The process of Gaussian elimination was introduced in Chapter 1 for a system of two equations. Example 3.1 illustrates the steps for a system of three equations in three unknowns.

Example 3.1 Three-by-Three System

The three-by-three system

\[
\begin{align*}
  x + 2y + 3z &= 1, \\
  2x + 6y + 10z &= 0, \\
  3x + 14y + 28z &= -8,
\end{align*}
\]

can be solved by Gaussian elimination as described in the following steps:

**Step 1**
Use the first equation to eliminate \(x\) in the second and third equations.
Multiply the first equation by \(-2\) and add it to the second equation to get a new second equation with the \(x\) variable eliminated.
Also, multiply the first equation by \(-3\) and add it to the third equation to get a new third equation with the \(x\) variable eliminated. The resulting system is

\[
\begin{align*}
  x + 2y + 3z &= 1, \\
  2y + 4z &= -2, \\
  8y + 19z &= -11.
\end{align*}
\]

**Step 2**
Use the second equation to eliminate the \(y\) term in the third equation.
Multiply the second equation by \(-4\) and add it to the third equation to get a new third equation with the \(y\) variable eliminated. The system now looks like this:

\[
\begin{align*}
  x + 2y + 3z &= 1, \\
  2y + 4z &= -2, \\
  3z &= -3.
\end{align*}
\]

This completes the "forward elimination" phase; we have an upper triangular system.
We now use "back substitution" to find the values of the unknowns:

\[
\begin{align*}
  3z &= -3 & \Rightarrow z &= -1, \\
  2y + 4(-1) &= -2 & \Rightarrow y &= 1, \\
  x + 2(1) + 3(-1) &= 1 & \Rightarrow x &= 2.
\end{align*}
\]
3.1.1 Using Matrix Notation

We see in the preceding example that all of the computations are based on the coefficients and elements of the right-hand side of the system of equations. This system is written in matrix-vector form as $\mathbf{Ax} = \mathbf{b}$ (see Chapter 1) with

$$
\mathbf{A} = \begin{bmatrix}
1 & 2 & 3 \\
2 & 6 & 10 \\
3 & 14 & 28 \\
\end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix}
1 \\
0 \\
-8 \\
\end{bmatrix}.
$$

Since the same operations are performed on the matrix $\mathbf{A}$ and the vector $\mathbf{b}$, they are often combined in the augmented matrix:

$$
\begin{bmatrix}
1 & 2 & 3 & 1 \\
2 & 6 & 10 & 0 \\
3 & 14 & 28 & -8 \\
\end{bmatrix}.
$$

There are a variety of ways to transform a linear system of equations into an equivalent system having the same solution. However, the basic Gaussian elimination procedure follows a specific sequence of steps and only uses operations of the following form:

Add a multiple $m$ of row $R_i$ onto row $R_j$ to form a new row $R_j$, or

$$
R_j \leftarrow mR_i + R_j.
$$

At the $k$th stage of the basic Gaussian elimination procedure, the appropriate multiples of the $k$th equation are used to eliminate the $k$th variable from equations $k+1, \ldots, n$; in terms of the coefficient matrix $\mathbf{A}$, the appropriate multiple of the $k$th row is used to reduce each of the entries in the $k$th column below the $k$th row to zero. The $k$th row is called the pivot row, the $k$th column is the pivot column, and the element $a_{kk}$ is the pivot element. Since finding the appropriate multiplier for each row requires dividing by the pivot element for that stage, the process fails if the pivot element is zero. Possible remedies for this situation are discussed later in the chapter; they allow interchanging the order of the rows of the augmented matrix.

The Gaussian elimination process is illustrated in the next example using the equations for the electrical circuit in Example 3–A.
3.1.3 Discussion

There are two key aspects to understanding why Gaussian elimination works. The first is to see why a linear combination of two equations passes through the point of intersection of the two equations. The second is to consider why (or when) the

sequence of steps for Gaussian elimination produces a system that can be solved by back substitution. Analyzing these two questions suggests when Gaussian elimination works well, when it works poorly or fails, and when it can be improved.

