Chapter 4

Solving a System of Linear Equations

Core Topics

Gauss elimination method (4.2).

Gauss elimination with pivoting (*4.3*).

Gauss–Jordan elimination method (4.4).

LU decomposition method (4.5).

Inverse of a matrix (4.6)

Iterative methods (Jacobi, Gauss–Seidel) (4.7).

Use of MATLAB's built-in functions for solving a system of linear equations (4.8).

Complementary Topics

Tridiagonal systems of equations (4.9).

Error, residual, norms, and condition number (4.10).

Ill-conditioned systems (4.11).

4.1 BACKGROUND

Systems of linear equations that have to be solved simultaneously arise in problems that include several (possibly many) variables that are dependent on each other. Such problems occur not only in engineering and science, which are the focus of this book, but in virtually any discipline (business, statistics, economics, etc.). A system of two (or three) equations with two (or three) unknowns can be solved manually by substitution or other mathematical methods (e.g., Cramer's rule, Section 2.4.6). Solving a system in this way is practically impossible as the number of equations (and unknowns) increases beyond three.

An example of a problem in electrical engineering that requires a solution of a system of equations is shown in Fig. 4-1. Using Kirchhoff's law, the currents i_1, i_2, i_3, and i_4 can be determined by solving the following system of four equations:

$$\begin{aligned} 9i_1 - 4i_2 - 2i_3 &= 24 \\ -4i_1 + 17i_2 - 6i_3 - 3i_4 &= -16 \\ -2i_1 - 6i_2 + 14i_3 - 6i_4 &= 0 \\ -3i_2 - 6i_3 + 11i_4 &= 18 \end{aligned} \tag{4.1}$$

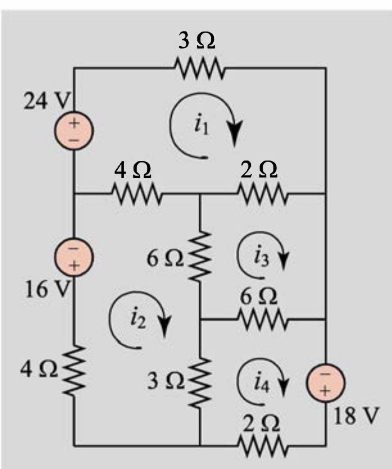


Figure 4-1: Electrical circuit.

Obviously, more complicated circuits may require the solution of a system with a larger number of equations. Another example that requires a solution of a system of equations is calculating the force in members of a truss. The forces in the eight members of the truss shown in Fig. 4-2 are determined from the solution of the following system of eight equations (equilibrium equations of pins A, B, C, and D):

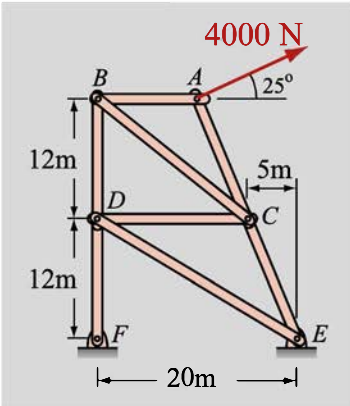


Figure 4-2: Eight-member truss.

$$\begin{aligned} 0.9231F_{AC} &= 1690 & -F_{AB} - 0.3846F_{AC} &= 3625 \\ F_{AB} - 0.7809F_{BC} &= 0 & 0.6247F_{BC} - F_{BD} &= 0 \\ F_{CD} + 0.8575F_{DE} &= 0 & F_{BD} - 0.5145F_{DE} - F_{DF} &= 0 \end{aligned} \tag{4.2}$$
$$\begin{aligned} 0.3846F_{CE} - 0.3846F_{AC} - 0.7809F_{BC} - F_{CD} &= 0 \\ 0.9231F_{AC} + 0.6247F_{BC} - 0.9231F_{CE} &= 0 \end{aligned}$$

There are applications, for example, in finite element and finite difference analysis, where the system of equations that has to be solved contains thousands (or even millions) of simultaneous equations.

4.1.1 Overview of Numerical Methods for Solving a System of Linear Algebraic Equations

The general form of a system of n linear algebraic equations is:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \vdots \qquad \vdots \qquad & \vdots \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned} \tag{4.3}$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix}$$

Figure 4-3: A system of n linear algebraic equations.

The matrix form of the equations is shown in Fig. 4-3. Two types of numerical methods, ***direct*** and ***iterative***, are used for solving systems of linear algebraic equations. In direct methods, the solution is calculated by performing arithmetic operations with the equations. In iterative methods, an initial approximate solution is assumed and then used in an iterative process for obtaining successively more accurate solutions.

Direct methods

In direct methods, the system of equations that is initially given in the general form, Eqs. (4.3), is manipulated to an equivalent system of equations that can be easily solved. Three systems of equations that can be easily solved are the ***upper triangular***, ***lower triangular***, and ***diagonal*** forms.

The ***upper triangular*** form is shown in Eqs. (4.4), and is written in a matrix form for a system of four equations in Fig. 4-4.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n &= b_2 \\ a_{33}x_3 + \ldots + a_{3n}x_n &= b_3 \\ \vdots \qquad \vdots \quad & \vdots \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n &= b_{n-1} \\ a_{nn}x_n &= b_n \end{aligned} \tag{4.4}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 4-4: A system of four equations in upper triangular form.

The system in this form has all zero coefficients below the diagonal and is solved by a procedure called ***back substitution***. It starts with the last equation, which is solved for x_n. The value of x_n is then substituted in the next-to-the-last equation, which is solved for x_{n-1}. The process continues in the same manner all the way up to the first equation. In the case of four equations, the solution is given by:

$$x_4 = \frac{b_4}{a_{44}}, \quad x_3 = \frac{b_3 - a_{34}x_4}{a_{33}}, \quad x_2 = \frac{b_2 - (a_{23}x_3 + a_{24}x_4)}{a_{22}}, \quad \text{and}$$

$$x_1 = \frac{b_1 - (a_{12}x_2 + a_{13}x_3 + a_{14}x_4)}{a_{11}}$$

For a system of n equations in upper triangular form, a general formula for the solution using back substitution is:

$$\begin{aligned} x_n &= \frac{b_n}{a_{nn}} \\ x_i &= \frac{b_i - \sum_{j=i+1}^{j=n} a_{ij}x_j}{a_{ii}} \qquad i = n-1, n-2, \ldots, 1 \end{aligned} \tag{4.5}$$

In Section 4.2 the upper triangular form and back substitution are used in the Gauss elimination method.

The ***lower triangular*** form is shown in Eqs. (4.6), and is written in matrix form for a system of four equations in Fig. 4-5.

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 4-5: A system of four equations in lower triangular form.

$$\begin{aligned} a_{11}x_1 \qquad\qquad\qquad\qquad\qquad &= b_1 \\ a_{21}x_1 + a_{22}x_2 \qquad\qquad\qquad\qquad &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \qquad\qquad\quad &= b_3 \\ \vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad &\;\;\vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n &= b_n \end{aligned} \tag{4.6}$$

The system in this form has zero coefficients above the diagonal. A system in lower triangular form is solved in the same way as the upper triangular form but in an opposite order. The procedure is called ***forward substitution***. It starts with the first equation, which is solved for x_1. The value of x_1 is then substituted in the second equation, which is solved for x_2. The process continues in the same manner all the way down to the last equation. In the case of four equations, the solution is given by:

$$x_1 = \frac{b_1}{a_{11}}, \quad x_2 = \frac{b_2 - a_{21}x_1}{a_{22}}, \quad x_3 = \frac{b_3 - (a_{31}x_1 + a_{32}x_2)}{a_{33}}, \quad \text{and}$$

$$x_4 = \frac{b_4 - (a_{41}x_1 + a_{42}x_2 + a_{43}x_3)}{a_{44}} \tag{4.7}$$

For a system of n equations in lower triangular form, a general formula for the solution using forward substitution is:

$$x_1 = \frac{b_1}{a_{11}}$$
$$x_i = \frac{b_i - \sum_{j=1}^{j=i-1} a_{ij}x_j}{a_{ii}} \qquad i = 2, 3, \ldots, n \tag{4.8}$$

In Section 4.5 the lower triangular form is used together with the upper triangular form in the *LU* decomposition method for solving a system of equations.

