.e., only one linearly independent row or column), and any rank-one matrix can be expressed as such an outer product of two vectors. For example, if a single entry of the matrix $\boldsymbol{A}$ changes (say the (j,k) entry changes from a_{jk} to $\tilde{a}_{jk}$), then the new matrix is $\boldsymbol{A} - \alpha \boldsymbol{e}_j \boldsymbol{e}_k^T$, where $\boldsymbol{e}_j$ and $\boldsymbol{e}_k$ are the corresponding columns of the identity matrix and $\alpha = a_{jk} - \tilde{a}_{jk}$.

The *Sherman-Morrison formula* gives the inverse of a matrix resulting from a rank-one change to a matrix whose inverse is already known:

$$(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{u}(1 - \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u})^{-1}\boldsymbol{v}^T\boldsymbol{A}^{-1},$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are n-vectors. Evaluation of this formula requires only $\mathcal{O}(n^2)$ work (for matrix-vector multiplications) rather than the $\mathcal{O}(n^3)$ work normally required for inversion.

To solve a linear system $(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)\boldsymbol{x} = \boldsymbol{b}$ with the new matrix, we could use the foregoing formula to obtain

$$\boldsymbol{x} = (\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)^{-1}\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{b} + \boldsymbol{A}^{-1}\boldsymbol{u}(1 - \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u})^{-1}\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{b}.$$

We would prefer to avoid explicit inversion altogether, however. If we have an LU factorization for $\boldsymbol{A}$, then the following steps can easily be computed to obtain the solution to the modified system:

1. Solve $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{u}$ for $\boldsymbol{z}$, so that $\boldsymbol{z} = \boldsymbol{A}^{-1}\boldsymbol{u}$.
2. Solve $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y}$, so that $\boldsymbol{y} = \boldsymbol{A}^{-1}\boldsymbol{b}$.
3. Compute $\boldsymbol{x} = \boldsymbol{y} + [(\boldsymbol{v}^T\boldsymbol{y})/(1 - \boldsymbol{v}^T\boldsymbol{z})]\boldsymbol{z}$.

Note that the first step is independent of $\boldsymbol{b}$ and hence need not be repeated if there are multiple right-hand-side vectors $\boldsymbol{b}$. Again, this procedure requires only triangular solutions and inner products, so it requires only $\mathcal{O}(n^2)$ work and no explicit inverses.

The *Woodbury formula*, in which $\boldsymbol{u}$ and $\boldsymbol{v}$ become $n \times k$ matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, generalizes the Sherman-Morrison formula to a rank-k change in the matrix:

$$(\boldsymbol{A} - \boldsymbol{U}\boldsymbol{V}^T)^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{I} - \boldsymbol{V}^T\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}^T\boldsymbol{A}^{-1}.$$

Using similar techniques, it is possible to update the factorization rather than the inverse or the solution. Caution must be exercised in using these updating formulas, however, because in general there is no guarantee of numerical stability through successive updates as the matrix changes.

Example 2.9 Rank-One Updating of Solutions. To illustrate the use of the Sherman-Morrison formula, we solve the linear system

$$\begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -1 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix},$$

which is a rank-one modification of the system in Example 2.6 (only the $(3,2)$ entry has changed). We take $\boldsymbol{A}$ to be the matrix of Example 2.6, so we can use the LU factorization already computed. One way to choose the update vectors is

$$\boldsymbol{u} = \begin{bmatrix} 0 \\ 0 \\ -2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{v} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},$$

so that the matrix of the new system is $\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T$, and the right-hand-side vector $\boldsymbol{b}$ has not changed. We can use the previously computed LU factorization of $\boldsymbol{A}$ to solve $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{u}$ to obtain $\boldsymbol{z} = [\,-\frac{3}{2} \quad \frac{1}{2} \quad -\frac{1}{2}\,]^T$, and we had already solved $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$ to obtain $\boldsymbol{y} = [\,-1 \quad 2 \quad 2\,]^T$. The final step is then to compute the updated solution

$$\boldsymbol{x} = \boldsymbol{y} + \frac{\boldsymbol{v}^T\boldsymbol{y}}{1 - \boldsymbol{v}^T\boldsymbol{z}}\,\boldsymbol{z} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} + \frac{2}{1 - \frac{1}{2}} \begin{bmatrix} -\frac{3}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} -7 \\ 4 \\ 0 \end{bmatrix}.$$

We have thus computed the solution to the new system without refactoring the modified matrix.