If two equations have a point in common, then that point is also a solution of any equation formed as a linear combination of the equations. This result can be shown by simple algebra. Consider two linear equations,

\[ S_1: \quad a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0 \]

and

\[ T_1: \quad b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n = 0, \]

and assume that the point \( r = (r_1, r_2, \ldots, r_n) \) satisfies both equations. Then \( r \) also satisfies the linear combination \( C = m_1S_1 + m_2T_1 \), or

\[
\begin{align*}
m_1a_0 + m_1a_1x_1 + m_1a_2x_2 + \ldots + m_1a_nx_n \\
+ m_2b_0 + m_2b_1x_1 + m_2b_2x_2 + \ldots + m_2b_nx_n = 0.
\end{align*}
\]

Substituting \((r_1, r_2, \ldots, r_n)\) in the equation \( C \) and using the fact that \( r \) satisfies \( S_1 \) and \( T_1 \) gives the desired result.

Now consider the specific sequence of transformations on the linear system given in the basic Gaussian elimination algorithm presented earlier. The description assumes that it is possible to find the necessary multiplier to reduce each column to zero as indicated. Two situations can arise if a zero pivot element is encountered, depending on whether or not there are any nonzero elements in the pivot column below the pivot row.

If a zero pivot occurs (in an \( n \)-by-\( n \) linear system), and the entire pivot column below the pivot row is also zero, then the system of equations does not have a unique solution. The equations are either inconsistent or redundant.

On the other hand, if a zero element is encountered in a pivot position, but there is a nonzero element in the pivot column below the pivot element, the Gaussian elimination process can be modified to allow for interchanging the row whose pivot element is zero with a row below it. This process is called (partial) pivoting and is the subject of the next section.

Several considerations are operative in determining how well a particular numerical method works. Among the most important are questions dealing with the quality of the solution, the sensitivity of the method to errors (including inexact arithmetic), and the computational effort required.

Computational effort is usually measured in terms of the number of multiplications and divisions \((m + d)\) or in terms of the number of floating-point operations (flops). On early computers, multiplication and division were much more time intensive than addition and subtraction, which led researchers to analyze algorithms in terms of multiplication and division. It is also typical for the number of additions and subtractions to be directly related to the number of multiplications and divisions. Today, because the difference in effort for different operations has been reduced, analysis in terms of flops has become more common. Since the linear systems that arise in practice are often very large, it is important to see how the computational effort required for Gaussian elimination is related to the size of the coefficient matrix \( A \) (assumed to be \( n \)-by-\( n \)).
At the first stage of Gaussian elimination, one division is required to find the multiplier for the second row \( m_{12} = -a_{12}/a_{11} \). Then, \( n \) multiplications (and \( n \) additions) are required to form the new second row. Note that we must also multiply the right-hand side, but we do not have to multiply the first element in the row, since we know that the new first element in the second row will be zero. This process must be performed for each of the rows below the first row. Thus, the first stage requires \( (n + 1)(n - 1) \) multiplications and divisions, as well as \( n(n - 1) \) additions.

At the \( k \)th stage of elimination, there is one division to form the multiplier \( m_{k} \) and \( (n - k) \) multiplications to generate the new \( k \)th row (for \( i = k + 1, \ldots, n \)). There are also \( n - k \) additions required for each new row.

The total number of multiplications and divisions is

\[
\sum_{k=1}^{n-1} (n - k + 1)(n - k) = \sum_{k=1}^{n-1} n^2 - 2nk + k^2 + n - k
\]

\[
= \sum_{k=1}^{n-1} (n^2 + n) - \sum_{k=1}^{n-1} (2n + 1)k + \sum_{k=1}^{n-1} k^2
\]

which simplifies to

\[
= \frac{n^3}{3} - \frac{n}{3}.
\]

If there is a unique solution, if computations are exact, and if the pivot element is not zero at any stage, Gaussian elimination gives the solution. However, since computer computations are not exact, we may be faced with errors in the calculations that result from round-off. We illustrate here two types of difficulties that can occur. The first can be avoided by a suitable reordering of the rows of the augmented matrix, which is the subject of the next section. The second example is indicative of a more serious problem.