The ***diagonal*** form of a system of linear equations is shown in Eqs. (4.9), and is written in matrix form for a system of four equations in Fig. 4-6.

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_2 \\ b_4 \end{bmatrix}$$

Figure 4-6: A system of four equations in diagonal form.

$$\begin{aligned} a_{11}x_1 \qquad\qquad\qquad\qquad &= b_1 \\ a_{12}x_2 \qquad\qquad\qquad &= b_2 \\ a_{13}x_3 \qquad\qquad &= b_3 \\ \vdots \qquad\qquad &\quad \vdots \\ a_{n}x_n &= b_n \end{aligned} \tag{4.9}$$

A system in diagonal form has nonzero coefficients along the diagonal and zeros everywhere else. Obviously, a system in this form can be easily solved. A similar form is used in the Gauss–Jordan method, which is presented in Section 4.4.

From the three forms of simultaneous linear equations (upper triangular, lower triangular, diagonal) it might appear that changing a given system of equations to the diagonal form is the best choice because the diagonal system is the easiest to solve. In reality, however, the total number of operations required for solving a system is smaller when other methods are used.

Three direct methods for solving systems of equations—Gauss elimination (Sections 4.2 and 4.3), Gauss–Jordan (Section 4.4), and *LU* decomposition (Section 4.5)—and two indirect (iterative) methods—Jacobi and Gauss–Seidel (Section 4.7)—are described in this chapter.

4.2 GAUSS ELIMINATION METHOD

The Gauss elimination method is a procedure for solving a system of linear equations. In this procedure, a system of equations that is given in a general form is manipulated to be in ***upper triangular*** form, which is then solved by using back substitution (see Section 4.1.1). For a set of four equations with four unknowns the general form is given by:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 && (4.10a) \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 && (4.10b) \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3 && (4.10c) \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= b_4 && (4.10d) \end{aligned} \tag{4.10}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 4-7: Matrix form of a system of four equations.

The matrix form of the system is shown in Fig. 4-7. In the Gauss elimination method, the system of equations is manipulated into an equivalent system of equations that has the form:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 & (4.11a) \\ a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 &= b'_2 & (4.11b) \\ a'_{33}x_3 + a'_{34}x_4 &= b'_3 & (4.11c) \\ a'_{44}x_4 &= b'_4 & (4.11d) \end{aligned} \qquad (4.11)$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & 0 & 0 & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

Figure 4-8: Matrix form of the equivalent system.

The first equation in the equivalent system, (4.11*a*), is the same as (4.10*a*). In the second equation, (4.11*b*), the variable x_1 is eliminated. In the third equation, (4.11*c*), the variables x_1 and x_2 are eliminated. In the fourth equation, (4.11*d*), the variables x_1, x_2, and x_3 are eliminated. The matrix form of the equivalent system is shown in Fig. 4-8. The system of equations (4.11) is in upper triangular form, which can be easily solved by using back substitution.

In general, various mathematical manipulations can be used for converting a system of equations from the general form displayed in Eqs. (4.10) to the ***upper triangular*** form in Eqs. (4.11). One in particular, the Gauss elimination method, is described next. The procedure can be easily programmed in a computer code.

Gauss elimination procedure (forward elimination)

The Gauss elimination procedure is first illustrated for a system of four equations with four unknowns. The starting point is the set of equations that is given by Eqs. (4.10). Converting the system of equations to the form given in Eqs. (4.11) is done in steps.

Step 1*:** In the first step, the first equation is unchanged, and the terms that include the variable x_1 in all the other equations are eliminated. This is done one equation at a time by using the first equation, which is called the ***pivot equation. The coefficient a_{11} is called the ***pivot coefficient***, or the pivot element. To eliminate the term $a_{21}x_1$ in Eq. (4.10*b*), the pivot equation, Eq. (4.10*a*), is multiplied by $m_{21} = a_{21}/a_{11}$, and then the equation is subtracted from Eq. (4.10*b*):

$$\begin{array}{rl} & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2 \\ - & m_{21}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4) = m_{21}b_1 \\ \hline & 0 + \underbrace{(a_{22} - m_{21}a_{12})}_{a'_{22}}x_2 + \underbrace{(a_{23} - m_{21}a_{13})}_{a'_{23}}x_3 + \underbrace{(a_{24} - m_{21}a_{14})}_{a'_{24}}x_4 = \underbrace{b_2 - m_{21}b_1}_{b'_2} \end{array}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 4-9: Matrix form of the system after eliminating a_{21}.

It should be emphasized here that the pivot equation, Eq. (4.10*a*), itself is not changed. The matrix form of the equations after this operation is shown in Fig. 4-9.

Next, the term $a_{31}x_1$ in Eq. (4.10*c*) is eliminated. The pivot equation, Eq. (4.10*a*), is multiplied by $m_{31} = a_{31}/a_{11}$ and then is subtracted

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b_4 \end{bmatrix}$$

Figure 4-10: Matrix form of the system after eliminating a_{31}.

from Eq. (4.10*c*):

$$\begin{array}{rl} & a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3 \\ - & m_{31}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4) = m_{31}b_1 \\ \hline & 0 + \underbrace{(a_{32} - m_{31}a_{12})}_{a'_{32}}x_2 + \underbrace{(a_{33} - m_{31}a_{13})}_{a'_{33}}x_3 + \underbrace{(a_{34} - m_{31}a_{14})}_{a'_{34}}x_4 = \underbrace{b_3 - m_{31}b_1}_{b'_3} \end{array}$$

The matrix form of the equations after this operation is shown in Fig. 4-10.

Next, the term $a_{41}x_1$ in Eq. (4.10*d*) is eliminated. The pivot equation, Eq. (4.10*a*), is multiplied by $m_{41} = a_{41}/a_{11}$ and then is subtracted from Eq. (4.10*d*):

$$\begin{array}{rl} & a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4 \\ - & m_{41}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4) = m_{41}b_1 \\ \hline & 0 + \underbrace{(a_{42} - m_{41}a_{12})}_{a'_{42}}x_2 + \underbrace{(a_{43} - m_{41}a_{13})}_{a'_{43}}x_3 + \underbrace{(a_{44} - m_{41}a_{14})}_{a'_{44}}x_4 = \underbrace{b_4 - m_{41}b_1}_{b'_4} \end{array}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

Figure 4-11: Matrix form of the system after eliminating a_{41}.

This is the end of ***Step 1***. The system of equations now has the following form:

$$\begin{array}{rll} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1 & (4.12a) & \\ 0 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 = b'_2 & (4.12b) & \\ 0 + a'_{32}x_2 + a'_{33}x_3 + a'_{34}x_4 = b'_3 & (4.12c) & (4.12) \\ 0 + a'_{42}x_2 + a'_{43}x_3 + a'_{44}x_4 = b'_4 & (4.12d) & \end{array}$$

The matrix form of the equations after this operation is shown in Fig. 4-11. Note that the result of the elimination operation is to reduce the first column entries, except a_{11} (the pivot element), to zero.