2.3 Norms and Condition Numbers

2.3.1 Vector Norms

To measure errors and sensitivity in solving linear systems, we need some notion of the "size" of vectors and matrices. The scalar concept of magnitude, modulus, or absolute value can be generalized to the concept of *norms* for vectors and matrices. Although a more general definition is possible, all of the vector norms we will use are instances of p-norms, which for an integer $p > 0$ and a vector $\boldsymbol{x}$ of dimension n are defined by

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

Important special cases are as follows:

- 1-norm:
$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|,$$
sometimes called the Manhattan norm because in the plane it corresponds to the distance between two points as measured in "city blocks."
- 2-norm:
$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2},$$
which corresponds to the usual notion of distance in Euclidean space.

- ∞-norm:

$$\|\boldsymbol{x}\|_\infty = \max_{1\le i\le n} |x_i|,$$

  which can be viewed as a limiting case as $p \to \infty$.

All of these norms give the same results qualitatively, but in certain circumstances a particular norm may be easiest to work with analytically or computationally. The 1-norm or the ∞-norm is usually used in analyzing the sensitivity of solutions to linear systems. We will also use the 2-norm later on in other contexts. The differences among these norms are illustrated in Fig. 2.1, which shows the unit sphere, $\{\boldsymbol{x}: \|\boldsymbol{x}\| = 1\}$, in two dimensions for each norm.

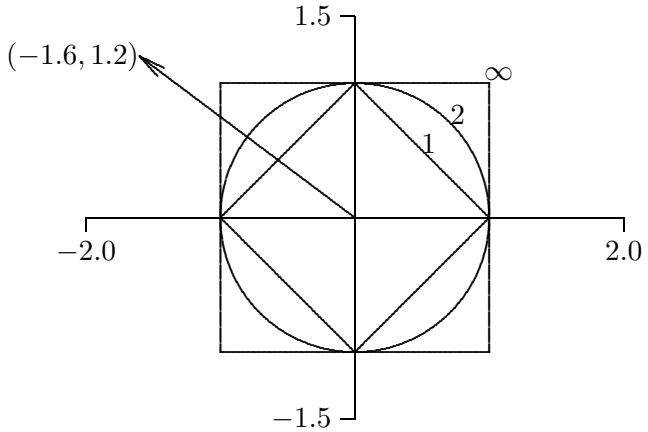


Figure 2.1: The unit sphere in various vector norms.

The norm of a vector is simply the factor by which the corresponding unit sphere must be expanded or shrunk to encompass the vector. For example, the norms have the following values for the vector shown in Fig. 2.1:

$$\left\| \begin{bmatrix} -1.6 \\ 1.2 \end{bmatrix} \right\|_1 = 2.8, \quad \left\| \begin{bmatrix} -1.6 \\ 1.2 \end{bmatrix} \right\|_2 = 2.0, \quad \left\| \begin{bmatrix} -1.6 \\ 1.2 \end{bmatrix} \right\|_\infty = 1.6.$$

In general, for any vector $\boldsymbol{x}$ in $\mathbb{R}^n$, we have

$$\|\boldsymbol{x}\|_1 \ge \|\boldsymbol{x}\|_2 \ge \|\boldsymbol{x}\|_\infty.$$

On the other hand, we also have

$$\|\boldsymbol{x}\|_1 \le \sqrt{n}\,\|\boldsymbol{x}\|_2, \quad \|\boldsymbol{x}\|_2 \le \sqrt{n}\,\|\boldsymbol{x}\|_\infty, \quad \text{and} \quad \|\boldsymbol{x}\|_1 \le n\,\|\boldsymbol{x}\|_\infty.$$

Thus, for a given n, any two of the norms differ by at most a constant, so they are all equivalent in the sense that if one is small, they must all be proportionally small. Hence, we can choose whichever norm is most convenient in a given context. In the remainder of this book, an appropriate subscript will be used to indicate a specific norm, when necessary, but the subscript will be omitted when it does not matter which particular norm is used.

For any vector norm, the following important properties hold, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are any vectors:

1. $\|\boldsymbol{x}\| > 0$ if $\boldsymbol{x} \neq \boldsymbol{o}$.
2. $\|\gamma\boldsymbol{x}\| = |\gamma| \cdot \|\boldsymbol{x}\|$ for any scalar γ.
3. $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ (triangle inequality).