Gaussian elimination works well for systems with coefficient matrices with special properties. For example, if \( A \) is strictly diagonally dominant (i.e., for each \( i, \mid a_{ii} \mid > \sum_{j \neq i} \mid a_{ij} \mid \)), Gaussian elimination will work well [Golub and Van Loan, 1996, p.120].

**Some Difficulties Are Solvable with Pivoting**

Consider the following system of two equations in two unknowns, and suppose that we have arithmetic with rounding to two digits at each stage of the Gaussian elimination process:

\[
0.001x_1 + x_2 = 3,
\]

\[
x_1 + 2x_2 = 5.
\]

Proceeding according to the basic Gaussian elimination procedure, we multiply the first equation by \(-1000\) and add it to the second equation to give (if arithmetic were exact)

\[
-998x_2 = -2995.
\]

After rounding, this equation becomes \(-1000x_2 = -3000\), which gives \( x_2 = 3 \). Substitution of this value into the first equation then yields \( x_1 = 0 \). Clearly, this is not
a good approximation to a solution of the second equation. In the next section, we discuss an enhancement to Gaussian elimination to avoid such a dilemma.

**An Ill-Conditioned Matrix Causes More Serious Difficulties**

Consider the linear system

\[ x_1 + \frac{1}{2} x_2 = \frac{3}{2}, \]

\[ \frac{1}{2} x_1 + \frac{1}{3} x_2 = \frac{5}{6}. \]

Using exact arithmetic gives the exact solution \( x_1 = x_2 = 1 \). However, if the right-hand side is modified slightly, to

\[ x_1 + \frac{1}{2} x_2 = \frac{3}{2}, \]

\[ \frac{1}{2} x_1 + \frac{1}{3} x_2 = 1, \]

the exact solution becomes \( x_1 = 0 \) and \( x_2 = 3 \).

This extreme sensitivity to small changes in the right-hand side is evidence of the fact that the coefficient matrix is “ill conditioned.” The difficulties arising from ill conditioning cannot be solved by simple refinements in the Gaussian elimination procedure. If the condition number of a matrix is defined as the ratio of the largest eigenvalue to the smallest eigenvalue, a matrix with a large condition number is ill-conditioned. Eigenvalues are discussed in Chapter 7. The condition number of a matrix can also be found by using the built-in MATLAB function `cond`. The coefficient matrix in this example is a two-by-two Hilbert matrix, and the MATLAB code is as follows:

```matlab
EDU> H = [ 1 1/2
          1/2 1/3]
EDU> cond(H)
ans = 19.281
```

### 3.2 GAUSSIAN ELIMINATION WITH ROW PIVOTING

Basic Gaussian elimination, as presented in the previous section, fails if the pivot element at any stage of the elimination process is zero, because division by zero is not possible. In addition, difficulties that are not as easy to detect arise if the pivot element is significantly smaller than the coefficients it is being used to eliminate. In this section, we investigate an enhancement to Gaussian elimination that prevents or alleviates some of these shortcomings of the basic procedure.

In order to reduce the inaccuracies that occur in solutions computed with Gaussian elimination and avoid (if possible) the failure of the method resulting from
a zero coefficient in the pivot position at some stage of the process, we may need to interchange selected rows of the augmented matrix. We illustrate this process, known as row pivoting, in Example 3.4.

**Example 3.4 Difficult System**

Consider again the following simple system of two equations in two unknowns, to be solved by Gaussian elimination with rounding to two significant digits at each stage of the process:

\[
\begin{align*}
0.001x_1 + & \quad x_2 = 3, \\
& \quad x_1 + 2x_2 = 5.
\end{align*}
\]

With basic Gaussian elimination (with rounding), we found that \(x_2 = 3\) and \(x_1 = 0\). This is not a good approximation to a solution of the second equation.