***Step 2*:** In this step, Eqs. (4.12*a*) and (4.12*b*) are not changed, and the terms that include the variable x_2 in Eqs. (4.12*c*) and (4.12*d*) are eliminated. In this step, Eq. (4.12*b*) is the pivot equation, and the coefficient a'_{22} is the pivot coefficient. To eliminate the term $a'_{32}x_2$ in Eq. (4.12*c*), the pivot equation, Eq. (4.12*b*), is multiplied by $m_{32} = a'_{32}/a'_{22}$ and then is subtracted from Eq. (4.12*c*):

$$\begin{array}{rl} & a'_{32}x_2 + a'_{33}x_3 + a'_{34}x_4 = b'_3 \\ - & m_{32}(a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4) = m_{32}b'_2 \\ \hline & 0 + \underbrace{(a'_{33} - m_{32}a'_{23})}_{a''_{33}}x_3 + \underbrace{(a'_{34} - m_{32}a'_{24})}_{a''_{34}}x_4 = \underbrace{b'_3 - m_{32}b'_2}_{b''_3} \end{array}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \\ b'_4 \end{bmatrix}$$

Figure 4-12: Matrix form of the system after eliminating a_{32}.

The matrix form of the equations after this operation is shown in Fig. 4-12.

Next, the term $a'_{42}x_2$ in Eq. (4.12*d*) is eliminated. The pivot equation, Eq. (4.12*b*), is multiplied by $m_{42} = a'_{42}/a'_{22}$ and then is subtracted from Eq. (4.12*d*):

$$
\begin{array}{rl}
 & a'_{42}x_2 + a'_{43}x_3 + a'_{44}x_4 = b'_4 \\
- & m_{42}(a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4) = m_{42}b'_2 \\
\hline
 & 0 + \underbrace{(a'_{43} - m_{42}a'_{23})}_{a''_{43}}x_3 + \underbrace{(a'_{44} - m_{42}a'_{24})}_{a''_{44}}x_4 = \underbrace{b'_4 - m_{42}b'_2}_{b''_4}
\end{array}
$$

This is the end of ***Step 2***. The system of equations now has the following form:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 && (4.13a)\\
0 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 &= b'_2 && (4.13b)\\
0 + \quad 0 \quad + a''_{33}x_3 + a''_{34}x_4 &= b''_3 && (4.13c)\\
0 + \quad 0 \quad + a''_{43}x_3 + a''_{44}x_4 &= b''_4 && (4.13d)
\end{aligned}
\qquad (4.13)
$$

$$
\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & a''_{43} & a''_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \\ b''_4 \end{bmatrix}
$$

Figure 4-13: Matrix form of the system after eliminating a_{42}.

The matrix form of the equations at the end of ***Step 2*** is shown in Fig. 4-13.

Step 3: In this step, Eqs. (4.13*a*), (4.13*b*), and (4.13*c*) are not changed, and the term that includes the variable x_3 in Eq. (4.13*d*) is eliminated. In this step, Eq. (4.13*c*) is the pivot equation, and the coefficient a''_{33} is the pivot coefficient. To eliminate the term $a''_{43}x_3$ in Eq. (4.13*d*), the pivot equation is multiplied by $m_{43} = a''_{43}/a''_{33}$ and then is subtracted from Eq. (4.13*d*):

$$
\begin{array}{rl}
 & a''_{43}x_3 + a''_{44}x_4 = b''_4 \\
- & m_{43}(a''_{33}x_3 + a''_{34}x_4) = m_{43}b''_3 \\
\hline
 & \underbrace{(a''_{44} - m_{43}a''_{34})}_{a'''_{44}}x_4 = \underbrace{b''_4 - m_{43}b''_3}_{b'''_4}
\end{array}
$$

This is the end of ***Step 3***. The system of equations is now in an upper triangular form:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 && (4.14a)\\
0 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 &= b'_2 && (4.14b)\\
0 + \quad 0 \quad + a''_{33}x_3 + a''_{34}x_4 &= b''_3 && (4.14c)\\
0 + \quad 0 \quad + \quad 0 \quad + a'''_{44}x_4 &= b'''_4 && (4.14d)
\end{aligned}
\qquad (4.14)
$$

$$
\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & 0 & a'''_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \\ b'''_4 \end{bmatrix}
$$

Figure 4-14: Matrix form of the system after eliminating a_{43}.

The matrix form of the equations is shown in Fig. 4-14. Once transformed to upper triangular form, the equations can be easily solved by using back substitution. The three steps of the Gauss elimination process are illustrated together in Fig. 4-15.

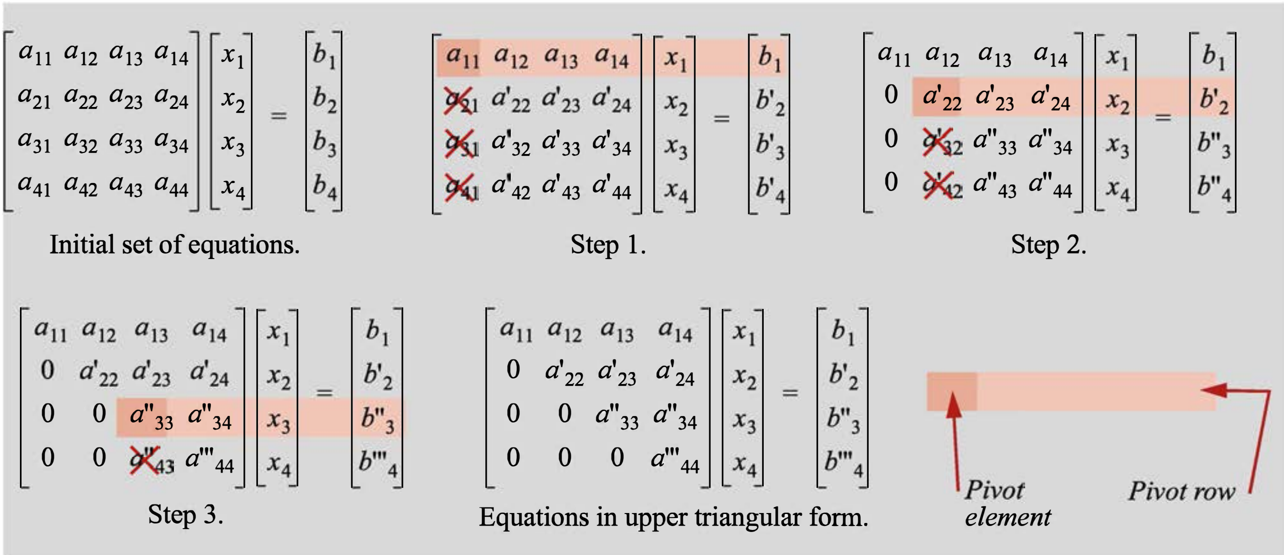


Figure 4-15: Gauss elimination procedure.

Example 4-1 shows a manual application of the Gauss elimination method for solving a system of four equations.

Example 4-1: Solving a set of four equations using Gauss elimination.

Solve the following system of four equations using the Gauss elimination method.

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ -6x_1 + 7x_2 + 6.5x_3 - 6x_4 &= -6.5 \\ x_1 + 7.5x_2 + 6.25x_3 + 5.5x_4 &= 16 \\ -12x_1 + 22x_2 + 15.5x_3 - x_4 &= 17 \end{aligned}$$

SOLUTION

The solution follows the steps presented in the previous pages.