In a more general treatment, these three properties can be taken as the *definition* of a vector norm. A useful variation on the triangle inequality is

$$\|\boldsymbol{x} - \boldsymbol{y}\| \geq \|\boldsymbol{x}\| - \|\boldsymbol{y}\|.$$

2.3.2 Matrix Norms

We also need some way to measure the size or magnitude of matrices. Again, a more general definition is possible, but all of the matrix norms we will use are defined in terms of an underlying vector norm. Specifically, given a vector norm, we define the corresponding matrix norm of a matrix $\boldsymbol{A}$ as follows:

$$\|\boldsymbol{A}\| = \max_{\boldsymbol{x} \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}.$$

Such a matrix norm is said to be *subordinate* to the vector norm. Intuitively, the norm of a matrix measures the maximum stretching the matrix does to any vector, as measured in the given vector norm.

Some matrix norms are much easier to compute than others. For example, the matrix norm corresponding to the vector 1-norm is simply the maximum absolute column sum of the matrix,

$$\|\boldsymbol{A}\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|,$$

and the matrix norm corresponding to the vector ∞-norm is simply the maximum absolute row sum of the matrix,

$$\|\boldsymbol{A}\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|.$$

A handy way to remember these is that the matrix norms agree with the corresponding vector norms for an $n \times 1$ matrix. Unfortunately, the matrix norm corresponding to the vector 2-norm is not so simple to compute; it turns out to be equal to the square root of the largest eigenvalue of the matrix $\boldsymbol{A}^T\boldsymbol{A}$, or, as we shall see later, the largest singular value of $\boldsymbol{A}$ (see Section 4.5.2).

The matrix norms we have defined satisfy the following important properties, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are any matrices:

1. $\|\boldsymbol{A}\| > 0$ if $\boldsymbol{A} \neq \boldsymbol{O}$.
2. $\|\gamma\boldsymbol{A}\| = |\gamma| \cdot \|\boldsymbol{A}\|$ for any scalar γ.
3. $\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$.
4. $\|\boldsymbol{A}\boldsymbol{B}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{B}\|$.
5. $\|\boldsymbol{A}\boldsymbol{x}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{x}\|$ for any vector x.

In a more general treatment, the first three properties can be taken as the *definition* of a matrix norm. The remaining two properties, known as *submultiplicative* or *consistency* conditions, may or may not hold for these more general matrix norms, but they always hold for the matrix norms subordinate to the vector p-norms.

2.3.3 Condition Number of a Matrix

The *condition number* of a square nonsingular matrix $\boldsymbol{A}$ with respect to a given norm is defined as

$$\text{cond}(\boldsymbol{A}) = \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\|.$$

By convention, $\text{cond}(\boldsymbol{A}) = \infty$ if $\boldsymbol{A}$ is singular. Since

$$\|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\| = \left(\max_{\boldsymbol{x} \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \right) \cdot \left(\min_{\boldsymbol{x} \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \right)^{-1},$$

the condition number of a matrix measures the ratio of the maximum stretching that the matrix does to any nonzero vector to the maximum shrinking. We will see in Section 2.4.2 that this concept is consistent with the general notion of condition number defined in Section 1.2.5: the condition number of the matrix bounds the ratio of the relative change in the solution of a linear system to a given relative change in the input data.

The condition number is a measure of how close a matrix is to being singular: a matrix with a large condition number (which we will quantify in Section 2.4.2) is nearly singular, whereas a matrix with a condition number close to 1 is far from being singular. Note that the determinant of a matrix is *not* a good indicator of near singularity: although a matrix $\boldsymbol{A}$ is singular if $\det(\boldsymbol{A}) = 0$, the magnitude of a nonzero determinant, large or small, gives no information on how close to singular the matrix may be. For example, $\det(\alpha \boldsymbol{I}_n) = \alpha^n$, which can be arbitrarily small for $|\alpha| < 1$, yet the matrix is perfectly well-conditioned for any nonzero α, with a condition number of 1.

Some important properties of the condition number are

1. For any matrix $\boldsymbol{A}$, $\text{cond}(\boldsymbol{A}) \geq 1$.
2. For the identity matrix, $\text{cond}(\boldsymbol{I}) = 1$.
3. For any permutation matrix $\boldsymbol{P}$, $\text{cond}(\boldsymbol{P}) = 1$.
4. For any matrix $\boldsymbol{A}$ and nonzero scalar γ, $\text{cond}(\gamma \boldsymbol{A}) = \text{cond}(\boldsymbol{A})$.
5. For any diagonal matrix $\boldsymbol{D} = \text{diag}(d_i)$, $\text{cond}(\boldsymbol{D}) = (\max |d_i|)/(\min |d_i|)$.