If, instead of solving the system with the equations in the order given, we recognize that the very small coefficient of \(x_1\) in the first equation is dangerous (because we would be dividing by something that is close to zero), we can interchange the order of the equations as follows

\[
\begin{align*}
& \quad x_1 + 2x_2 = 5, \\
0.001x_1 + & \quad x_2 = 3.
\end{align*}
\]

Now we multiply the first equation by \(-0.001\) and add it to the second equation to give (if arithmetic were exact)

\[0.998x_2 = 2.995.\]

After rounding, this equation becomes \(x_2 = 3\). Substitution of this value into the first equation yields \(x_1 = -1\), a much better approximation to a solution of the system.

---

The small pivot element shown in the previous example could occur at any stage of elimination. Gaussian elimination with row pivoting checks all entries in the pivot column (from the current diagonal element to the bottom of the column) and chooses the largest element as the pivot. The current row and the selected pivot row are interchanged. The following example illustrates a more limited use of pivoting, with row interchanges performed only if a zero pivot is encountered.

**Example 3.5 A Three-By-Three System in which Pivoting Is Required**

Consider the system of equations

\[
\begin{align*}
2x + & \quad 6y + 10z = 0, \\
x + & \quad 3y + 3z = 2, \\
3x + 14y + 28z = & \quad -8.
\end{align*}
\]
The augmented matrix is
\[
\begin{bmatrix}
2 & 6 & 10 & | & 0 \\ 
1 & 3 & 3 & | & 2 \\ 
3 & 14 & 28 & | & -8 \\
\end{bmatrix}
\]

The first stage of elimination gives
\[
\begin{bmatrix}
2 & 6 & 10 & | & 0 \\ 
0 & 0 & -2 & | & 2 \\ 
0 & 5 & 13 & | & -8 \\
\end{bmatrix}
\]

We are unable to continue, unless we interchange the second and third rows:
\[
\begin{bmatrix}
2 & 6 & 10 & | & 0 \\ 
0 & 5 & 13 & | & -8 \\ 
0 & 0 & -2 & | & 2 \\
\end{bmatrix}
\]

In this simple example, after the row pivoting, no further elimination is required.

By back substitution,
\[x_3 = -1;\]
\[x_2 = \frac{1}{5} [-13(-1) - 8] = 1;\]
\[x_1 = \frac{1}{2} [-6(1) - 10(-1) + 0] = 2.\]
3.3 GAUSSIAN ELIMINATION FOR TRIDIAGONAL SYSTEMS

In many applications, the linear system to be solved has a banded structure. For a tridiagonal system, the only nonzero entries in the coefficient matrix are the diagonal, the subdiagonal, and the superdiagonal. To take advantage of this special structure, we apply a modified form of Gaussian elimination in which, following each elimination step, the pivot row is scaled so that the diagonal element is 1. We begin by illustrating the process for a four-by-four system.

Example 3.7 Solving a Tridiagonal System by Gaussian Elimination with Row Scaling

Consider the following system of equations:

\[
\begin{align*}
2x_1 & - x_2 & = 1, \\
-x_1 + 2x_2 & - x_3 & = 0, \\
-x_2 + 2x_3 & - x_4 &= 0, \\
-x_3 + 2x_4 & = 1.
\end{align*}
\]

First, scale the first equation by dividing through by \(a_{11}\), so that the new first equation has 1 on the diagonal:

\[
\begin{align*}
x_1 & - \frac{1}{2} x_2 & = \frac{1}{2}, \\
-x_1 + 2x_2 & - x_3 & = 0, \\
-x_2 + 2x_3 & - x_4 &= 0, \\
-x_3 + 2x_4 & = 1.
\end{align*}
\]

This modifies two other elements also: the element on the upper diagonal and the one on the right-hand side.

Second, use the first equation to eliminate the \(x_1\) term in the second equation (because of the tridiagonal structure, that is the only equation below the first in which \(x_1\) appears):

\[
\begin{align*}
x_1 & - \frac{1}{2} x_2 & = \frac{1}{2}, \\
+ \frac{3}{2} x_2 & - x_3 & = \frac{1}{2}, \\
-x_2 + 2x_3 & - x_4 &= 0, \\
-x_3 + 2x_4 & = 1.
\end{align*}
\]
Complete this step by scaling the second equation:

\[ x_1 - \frac{1}{2} x_2 = \frac{1}{2}, \]
\[ + x_2 - \frac{2}{3} x_3 = \frac{1}{3}, \]
\[ -x_2 + 2x_3 - x_4 = 0, \]
\[ -x_3 + 2x_4 = 1. \]