***Step 1*:** The first equation is the pivot equation, and 4 is the pivot coefficient.

Multiply the pivot equation by $m_{21} = (-6)/4 = -1.5$ and subtract it from the second equation:

$$\begin{array}{rrcl} & -6x_1 + 7x_2 + 6.5x_3 - 6x_4 &=& -6.5 \\ - & (-1.5)(4x_1 - 2x_2 - 3x_3 + 6x_4) &=& (-6/4)\cdot 12 \\ \hline & 0x_1 + 4x_2 + 2x_3 + 3x_4 &=& 11.5 \end{array}$$

Multiply the pivot equation by $m_{31} = (1/4) = 0.25$ and subtract it from the third equation:

$$\begin{array}{rrcl} & x_1 + 7.5x_2 + 6.25x_3 + 5.5x_4 &=& 16 \\ - & (0.25)(4x_1 - 2x_2 - 3x_3 + 6x_4) &=& (1/4)\cdot 12 \\ \hline & 0x_1 + 8x_2 + 7x_3 + 4x_4 &=& 13 \end{array}$$

Multiply the pivot equation by $m_{41} = (-12)/4 = -3$ and subtract it from the fourth equation:

$$\begin{array}{rrcl} & -12x_1 + 22x_2 + 15.5x_3 - x_4 &=& 17 \\ - & (-3)(4x_1 - 2x_2 - 3x_3 + 6x_4) &=& -3\cdot 12 \\ \hline & 0x_1 + 16x_2 + 6.5x_3 + 17x_4 &=& 53 \end{array}$$

At the end of ***Step 1***, the four equations have the form:

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ 4x_2 + 2x_3 + 3x_4 &= 11.5 \\ 8x_2 + 7x_3 + 4x_4 &= 13 \\ 16x_2 + 6.5x_3 + 17x_4 &= 53 \end{aligned}$$

Step 2: The second equation is the pivot equation, and 4 is the pivot coefficient.
Multiply the pivot equation by $m_{32} = 8/4 = 2$ and subtract it from the third equation:

$$\begin{array}{rrcl} & 8x_2 + 7x_3 + 4x_4 &=& 13 \\ - & 2(4x_2 + 2x_3 + 3x_4) &=& 2 \cdot 11.5 \\ \hline & 0x_2 + 3x_3 - 2x_4 &=& -10 \end{array}$$

Multiply the pivot equation by $m_{42} = 16/4 = 4$ and subtract it from the fourth equation:

$$\begin{array}{rrcl} & 16x_2 + 6.5x_3 + 17x_4 &=& 53 \\ - & 4(4x_2 + 2x_3 + 3x_4) &=& 4 \cdot 11.5 \\ \hline & 0x_2 - 1.5x_3 + 5x_4 &=& 7 \end{array}$$

At the end of ***Step 2***, the four equations have the form:

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ 4x_2 + 2x_3 + 3x_4 &= 11.5 \\ 3x_3 - 2x_4 &= -10 \\ -1.5x_3 + 5x_4 &= 7 \end{aligned}$$

Step 3: The third equation is the pivot equation, and 3 is the pivot coefficient.
Multiply the pivot equation by $m_{43} = (-1.5)/3 = -0.5$ and subtract it from the fourth equation:

$$\begin{array}{rrcl} & -1.5x_3 + 5x_4 &=& 7 \\ - & -0.5(3x_3 - 2x_4) &=& -0.5 \cdot -10 \\ \hline & 0x_3 + 4x_4 &=& 2 \end{array}$$

At the end of ***Step 3***, the four equations have the form:

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ 4x_2 + 2x_3 + 3x_4 &= 11.5 \\ 3x_3 - 2x_4 &= -10 \\ 4x_4 &= 2 \end{aligned}$$

Once the equations are in this form, the solution can be determined by back substitution. The value of x_4 is determined by solving the fourth equation:

$$x_4 = 2/4 = 0.5$$

Next, x_4 is substituted in the third equation, which is solved for x_3:

$$x_3 = \frac{-10 + 2x_4}{3} = \frac{-10 + 2 \cdot 0.5}{3} = -3$$

Next, x_4 and x_3 are substituted in the second equation, which is solved for x_2:

$$x_2 = \frac{11.5 - 2x_3 - 3x_4}{4} = \frac{11.5 - (2 \cdot -3) - (3 \cdot 0.5)}{4} = 4$$

Lastly, x_4, x_3 and x_2 are substituted in the first equation, which is solved for x_1:

$$x_1 = \frac{12 + 2x_2 + 3x_3 - 6x_4}{4} = \frac{12 + 2 \cdot 4 + 3 \cdot -3 - (6 \cdot 0.5)}{4} = 2$$

The extension of the Gauss elimination procedure to a system with *n* number of equations is straightforward. The elimination procedure starts with the first row as the pivot row and continues row after row down to one row before the last. At each step, the pivot row is used to eliminate the terms that are below the pivot element in all the rows that are below. Once the original system of equations is changed to upper triangular form, back substitution is used for determining the solution.

When the Gauss elimination method is programmed, it is convenient and more efficient to create one matrix that includes the matrix of coefficients $[a]$ and the right-hand-side vector $[b]$. This is done by appending the vector $[b]$ to the matrix $[a]$, as shown in Example 4-2, where the Gauss elimination method is programmed in MATLAB.

Example 4-2: MATLAB user-defined function for solving a system of equations using Gauss elimination.

Write a user-defined MATLAB function for solving a system of linear equations, $[a][x]=[b]$, using the Gauss elimination method. For function name and arguments, use `x = Gauss(a,b)`, where `a` is the matrix of coefficients, `b` is the right-hand-side column vector of constants, and `x` is a column vector of the solution.

Use the user-defined function `Gauss` to

(*a*) Solve the system of equations of Example 4-1.

(b) Solve the system of Eqs. (4.1).

SOLUTION

The following user-defined MATLAB function solves a system of linear equations. The program starts by appending the column vector $[b]$ to the matrix $[a]$. The new augmented matrix, named in the program `ab`, has the form:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} & b_3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} & b_n \end{bmatrix}$$

Next, the Gauss elimination procedure is applied (forward elimination). The matrix is changed such that all the elements below the diagonal of `a` are zero:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ 0 & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\ 0 & 0 & a_{33} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{nn} & b_n \end{bmatrix}$$

At the end of the program, back substitution is used to solve for the unknowns, and the results are assigned to the column vector `x`.

Program 4-1: User-defined function. Gauss elimination.

```
function x = Gauss(a,b)
% The function solves a system of linear equations [a][x] = [b] using the Gauss
% elimination method.
% Input variables:
% a  The matrix of coefficients.
% b  Right-hand-side column vector of constants.
% Output variable:
% x  A column vector with the solution.

ab = [a,b];
[R, C] = size(ab);
for j = 1:R - 1
   for i = j + 1:R
      ab(i,j:C) = ab(i,j:C)- ab(i,j)/ab(j,j)*ab(j,j:C);
   end
end
x = zeros(R,1);
x(R) = ab(R,C)/ab(R,R);
for i = R - 1:-1:1
    x(i) = (ab(i,C) - ab(i,i + 1:R)*x(i + 1:R))/ab(i,i);
end
```

Append the column vector $[b]$ to the matrix $[a]$.

Pivot element.

The multiplier m_{ij}.

Pivot equation.

Gauss elimination procedure (forward elimination).

Back substitution.

The user-defined function `Gauss` is next used in the Command Window, first to solve the system of equations of Example 4-1, and then to solve the system of Eqs. (4.1).

```
>> A=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
>> B = [12; -6.5; 16; 17];
>> sola = Gauss(A,B)
sola =
    2.0000
    4.0000
   -3.0000
    0.5000
>> C = [9 -4 -2 0; -4 17 -6 -3; -2 -6 14 -6; 0 -3 -6 11];
>> D = [24; -16; 0; 18];
>> solb = Gauss(C,D)
solb =
    4.0343
    1.6545
    2.8452
    3.6395
```

Solution for part (*a*).

Solution for part (*b*).

4.2.1 Potential Difficulties When Applying the Gauss Elimination Method

The pivot element is zero

Since the pivot row is divided by the pivot element, a problem will arise during the execution of the Gauss elimination procedure if the value of the pivot element is equal to zero. As shown in the next section, this situation can be corrected by changing the order of the rows. In a procedure called pivoting, the pivot row that has the zero pivot element is exchanged with another row that has a nonzero pivot element.

The pivot element is small relative to the other terms in the pivot row

Significant errors due to rounding can occur when the pivot element is small relative to other elements in the pivot row. This is illustrated by the following example.