As we will see shortly, the usefulness of the condition number is in assessing the accuracy of solutions to linear systems. Since the definition of the condition number involves the inverse of the matrix, computing its value is obviously a nontrivial task. In fact, to compute the condition number literally would require substantially more work than solving the linear system whose accuracy is to be assessed using the condition number. In practice, therefore, the condition number is merely estimated, to perhaps within an order of magnitude, as a relatively inexpensive byproduct of the solution process.

The matrix norm $\|\boldsymbol{A}\|$ is easily computed as the maximum absolute column sum (or row sum, depending on the norm used). It is estimating $\|\boldsymbol{A}^{-1}\|$ at low cost that presents a

challenge. From the properties of norms, we know that if $\boldsymbol{z}$ is the solution to $\boldsymbol{Az} = \boldsymbol{y}$, then

$$\frac{\|\boldsymbol{z}\|}{\|\boldsymbol{y}\|} \leq \|\boldsymbol{A}^{-1}\|,$$

and the bound is achieved for some optimally chosen vector $\boldsymbol{y}$. We thus wish to pick a vector $\boldsymbol{y}$ so that the ratio $\|\boldsymbol{z}\|/\|\boldsymbol{y}\|$ is as large as possible and therefore is a reasonable estimate for $\|\boldsymbol{A}^{-1}\|$. Finding the optimal $\boldsymbol{y}$ would be prohibitively expensive, but a useful approximation can be obtained much more cheaply. One heuristic is to choose $\boldsymbol{y}$ as the solution to the system $\boldsymbol{A}^T\boldsymbol{y} = \boldsymbol{c}$, where $\boldsymbol{c}$ is a vector whose components are ± 1, with the signs chosen successively to make the resulting $\boldsymbol{y}$ as large as possible.

The motivation for this approach may not be obvious now, but it is essentially equivalent to one step of inverse iteration for computing the singular vector corresponding to the smallest singular value of $\boldsymbol{A}$ (see Chapter 4). An alternative approach to condition estimation is to treat it as a convex optimization problem that can be solved very efficiently in practice using a heuristic algorithm. Most good modern software packages for solving linear systems provide an efficient and reliable condition estimator, based on a sophisticated implementation of one of the methods outlined here.

2.4 Accuracy of Solutions

2.4.1 Residual of a Solution

Intuitively, the most obvious way to check the validity of a solution is to substitute it into the equation to see how closely the two sides match. The *residual vector* of an approximate solution $\hat{\boldsymbol{x}}$ to the $n \times n$ linear system $\boldsymbol{Ax} = \boldsymbol{b}$ is defined as

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}.$$

In theory, if $\boldsymbol{A}$ is nonsingular, then the error $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\| = 0$ if and only if $\|\boldsymbol{r}\| = 0$. In practice, however, these quantities are not necessarily small simultaneously. If the computed solution $\hat{\boldsymbol{x}}$ exactly satisfies

$$(\boldsymbol{A} + \boldsymbol{E})\hat{\boldsymbol{x}} = \boldsymbol{b},$$

then

$$\|\boldsymbol{r}\| = \|\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}\| = \|\boldsymbol{E}\hat{\boldsymbol{x}}\| \leq \|\boldsymbol{E}\| \cdot \|\hat{\boldsymbol{x}}\|,$$

so that we have the inequality

$$\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{A}\| \cdot \|\hat{\boldsymbol{x}}\|} \leq \frac{\|\boldsymbol{E}\|}{\|\boldsymbol{A}\|}$$

relating the *relative residual* to the relative change in the matrix. Thus, a large relative residual implies a large backward error in the matrix, which means that the algorithm used to compute the solution is unstable.

But how large is $\|\boldsymbol{E}\|$ likely to be in practice? Wilkinson [273] showed that for LU factorization by Gaussian elimination, a bound of the form

$$\frac{\|\boldsymbol{E}\|}{\|\boldsymbol{A}\|} \leq \rho\, n\, \epsilon_{\text{mach}}$$

holds, where ρ, called the *growth factor*, is the ratio of the largest entry of $\boldsymbol{U}$ to the largest entry of $\boldsymbol{A}$. Without pivoting, ρ can be arbitrarily large, and hence Gaussian elimination without pivoting is unstable, as we have already seen. With partial pivoting, the growth factor can still be as large as 2^{n-1} (since in the worst case the size of the entries can double at each stage of elimination), but such behavior is extremely rare. In practice, there is little or no growth in the size of the entries, so that

$$\frac{\|\boldsymbol{E}\|}{\|\boldsymbol{A}\|} \approx n\, \epsilon_{\text{mach}}.$$

This relation means that solving a linear system by Gaussian elimination with partial pivoting followed by back-substitution almost always yields a very small relative residual, regardless of how ill-conditioned the system may be. Thus, a small relative residual is not necessarily a good indicator that a computed solution is close to the "true" solution unless the system is well-conditioned.