Next, use the second equation to eliminate the \( x_2 \) term in the third equation:

\[ x_1 - \frac{1}{2} x_2 = \frac{1}{2}, \]
\[ + x_2 - \frac{2}{3} x_3 = \frac{1}{3}, \]
\[ \frac{4}{3} x_3 - x_4 = \frac{1}{3}, \]
\[ -x_3 + 2x_4 = 1. \]

Now scale the third equation:

\[ x_1 - \frac{1}{2} x_2 = \frac{1}{2}, \]
\[ + x_2 - \frac{2}{3} x_3 = \frac{1}{3}, \]
\[ + x_3 - \frac{3}{4} x_4 = \frac{1}{4}, \]
\[ -x_3 + 2x_4 = 1. \]

Finally, use the third equation to eliminate \( x_3 \) in the last equation:

\[ x_1 - \frac{1}{2} x_2 = \frac{1}{2}, \]
\[ + x_2 - \frac{2}{3} x_3 = \frac{1}{3}, \]
\[ + x_3 - \frac{3}{4} x_4 = \frac{1}{4}, \]
\[ + \frac{5}{4} x_4 = \frac{5}{4}. \]

And scale the last equation:
\begin{align*}
x_1 - \frac{1}{2} x_2 &= \frac{1}{2}, \\
+ x_2 - \frac{2}{3} x_3 &= \frac{1}{3}, \\
+ x_3 - \frac{3}{4} x_4 &= \frac{1}{4}, \\
+ x_4 &= 1.
\end{align*}

Now solve by back substitution to obtain
\begin{align*}
x_4 &= 1; & x_3 &= 1/4 - (-3/4)(1) = 1; \\
x_2 &= 1/3 - (-2/3)(1) = 1; & x_1 &= 1/2 - (-1/2)(1) = 1.
\end{align*}

Because of the special form of the matrix $A$, we can reduce the storage requirements from $n^2$ to $3n$ by storing only the vector $d$ containing the diagonal elements, the vector $a$ containing the elements above the diagonal, and the vector $b$ containing the elements below the diagonal. Note that elements $b_1$ and $a_n$ are zero. The right-hand side is stored as the vector $r$. In this notation, the general tridiagonal system of equations can be written as
\begin{align*}
d_1 x_1 + a_1 x_2 &= r_1, \\
b_2 x_1 + d_2 x_2 + a_2 x_3 &= r_2, \\
&\quad \ldots \ldots \\
+ b_{n-1} x_{n-2} + d_{n-1} x_{n-1} + a_{n-1} x_n &= r_{n-1}, \\
+ b_n x_{n-1} + d_n x_n &= r_n.
\end{align*}

An efficient algorithm for the solution of a tridiagonal system is based on Gaussian elimination with the coefficients of the diagonal elements scaled to 1 at each stage. This algorithm takes advantage of the zero elements that are already present in the coefficient matrix and avoids unnecessary arithmetic operations. Thus, we need to store only the new vectors $a$ and $r$. This procedure is known in the engineering literature as the Thomas method.

**Step 1**

For the first equation, form the new elements $a_1$ and $r_1$:
\begin{align*}
a_1 &= \frac{a_1}{d_1}, & r_1 &= \frac{r_1}{d_1}.
\end{align*}

**Step 2**

For each of the equations, from $i = 2, \ldots, n - 1$,
\begin{align*}
a_i &= \frac{a_i}{d_i - b_i a_{i-1}}, & r_i &= \frac{r_i - b_i r_{i-1}}{d_i - b_i a_{i-1}}.
\end{align*}
Step 3
For the last equation:
\[ r_n = \frac{r_n - b_nr_{n-1}}{d_n - b_na_{n-1}}. \]

Step 4
Solve by back substitution:
\[ x_n = r_n, \]
\[ x_i = r_i - a_i x_{i+1}, \quad i = n-1, n-2, n-3, \ldots, 2, 1. \]