Consider the following system of simultaneous equations for the unknowns x_1 and x_2:

$$\begin{aligned} 0.0003x_1 + 12.34x_2 &= 12.343 \\ 0.4321x_1 + x_2 &= 5.321 \end{aligned} \tag{4.15}$$

The exact solution of the system is $x_1 = 10$ and $x_2 = 1$.

The error due to rounding is illustrated by solving the system using Gaussian elimination on a machine with limited precision so that only four significant figures are retained with rounding. When the first equation of Eqs. (4.15) is entered, the constant on the right-hand side is rounded to 12.34.

The solution starts by using the first equation as the pivot equation and $a_{11} = 0.0003$ as the pivot coefficient. In the first step, the pivot equation is multiplied by $m_{21} = 0.4321/0.0003 = 1440$. With four significant figures and rounding, this operation gives:

$$(1440)(0.0003x_1 + 12.34x_2) = 1440 \cdot 12.34$$

or:

$$0.4320x_1 + 17770x_2 = 17770$$

The result is next subtracted from the second equation in Eqs. (4.15):

$$\begin{array}{rl} & 0.4321x_1 + x_2 = 5.321 \\ - & \underline{0.4320x_1 + 17770x_2 = 17770} \\ & 0.0001x_1 - 17770x_2 = -17760 \end{array}$$

After this operation, the system is:

$$\begin{aligned} 0.0003x_1 + 12.34x_2 &= 12.34 \\ 0.0001x_1 - 17770x_2 &= -17760 \end{aligned}$$

Note that the a_{21} element is not zero but a very small number. Next, the value of x_2 is calculated from the second equation:

$$x_2 = \frac{-17760}{-17770} = 0.9994$$

Then x_2 is substituted in the first equation, which is solved for x_1:

$$x_1 = \frac{12.34 - (12.34 \cdot 0.9994)}{0.0003} = \frac{12.34 - 12.33}{0.0003} = \frac{0.01}{0.0003} = 33.33$$

The solution that is obtained for x_1 is obviously incorrect. The incorrect value is obtained because the magnitude of a_{11} is small when compared to the magnitude of a_{12}. Consequently, a relatively small error (due to round-off arising from the finite precision of a computing machine) in the value of x_2 can lead to a large error in the value of x_1.

The problem can be easily remedied by exchanging the order of the two equations in Eqs. (4.15):

$$\begin{aligned} 0.4321x_1 + x_2 &= 5.321 \\ 0.0003x_1 + 12.34x_2 &= 12.343 \end{aligned} \tag{4.16}$$

Now, as the first equation is used as the pivot equation, the pivot coefficient is $a_{11} = 0.4321$. In the first step, the pivot equation is multiplied by $m_{21} = 0.0003/0.4321 = 0.0006943$. With four significant figures and rounding this operation gives:

$$(0.0006943)(0.4321x_1 + x_2) = 0.0006943 \cdot 5.321$$

or:

$$0.0003x_1 + 0.0006943x_2 = 0.003694$$

The result is next subtracted from the second equation in Eqs. (4.16):

$$\begin{array}{rrl} & 0.0003x_1 + 12.34x_2 &= 12.34 \\ - & 0.0003x_1 + 0.0006943x_2 &= 0.003694 \\ \hline & 12.34x_2 &= 12.34 \end{array}$$

After this operation, the system is:

$$\begin{aligned} 0.4321x_1 + x_2 &= 5.321 \\ 0x_1 + 12.34x_2 &= 12.34 \end{aligned}$$

Next, the value of x_2 is calculated from the second equation:

$$x_2 = \frac{12.34}{12.34} = 1$$

Then x_2 is substituted in the first equation that is solved for x_1:

$$x_1 = \frac{5.321 - 1}{0.4321} = 10$$

The solution that is obtained now is the exact solution.

In general, a more accurate solution is obtained when the equations are arranged (and rearranged every time a new pivot equation is used) such that the pivot equation has the largest possible pivot element. This is explained in more detail in the next section.

Round-off errors can also be significant when solving large systems of equations even when all the coefficients in the pivot row are of the same order of magnitude. This can be caused by a large number of operations (multiplication, division, addition, and subtraction) associated with large systems.

4.3 GAUSS ELIMINATION WITH PIVOTING

In the Gauss elimination procedure, the pivot equation is divided by the pivot coefficient. This, however, cannot be done if the pivot coefficient is zero. For example, for the following system of three equations:

$$\begin{aligned} 0x_1 + 2x_2 + 3x_3 &= 46 \\ 4x_1 - 3x_2 + 2x_3 &= 16 \\ 2x_1 + 4x_2 - 3x_3 &= 12 \end{aligned}$$

the procedure starts by taking the first equation as the pivot equation and the coefficient of x_1, which is 0, as the pivot coefficient. To eliminate the term $4x_1$ in the second equation, the pivot equation is supposed to be multiplied by $4/0$ and then subtracted from the second equation. Obviously, this is not possible when the pivot element is equal to zero. The division by zero can be avoided if the order in which the equations are written is changed such that in the first equation the first coefficient is not zero. For example, in the system above, this can be done by exchanging the first two equations.

In the general Gauss elimination procedure, an equation (or a row) can be used as the pivot equation (pivot row) only if the pivot coefficient (pivot element) is not zero. If the pivot element is zero, the equation (i.e., the row) is exchanged with one of the equations (rows) that are below, which has a nonzero pivot coefficient. This exchange of rows, illustrated in Fig. 4-16, is called ***pivoting***.

After the first step, the second equation has a pivot element that is equal to zero.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & 0 & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

Using pivoting, the second equation is exchanged with the third equation.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & 0 & a'_{23} & a'_{24} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_3 \\ b'_2 \\ b'_4 \end{bmatrix}$$

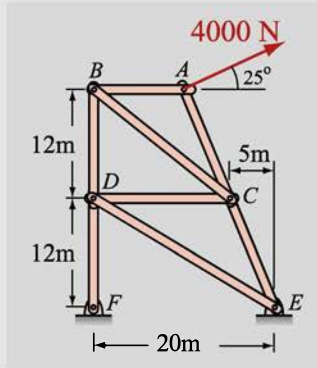
Figure 4-16: Illustration of pivoting.

Additional comments about pivoting

- If during the Gauss elimination procedure a pivot equation has a pivot element that is equal to zero, then if the system of equations that is being solved has a solution, an equation with a nonzero element in the pivot position can always be found.
- The numerical calculations are less prone to error and will have fewer round-off errors (see Section 4.2.1) if the pivot element has a larger numerical absolute value compared to the other elements in the same row. Consequently, among all the equations that can be exchanged to be the pivot equation, it is better to select the equation whose pivot element has the largest absolute numerical value. Moreover, it is good to employ pivoting for the purpose of having a pivot equation with the pivot element that has a largest absolute numerical value at all times (even when pivoting is not necessary).

The addition of pivoting to the programming of the Gauss elimination method is shown in the next example. The addition of pivoting every time a new pivot equation is used, such that the pivot row will have the largest absolute pivot element, is assigned as an exercise in Problem 4.21.

Example 4-3: MATLAB user-defined function for solving a system of equations using Gauss elimination with pivoting.



Write a user-defined MATLAB function for solving a system of linear equations $[a][x] = [b]$ using the Gauss elimination method with pivoting. Name the function `x = GaussPivot(a,b)`, where `a` is the matrix of coefficients, `b` is the right-hand-side column vector of constants, and `x` is a column vector of the solution. Use the function to determine the forces in the loaded eight-member truss that is shown in the figure (same as in Fig. 4-2).