Complete pivoting yields an even smaller growth factor, in both theory and practice, but the additional margin of stability it provides is usually not worth the extra expense.

Example 2.10 Small Residual. Using three-digit decimal arithmetic to solve the system

$$\begin{bmatrix} 0.641 & 0.242 \\ 0.321 & 0.121 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.883 \\ 0.442 \end{bmatrix},$$

Gaussian elimination with partial pivoting yields the triangular system

$$\begin{bmatrix} 0.641 & 0.242 \\ 0 & -0.000242 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.883 \\ -0.000383 \end{bmatrix},$$

and back-substitution then gives the computed solution

$$\boldsymbol{x} = \begin{bmatrix} 0.782 \\ 1.58 \end{bmatrix}.$$

The exact residual for this solution is

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} -0.000622 \\ -0.000202 \end{bmatrix},$$

which is as small as we can expect using only three-digit arithmetic. Yet the exact solution for this system is easily seen to be

$$\boldsymbol{x} = \begin{bmatrix} 1.00 \\ 1.00 \end{bmatrix},$$

so that the error is almost as large as the solution. The cause of this phenomenon is that the matrix is very nearly singular (its condition number is more than 4000). The division that determines x_2 is between two quantities that are both on the order of rounding error, and hence the result is essentially arbitrary. Yet, by design, when this arbitrary value for x_2 is then substituted into the first equation, a value for x_1 is computed so that the first equation is satisfied. Thus, we get a small residual, but a poor solution.

2.4.2 Estimating Accuracy

In addition to being a reliable indicator of near singularity, the condition number also provides a quantitative estimate for the error in the computed solution to a linear system, as we will now see. Let $\boldsymbol{x}$ be the solution to the nonsingular linear system $\boldsymbol{Ax} = \boldsymbol{b}$, and let $\hat{\boldsymbol{x}}$ be the solution to the system $\boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{b} + \Delta\boldsymbol{b}$ with a perturbed right-hand side. If we define $\Delta\boldsymbol{x} = \hat{\boldsymbol{x}} - \boldsymbol{x}$, then we have

$$\boldsymbol{b} + \Delta\boldsymbol{b} = \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{Ax} + \boldsymbol{A}\Delta\boldsymbol{x}.$$

Since $\boldsymbol{Ax} = \boldsymbol{b}$, we must have $\boldsymbol{A}\Delta\boldsymbol{x} = \Delta\boldsymbol{b}$, and hence $\Delta\boldsymbol{x} = \boldsymbol{A}^{-1}\Delta\boldsymbol{b}$. Now

$$\boldsymbol{b} = \boldsymbol{Ax} \;\Rightarrow\; \|\boldsymbol{b}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{x}\|,$$

and

$$\Delta\boldsymbol{x} = \boldsymbol{A}^{-1}\Delta\boldsymbol{b} \;\Rightarrow\; \|\Delta\boldsymbol{x}\| \leq \|\boldsymbol{A}^{-1}\| \cdot \|\Delta\boldsymbol{b}\|,$$

which, upon using the definition $\text{cond}(\boldsymbol{A}) = \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\|$, yields the estimate

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \text{cond}(\boldsymbol{A})\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}.$$

Thus, the condition number of the matrix determines the possible relative change in the solution due to a given relative change in the right-hand-side vector, regardless of the algorithm used to compute the solution (compare with the general notion of condition number defined in Section 1.2.5). A similar result holds for relative changes in the entries of the matrix $\boldsymbol{A}$. If $\boldsymbol{Ax} = \boldsymbol{b}$ and

$$(\boldsymbol{A} + \boldsymbol{E})\hat{\boldsymbol{x}} = \boldsymbol{b},$$

then

$$\boldsymbol{x} - \hat{\boldsymbol{x}} = \boldsymbol{A}^{-1}(\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}) = \boldsymbol{A}^{-1}\boldsymbol{E}\hat{\boldsymbol{x}},$$

so that

$$\|\Delta\boldsymbol{x}\| \leq \|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{E}\| \cdot \|\hat{\boldsymbol{x}}\|,$$

which yields the estimate

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\hat{\boldsymbol{x}}\|} \leq \text{cond}(\boldsymbol{A})\frac{\|\boldsymbol{E}\|}{\|\boldsymbol{A}\|}.$$