Example 3.8 Solving a Tridiagonal System Using the Thomas Method

We use the tridiagonal system from Example 3.7 to illustrate the steps of the Thomas method:
\[ 2x_1 - x_2 = 1, \]
\[-x_1 + 2x_2 - x_3 = 0, \]
\[-x_2 + 2x_3 - x_4 = 0, \]
\[-x_3 + 2x_4 = 1. \]
\[ d = (2, 2, 2, 2); a = (-1, -1, -1, 0); b = (0, -1, -1, -1); r = (1, 0, 0, 1). \]

First, form the new elements \( a_1 \) and \( r_1 \):
\[ a_1 = \frac{a_1}{d_1} = \frac{-1}{2}, \quad r_1 = \frac{r_1}{d_1} = \frac{1}{2}. \]

For the second equation,
\[ a_2 = \frac{a_2}{d_2 - b_2a_1} = \frac{-1}{2 - (-1)(-1/2)} = \frac{2}{3}, \]
\[ r_2 = \frac{r_2 - b_2r_1}{d_2 - b_2a_1} = \frac{0 - (-1)(1/2)}{2 - (-1)(-1/2)} = \frac{1}{3}. \]

For the third equation,
\[ a_3 = \frac{a_3}{d_3 - b_3a_2} = \frac{-1}{2 - (-1)(-2/3)} = \frac{3}{4}, \]
\[ r_3 = \frac{r_3 - b_3r_2}{d_3 - b_3a_2} = \frac{0 - (-1)(1/3)}{2 - (-1)(-2/3)} = \frac{1}{4}. \]

For the last equation,
\[ r_4 = \frac{r_4 - b_4r_3}{d_4 - b_4a_3} = \frac{1 - (-1)(1/4)}{2 - (-1)(-3/4)} = 1. \]

Finally, solve by back substitution to obtain
\[ x_4 = r_4 = 1, \]
\[ x_3 = r_3 - a_3x_4 = 1/4 - (-3/4)(1) = 1, \]
\[ x_2 = r_2 - a_2x_3 = 1/3 - (-2/3)(1) = 1, \]
\[ x_1 = r_1 - a_1x_2 = 1/2 - (-1/2)(1) = 1. \]

The Thomas algorithm requires that \( d_i \neq 0 \) and that \( d_i - b_ia_{i-1} \neq 0 \) for each \( i \). For many applications, the structure of the tridiagonal matrix guarantees that these quantities will not be zero. In other cases, if we do encounter a zero value (but the system is, in fact, nonsingular) we can solve for the appropriate variable directly, reduce the size of the system, and solve the new reduced system, as illustrated in the next example.

In general, the Thomas method works well when the system is diagonally dominant.

Example 3.9 Using the Thomas Method for a System That Would Require Pivoting for Gaussian Elimination

To illustrate the possibility of continuing the solution process with the Thomas method when a division by zero is encountered, consider the following system:

\[
\begin{align*}
2x_1 - x_2 &= 1, \\
-x_1 + 2x_2 - x_3 &= 0, \\
-x_2 + \frac{2}{3}x_3 - x_4 &= -\frac{4}{3}, \\
-x_3 + 2x_4 - x_5 &= 0, \\
-x_4 + 2x_5 - x_6 &= 0, \\
-x_5 + 2x_6 &= 1.
\end{align*}
\]

The solution begins in the normal manner by scaling the first equation and using the result to eliminate the \( x_1 \) term in the second equation. Continuing by scaling the second equation and using the result to eliminate the \( x_2 \) term in the third equation, we obtain

\[
\begin{align*}
x_1 - \frac{1}{2}x_2 &= \frac{1}{2}, \\
\frac{1}{2}x_2 - \frac{2}{3}x_3 &= \frac{1}{3}, \\
x_3 &= -1, \\
x_5 &= 0, \\
x_6 &= 0, \\
x_6 &= 1.
\end{align*}
\]