SOLUTION

The forces in the eight truss members are determined from the set of eight equations, Eqs. (4.2). The equations are derived by drawing free body diagrams of pins A, B, C, and D and applying equations of equilibrium. The equations are rewritten here in a matrix form (intentionally, the equations are written in an order that requires pivoting):

$$\begin{bmatrix} 0 & 0.9231 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -0.3846 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0.8575 & 0 \\ 1 & 0 & -0.7809 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.3846 & -0.7809 & 0 & -1 & 0.3846 & 0 & 0 \\ 0 & 0.9231 & 0.6247 & 0 & 0 & -0.9231 & 0 & 0 \\ 0 & 0 & 0.6247 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -0.5145 & -1 \end{bmatrix} \begin{bmatrix} F_{AB} \\ F_{AC} \\ F_{BC} \\ F_{BD} \\ F_{CD} \\ F_{CE} \\ F_{DE} \\ F_{DF} \end{bmatrix} = \begin{bmatrix} 1690 \\ 3625 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.17}$$

The function `GaussPivot` is created by modifying the function `Gauss` listed in the solution of Example 4-2.

Program 4-2: User-defined function. Gauss elimination with pivoting.

```
function x = GaussPivot(a,b)
% The function solves a system of linear equations ax = b using the Gauss
% elimination method with pivoting.
% Input variables:
% a  The matrix of coefficients.
% b  Right-hand-side column vector of constants.
% Output variable:
% x  A column vector with the solution.

ab = [a,b];
[R, C] = size(ab);
for j = 1:R - 1
% Pivoting section starts
   if ab(j,j) = = 0
```

Check if the pivot element is zero.

```
      for k = j + 1:R
          if ab(k,j) ~ = 0
              abTemp = ab(j,:);
              ab(j,:) = ab(k,:);
              ab(k,:) = abTemp;
              break
          end
      end
    end
% Pivoting section ends
    for i = j + 1:R
        ab(i,j:C) = ab(i,j:C) - ab(i,j)/ab(j,j)*ab(j,j:C);
    end
end
x = zeros(R,1);
x(R) = ab(R,C)/ab(R,R);
for i = R - 1:-1:1
    x(i) = (ab(i,C) - ab(i,i + 1:R)*x(i + 1:R))/ab(i,i);
end
```

If pivoting is required, search in the rows below for a row with nonzero pivot element.

Switch the row that has a zero pivot element with the row that has a nonzero pivot element.

Stop searching for a row with a nonzero pivot element.

The user-defined function `GaussPivot` is next used in a script file program to solve the system of equations Eq. (4.17).

```
% Example 4-3
a=[0 0.9231 0 0 0 0 0 0; -1 -0.3846 0 0 0 0 0 0; 0 0 0 0 1 0 0.8575 0; 1 0 -0.7809 0 0 0 0 0
    0 -0.3846 -0.7809 0 -1 0.3846 0 0; 0 0.9231 0.6247 0 0 -0.9231 0 0
    0 0 0.6247 -1 0 0 0 0; 0 0 0 1 0 0 -0.5145 -1];
b = [1690;3625;0;0;0;0;0;0];
Forces = GaussPivot(a,b)
```

When the script file is executed, the following solution is displayed in the Command Window.

```
Forces =
  -4.3291e+003
   1.8308e+003
  -5.5438e+003
  -3.4632e+003
   2.8862e+003
  -1.9209e+003
  -3.3659e+003
  -1.7315e+003
>>
```

$$\begin{bmatrix} F_{AB} \\ F_{AC} \\ F_{BC} \\ F_{BD} \\ F_{CD} \\ F_{CE} \\ F_{DE} \\ F_{DF} \end{bmatrix}$$

4.4 GAUSS–JORDAN ELIMINATION METHOD

The Gauss–Jordan elimination method is a procedure for solving a system of linear equations, $[a][x] = [b]$. In this procedure, a system of equations that is given in a general form is manipulated into an equivalent system of equations in ***diagonal*** form (see Section 4.1.1) with normalized elements along the diagonal. This means that when the diagonal form of the matrix of the coefficients, $[a]$, is reduced to the identity matrix, the new vector $[b']$ is the solution. The starting point of the procedure is a system of equations given in a general form (the illustration that follows is for a system of four equations):

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 && (4.18a)\\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 && (4.18b)\\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3 && (4.18c)\\
a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= b_4 && (4.18d)
\end{aligned} \qquad (4.18)$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 4-17: Matrix form of a system of four equations.

The matrix form of the system is shown in Fig. 4-17. In the Gauss–Jordan elimination method, the system of equations is manipulated to have the following diagonal form:

$$\begin{aligned}
x_1 + 0 + 0 + 0 &= b'_1 && (4.19a)\\
0 + x_2 + 0 + 0 &= b'_2 && (4.19b)\\
0 + 0 + x_3 + 0 &= b'_3 && (4.19c)\\
0 + 0 + 0 + x_4 &= b'_4 && (4.19d)
\end{aligned} \qquad (4.19)$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

Figure 4-18: Matrix form of the equivalent system after applying the Gauss–Jordan method.

The matrix form of the equivalent system is shown in Fig. 4-18. The terms on the right-hand side of the equations (column $[b']$) are the solution. In matrix form, the matrix of the coefficients is transformed into an identity matrix.

Gauss–Jordan elimination procedure

The Gauss–Jordan elimination procedure for transforming the system of equations from the form in Eqs. (4.18) to the form in Eqs. (4.19) is the same as the Gauss elimination procedure (see Section 4.2), except for the following two differences:

- The pivot equation is normalized by dividing all the terms in the equation by the pivot coefficient. This makes the pivot coefficient equal to 1.
- The pivot equation is used to eliminate the off-diagonal terms in **ALL** the other equations. This means that the elimination process is applied to the equations (rows) that are above and below the pivot equation. (In the Gaussian elimination method, only elements that are below the pivot element are eliminated.)

When the Gauss–Jordan procedure is programmed, it is convenient and more efficient to create a single matrix that includes the matrix of coefficients $[a]$ and the vector $[b]$. This is done by appending the vector $[b]$ to the matrix $[a]$. The augmented matrix at the starting point of the procedure is shown (for a system of four equations) in Fig. 4-19*a*. At the end of the procedure, shown in Fig. 4-19*b*, the elements of $[a]$ are replaced by an identity matrix, and the column $[b']$ is the solution.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ a_{21} & a_{22} & a_{23} & a_{24} & b_2 \\ a_{31} & a_{32} & a_{33} & a_{34} & b_3 \\ a_{41} & a_{42} & a_{43} & a_{44} & b_4 \end{bmatrix} \xrightarrow{\text{Gauss–Jordan procedure}} \begin{bmatrix} 1 & 0 & 0 & 0 & b'_1 \\ 0 & 1 & 0 & 0 & b'_2 \\ 0 & 0 & 1 & 0 & b'_3 \\ 0 & 0 & 0 & 1 & b'_4 \end{bmatrix}$$

(*a*) (*b*)

Figure 4-19: Schematic illustration of the Gauss–Jordan method.

The Gauss–Jordan method can also be used for solving several systems of equations $[a][x] = [b]$ that have the same coefficients $[a]$ but different right-hand-side vectors $[b]$. This is done by augmenting the matrix $[a]$ to include all of the vectors $[b]$. In Section 4.6.2 the method is used in this way for calculating the inverse of a matrix.

The Gauss–Jordan elimination method is demonstrated in Example 4-4 where it is used to solve the set of equations solved in Example 4-1.

Example 4-4: Solving a set of four equations using Gauss–Jordan elimination.

Solve the following set of four equations using the Gauss–Jordan elimination method.