As an alternative to the algebraic derivations just given, calculus can be used to estimate the sensitivity of linear systems. Introducing the real-valued parameter t, we define $\boldsymbol{A}(t) = \boldsymbol{A} + t\boldsymbol{E}$ and $\boldsymbol{b}(t) = \boldsymbol{b} + t\Delta\boldsymbol{b}$, and consider the solution $\boldsymbol{x}(t)$ to the linear system $\boldsymbol{A}(t)\boldsymbol{x}(t) = \boldsymbol{b}(t)$. Differentiating this equation with respect to t, we obtain

$$\boldsymbol{A}'(t)\boldsymbol{x}(t) + \boldsymbol{A}(t)\boldsymbol{x}'(t) = \boldsymbol{b}'(t),$$

so that we have

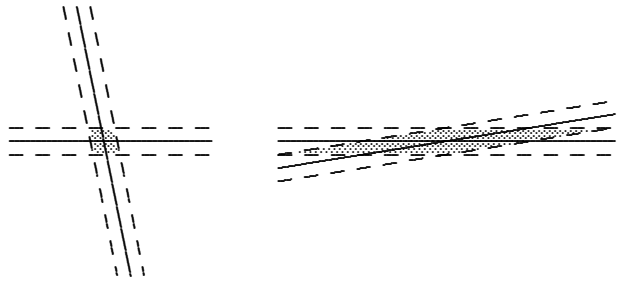
$$\boldsymbol{x}'(t) = -\boldsymbol{A}^{-1}(t)\boldsymbol{A}'(t)\boldsymbol{x}(t) + \boldsymbol{A}^{-1}(t)\boldsymbol{b}'(t),$$

and hence, evaluating at $t = 0$ and taking norms,

$$\frac{\|\boldsymbol{x}'\|}{\|\boldsymbol{x}\|} \leq \|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}'\| + \|\boldsymbol{A}^{-1}\| \cdot \frac{\|\boldsymbol{b}'\|}{\|\boldsymbol{x}\|} \leq \text{cond}(\boldsymbol{A})\left(\frac{\|\boldsymbol{A}'\|}{\|\boldsymbol{A}\|} + \frac{\|\boldsymbol{b}'\|}{\|\boldsymbol{b}\|}\right).$$

Thus, we again see that the relative change in the solution is bounded by the condition number times the relative change in the problem data.

A geometric interpretation in two dimensions of these sensitivity results is that if the two straight lines defined by the two equations are nearly parallel, then their point of intersection is not sharply defined if the lines are a bit fuzzy because of rounding errors or other sources of error. If, on the other hand, the lines are far from parallel, say nearly perpendicular, then their intersection is relatively sharply defined. These two cases are illustrated in Fig. 2.2, where the dashed lines indicate the region of uncertainty for each solid line, so that the intersection point in each case could be anywhere within the shaded parallelogram. Thus, a large condition number is associated with a large uncertainty in the solution.



well-conditioned ill-conditioned

Figure 2.2: Well-conditioned and ill-conditioned linear systems.

To summarize, if the input data are accurate to machine precision, then a reasonable estimate for the relative error in the computed solution to a linear system is given by

$$\frac{\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \approx \text{cond}(\boldsymbol{A})\ \epsilon_{\text{mach}}.$$

One simple way of interpreting these results is that the computed solution loses about $\log_{10}(\text{cond}(\boldsymbol{A}))$ decimal digits of accuracy relative to the accuracy of the input. In Example 2.10, for instance, with a condition number greater than 10^3, we lost all of the three-digit precision available and obtained an arbitrary solution.

Before leaving the subject of assessing accuracy in terms of condition numbers, note these two caveats:

- The foregoing analysis estimates the relative error in the *largest* components of the solution vector. The *relative* error in the smaller components can be much larger, because a vector norm is dominated by the largest components of a vector. Componentwise error bounds can be obtained but are somewhat more complicated to compute, and we will not pursue this topic. Componentwise bounds are of particular interest when the system is poorly scaled.
- The condition number of a matrix is affected by the scaling of the matrix (recall Example 2.3). A large condition number can result simply from poor scaling, as well as from near singularity. Rescaling the matrix can help the former, but not the latter (see Section 2.4.3).