However, we are unable to scale the third equation so as to have 1 on the diagonal, since the coefficient of \( x_3 \) is now 0. But because the third row does have a nonzero coefficient (for variable \( x_4 \)), we can solve for that variable and proceed. Thus, the third equation is solved for \( x_4 \), giving \( x_4 = 1 \). The fourth equation is skipped for now, and the computed value of \( x_4 \) is substituted into the fifth equation. The elimination proceeds, using the fifth equation to eliminate \( x_5 \) from the final equation:
\[
\begin{align*}
  x_1 - \frac{1}{2} x_2 &= \frac{1}{2}, \\
  + x_2 - \frac{2}{3} x_3 &= \frac{1}{3}, \\
  -x_4 &= -1 \text{ (solve)}, \\
  -x_3 + 2x_4 - x_5 &= 0 \text{ (skip for now)}, \\
  + 2x_5 - x_6 &= 1 \text{ (using } x_4 = 1), \\
  -x_5 + 2x_6 &= 1.
\end{align*}
\]

Finally, we scale the fifth equation, use it to eliminate \( x_5 \) in the last equation, and scale the last equation:

\[
\begin{align*}
  x_1 - \frac{1}{2} x_2 &= \frac{1}{2}, \\
  + x_2 - \frac{2}{3} x_3 &= \frac{1}{3}, \\
  -x_4 &= -1, \\
  -x_3 + 2x_4 - x_5 &= 0, \\
  x_5 - \frac{1}{2} x_6 &= \frac{1}{2}, \\
  x_6 &= 1.
\end{align*}
\]

Solving by back substitution yields

\[
\begin{align*}
  x_6 &= 1, \\
  x_5 &= \frac{1}{2} + \frac{1}{2} x_6 = 1, \\
  x_4 &= 1 \text{ (computed previously),} \\
  x_3 &= 2x_4 - x_5 = 1 \text{ (skipped previously),} \\
  x_2 &= \frac{1}{3} + \frac{2}{3} x_3 = 1, \\
  x_1 &= \frac{1}{2} + \frac{1}{2} x_2 = 1.
\end{align*}
\]

3.4 **MATLAB’S METHODS**

We conclude the chapter with a brief discussion of solving linear systems by means of MATLAB’s built-in capabilities. The methods implemented in high-quality professionally developed software, such as MATLAB, are efficient and easy to use. They
often combine the best features from several of the basic techniques presented in this chapter. In other cases, the methods use more sophisticated procedures that are beyond the scope of the text.

There are two division symbols in MATLAB, the forward slash (/) and the backslash (\). It is useful to think of each as indicating multiplication by the inverse of the quantity under the slash. In this way of interpreting the symbols, $a/b = a(b^{-1})$ and $c\backslash d = c^{-1}(d)$. Since scalar multiplication is commutative, we can write the fraction $\frac{1}{2}$ or $\frac{2}{1}$. However, matrix multiplication is not commutative, and “backslash” division gives us a convenient way of solving the linear system $Ax = b$. The solution of $Ax = b$ is $x = A\backslash b$, which is suggestive of $x = A^{-1}b$, where it is important that the implied multiplication by the inverse of $A$ is from the left. The backslash division operator is also called the matrix left-division operator.

If $A$ is an $n$-by-$n$ matrix and $b$ is a column vector with $n$ components (or a matrix with several such columns), then $x = A\backslash b$ is the solution of the equation $Ax = b$, computed by Gaussian elimination (not computed by multiplying by the inverse of $A$). A warning message is printed if $A$ is badly scaled or nearly singular. The left-division operator can also be used when $A$ is $n$-by-$m$ ($n \neq m$) to obtain a solution of the over- or under-determined system. A system is underdetermined if there are more unknowns than equations and overdetermined if there are fewer unknowns than equations. In each case, it is the number of linearly independent equations that is important. The concept of linear independence is discussed in standard texts on linear algebra.

Information about MATLAB’s operations can be found using the command

```
help matlab:ops
```

The matrix left-division operator is implemented in the function `mldivide`, which provides a brief description of its approach in the comments at the beginning of the function.

It is also possible to solve the linear system $Ax = b$ by using MATLAB’s function for finding a matrix inverse (discussed in Chapter 5); however, multiplication by a matrix inverse is usually not the best way to solve a linear system.