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ -6x_1 + 7x_2 + 6.5x_3 - 6x_4 &= -6.5 \\ x_1 + 7.5x_2 + 6.25x_3 + 5.5x_4 &= 16 \\ -12x_1 + 22x_2 + 15.5x_3 - x_4 &= 17 \end{aligned}$$

SOLUTION

The solution is carried out by using the matrix form of the equations. In matrix form, the system is:

$$\begin{bmatrix} 4 & -2 & -3 & 6 \\ -6 & 7 & 6.5 & -6 \\ 1 & 7.5 & 6.25 & 5.5 \\ -12 & 22 & 15.5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ -6.5 \\ 16 \\ 17 \end{bmatrix}$$

For the numerical procedure, a new matrix is created by augmenting the coefficient matrix to include the right-hand side of the equation:

$$\begin{bmatrix} 4 & -2 & -3 & 6 & 12 \\ -6 & 7 & 6.5 & -6 & -6.5 \\ 1 & 7.5 & 6.25 & 5.5 & 16 \\ -12 & 22 & 15.5 & -1 & 17 \end{bmatrix}$$

The first pivoting row is the first row, and the first element in this row is the pivot element. The row is normalized by dividing it by the pivot element:

$$\begin{bmatrix} \frac{4}{4} & \frac{-2}{4} & \frac{-3}{4} & \frac{6}{4} & \frac{12}{4} \\ -6 & 7 & 6.5 & -6 & -6.5 \\ 1 & 7.5 & 6.25 & 5.5 & 16 \\ -12 & 22 & 15.5 & -1 & 17 \end{bmatrix} = \begin{bmatrix} 1 & -0.5 & -0.75 & 1.5 & 3 \\ -6 & 7 & 6.5 & -6 & -6.5 \\ 1 & 7.5 & 6.25 & 5.5 & 16 \\ -12 & 22 & 15.5 & -1 & 17 \end{bmatrix}$$

Next, all the first elements in rows 2, 3, and 4 are eliminated:

$$\begin{bmatrix} 1 & -0.5 & -0.75 & 1.5 & 3 \\ -6 & 7 & 6.5 & -6 & -6.5 \\ 1 & 7.5 & 6.25 & 5.5 & 16 \\ -12 & 22 & 15.5 & -1 & 17 \end{bmatrix} \begin{matrix} \\ \leftarrow -(-6)[1\ -0.5\ -0.75\ 1.5\ 3] \\ \leftarrow -(1)[1\ -0.5\ -0.75\ 1.5\ 3] \\ \leftarrow -(-12)[1\ -0.5\ -0.75\ 1.5\ 3] \end{matrix} = \begin{bmatrix} 1 & -0.5 & -0.75 & 1.5 & 3 \\ 0 & 4 & 2 & 3 & 11.5 \\ 0 & 8 & 7 & 4 & 13 \\ 0 & 16 & 6.5 & 17 & 53 \end{bmatrix}$$

The next pivot row is the second row, with the second element as the pivot element. The row is normalized by dividing it by the pivot element:

$$\begin{bmatrix} 1 & -0.5 & -0.75 & 1.5 & 3 \\ 0 & \frac{4}{4} & \frac{2}{4} & \frac{3}{4} & \frac{11.5}{4} \\ 0 & 8 & 7 & 4 & 13 \\ 0 & 16 & 6.5 & 17 & 53 \end{bmatrix} = \begin{bmatrix} 1 & -0.5 & -0.75 & 1.5 & 3 \\ 0 & 1 & 0.5 & 0.75 & 2.875 \\ 0 & 8 & 7 & 4 & 13 \\ 0 & 16 & 6.5 & 17 & 53 \end{bmatrix}$$

Next, all the second elements in rows 1, 3, and 4 are eliminated:

$$\begin{bmatrix} 1 & -0.5 & -0.75 & 1.5 & 3 \\ 0 & 1 & 0.5 & 0.75 & 2.875 \\ 0 & 8 & 7 & 4 & 13 \\ 0 & 16 & 6.5 & 17 & 53 \end{bmatrix} \begin{matrix} \leftarrow -(-0.5)[0\ 1\ 0.5\ 0.75\ 2.875] \\ \\ \leftarrow -(8)[0\ 1\ 0.5\ 0.75\ 2.875] \\ \leftarrow -(16)[0\ 1\ 0.5\ 0.75\ 2.875] \end{matrix} = \begin{bmatrix} 1 & 0 & -0.5 & 1.875 & 4.4375 \\ 0 & 1 & 0.5 & 0.75 & 2.875 \\ 0 & 0 & 3 & -2 & -10 \\ 0 & 0 & -1.5 & 5 & 7 \end{bmatrix}$$

The next pivot row is the third row, with the third element as the pivot element. The row is normalized by dividing it by the pivot element:

$$\begin{bmatrix} 1 & 0 & -0.5 & 1.875 & 4.4375 \\ 0 & 1 & 0.5 & 0.75 & 2.875 \\ 0 & 0 & \frac{3}{3} & \frac{-2}{3} & \frac{-10}{3} \\ 0 & 0 & -1.5 & 5 & 7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -0.5 & 1.875 & 4.4375 \\ 0 & 1 & 0.5 & 0.75 & 2.875 \\ 0 & 0 & 1 & -0.667 & -3.333 \\ 0 & 0 & -1.5 & 5 & 7 \end{bmatrix}$$

Next, all the third elements in rows 1, 2, and 4 are eliminated:

$$\begin{bmatrix} 1 & 0 & -0.5 & 1.875 & 4.4375 \\ 0 & 1 & 0.5 & 0.75 & 2.875 \\ 0 & 0 & 1 & -0.667 & -3.333 \\ 0 & 0 & -1.5 & 5 & 7 \end{bmatrix} \begin{matrix} \leftarrow -(-0.5)[0\ 0\ 1\ -0.667\ -3.333] \\ \leftarrow -(0.5)[0\ 0\ 1\ -0.667\ -3.333] \\ \\ \leftarrow -(-1.5)[0\ 0\ 1\ -0.667\ -3.333] \end{matrix} = \begin{bmatrix} 1 & 0 & 0 & 1.5417 & 2.7708 \\ 0 & 1 & 0 & 1.0833 & 4.5417 \\ 0 & 0 & 1 & -0.667 & -3.333 \\ 0 & 0 & 0 & 4 & 2 \end{bmatrix}$$

The next pivot row is the fourth row, with the fourth element as the pivot element. The row is normalized by dividing it by the pivot element:

$$\begin{bmatrix} 1 & 0 & 0 & 1.5417 & 2.7708 \\ 0 & 1 & 0 & 1.0833 & 4.5417 \\ 0 & 0 & 1 & -0.667 & -3.333 \\ 0 & 0 & 0 & \frac{4}{4} & \frac{2}{4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1.5417 & 2.7708 \\ 0 & 1 & 0 & 1.0833 & 4.5417 \\ 0 & 0 & 1 & -0.667 & -3.333 \\ 0 & 0 & 0 & 1 & 0.5 \end{bmatrix}$$

Next, all the fourth elements in rows 1, 2, and 3 are eliminated:

$$\begin{bmatrix} 1 & 0 & 0 & 1.5417 & 2.7708 \\ 0 & 1 & 0 & 1.0833 & 4.5417 \\ 0 & 0 & 1 & -0.667 & -3.333 \\ 0 & 0 & 0 & 1 & 0.5 \end{bmatrix} \begin{matrix} \leftarrow -(1.5417)\begin{bmatrix} 0 & 0 & 0 & 1 & 0.5 \end{bmatrix} \\ \leftarrow -(1.0833)\begin{bmatrix} 0 & 0 & 0 & 1 & 0.5 \end{bmatrix} \\ \leftarrow -(-0.667)\begin{bmatrix} 0 & 0 & 0 & 1 & 0.5 \end{bmatrix} \\ \end{matrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 0 & -3 \\ 0 & 0 & 0 & 1 & 0.5 \end{bmatrix}$$

The solution is:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ -3 \\ 0.5 \end{bmatrix}$$

The Gauss–Jordan elimination method with pivoting

It is possible that the equations are written in such an order that during the elimination procedure a pivot equation has a pivot element that is equal to zero. Obviously, in this case it is impossible to normalize the pivot row (divide by the pivot element). As with the Gauss elimination method, the problem can be corrected by using pivoting. This is left as an exercise in Problem 4.22.

4.5 *LU* DECOMPOSITION METHOD

Background

The Gauss elimination method consists of two parts. The first part is the elimination procedure in which a system of linear equations that is given in a general form, $[a][x] = [b]$, is transformed into an equivalent system of equations $[a'][x] = [b']$ in which the matrix of coefficients $[a']$ is upper triangular. In the second part, the equivalent system is solved by using back substitution. The elimination procedure requires many mathematical operations and significantly more computing time than the back substitution calculations. During the elimination procedure, the matrix of coefficients $[a]$ and the vector $[b]$ are both changed. This means that if there is a need to solve systems of equations that have the same left-hand-side terms (same coefficient matrix $[a]$) but different right-hand-side constants (different vectors $[b]$), the elimination procedure has to be carried out for each $[b]$ again. Ideally, it would be better if the operations on the matrix of coefficients $[a]$ were dissociated from those on the vector of constants $[b]$. In this way, the elimination procedure with $[a]$ is done only once and then is used for solving systems of equations with different vectors $[b]$.

One option for solving various systems of equations $[a][x]=[b]$ that have the same coefficient matrices $[a]$ but different constant vectors $[b]$ is to first calculate the inverse of the matrix $[a]$. Once the inverse matrix $[a]^{-1}$ is known, the solution can be calculated by:

$$[x] = [a]^{-1}[b]$$

Calculating the inverse of a matrix, however, requires many mathematical operations, and is computationally inefficient. A more efficient method of solution for this case is the *LU* decomposition method.

In the *LU* decomposition method, the operations with the matrix $[a]$ are done without using, or changing, the vector $[b]$, which is used only in the substitution part of the solution. The *LU* decomposition method can be used for solving a single system of linear equations, but it is especially advantageous for solving systems that have the same coefficient matrices $[a]$ but different constant vectors $[b]$.

The LU decomposition method

The *LU* decomposition method is a method for solving a system of linear equations $[a][x]=[b]$. In this method the matrix of coefficients $[a]$ is decomposed (factored) into a product of two matrices $[L]$ and $[U]$:

$$[a] = [L][U] \tag{4.20}$$

where the matrix $[L]$ is a lower triangular matrix and $[U]$ is an upper triangular matrix. With this decomposition, the system of equations to be solved has the form:

$$[L][U][x] = [b] \tag{4.21}$$

To solve this equation, the product $[U][x]$ is defined as:

$$[U][x] = [y] \tag{4.22}$$

and is substituted in Eq. (4.21) to give:

$$[L][y] = [b] \tag{4.23}$$

Now, the solution $[x]$ is obtained in two steps. First, Eq. (4.23) is solved for $[y]$. Then, the solution $[y]$ is substituted in Eq. (4.22), and that equation is solved for $[x]$.

Since the matrix $[L]$ is a lower triangular matrix, the solution $[y]$ in Eq. (4.23) is obtained by using the forward substitution method. Once $[y]$ is known and is substituted in Eq. (4.22), this equation is solved by using back substitution, since $[U]$ is an upper triangular matrix.

For a given matrix $[a]$ several methods can be used to determine the corresponding $[L]$ and $[U]$. Two of the methods, one related to the Gauss elimination method and another called Crout's method, are described next.

4.5.1 LU Decomposition Using the Gauss Elimination Procedure

When the Gauss elimination procedure is applied to a matrix $[a]$, the elements of the matrices $[L]$ and $[U]$ are actually calculated. The upper triangular matrix $[U]$ is the matrix of coefficients $[a]$ that is obtained at the end of the procedure, as shown in Figs. 4-8 and 4-14. The lower triangular matrix $[L]$ is not written explicitly during the procedure, but the elements that make up the matrix are actually calculated along the way. The elements of $[L]$ on the diagonal are all 1, and the elements below the diagonal are the multipliers m_{ij} that multiply the pivot equation when it is used to eliminate the elements below the pivot coefficient (see the ***Gauss elimination procedure*** in Section 4.2). For the case of a system of four equations, the matrix of coefficients $[a]$ is (4×4), and the decomposition has the form:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 \\ m_{31} & m_{32} & 1 & 0 \\ m_{41} & m_{42} & m_{43} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & 0 & a'''_{44} \end{bmatrix} \tag{4.24}$$

A numerical example illustrating LU decomposition is given next. It uses the information in the solution of Example 4-1, where a system of four equations is solved by using the Gauss elimination method. The matrix $[a]$ can be written from the given set of equations in the problem statement, and the matrix $[U]$ can be written from the set of equations at the end of ***step 3*** (page 107). The matrix $[L]$ can be written by using the multipliers that are calculated in the solution. The decomposition has the form:

$$\begin{bmatrix} 4 & -2 & -3 & 6 \\ -6 & 7 & 6.5 & -6 \\ 1 & 7.5 & 6.25 & 5.5 \\ -12 & 22 & 15.5 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1.5 & 1 & 0 & 0 \\ 0.25 & 2 & 1 & 0 \\ -3 & 4 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 & -2 & -3 & 6 \\ 0 & 4 & 2 & 3 \\ 0 & 0 & 3 & -2 \\ 0 & 0 & 0 & 4 \end{bmatrix} \tag{4.25}$$

The decomposition in Eq. (4.25) can be verified by using MATLAB:

```
>> L = [1,0,0,0;-1.5,1,0,0;0.25,2,1,0;-3,4,-0.5,1]
L =
    1.0000         0         0         0
   -1.5000    1.0000         0         0
    0.2500    2.0000    1.0000         0
   -3.0000    4.0000   -0.5000    1.0000
>> U = [4,-2,-3,6;0,4,2,3;0,0,3,-2;0,0,0,4]
```

```
U =
     4    -2    -3     6
     0     4     2     3
     0     0     3    -2
     0     0     0     4
>> L*U
ans =
   4.0000   -2.0000   -3.0000    6.0000
  -6.0000    7.0000    6.5000   -6.0000
   1.0000    7.5000    6.2500    5.5000
 -12.0000   22.0000   15.5000   -1.0000
```

Multiplication of the matrices L and U verifies that the answer is the matrix $[a]$.

4.5.2 LU Decomposition Using Crout's Method

In this method the matrix $[a]$ is decomposed into the product $[L][U]$, where the diagonal elements of the matrix $[U]$ are all 1s. It turns out that in this case, the elements of both matrices can be determined using formulas that can be easily programmed. This is illustrated for a system of four equations. In Crout's method, the LU decomposition has the form:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix} \begin{bmatrix} 1 & U_{12} & U_{13} & U_{14} \\ 0 & 1 & U_{23} & U_{24} \\ 0 & 0 & 1 & U_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.26}$$

Executing the matrix multiplication on the right-hand side of the equation gives:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} L_{11} & (L_{11}U_{12}) & (L_{11}U_{13}) & (L_{11}U_{14}) \\ L_{21} & (L_{21}U_{12}+L_{22}) & (L_{21}U_{13}+L_{22}U_{23}) & (L_{21}U_{14}+L_{22}U_{24}) \\ L_{31} & (L_{31}U_{12}+L_{32}) & (L_{31}U_{13}+L_{32}U_{23}+L_{33}) & (L_{31}U_{14}+L_{32}U_{24}+L_{33}U_{34}) \\ L_{41} & (L_{41}U_{12}+L_{42}) & (L_{41}U_{13}+L_{42}U_{23}+L_{43}) & (L_{41}U_{14}+L_{42}U_{24}+L_{43}U_{34}+L_{44}) \end{bmatrix} \tag{4.27}$$

The elements of the matrices $[L]$ and $[U]$ can be determined by solving Eq. (4.27). The solution is obtained by equating the corresponding elements of the matrices on both sides of the equation. Looking at Eq. (4.27), one can observe that the elements of the matrices $[L]$ and $[U]$ can be easily determined row after row from the known elements of $[a]$ and the elements of $[L]$ and $[U]$ that are already calculated. Starting with the first row, the value of L_{11} is calculated from $L_{11} = a_{11}$. Once L_{11} is known, the values of U_{12}, U_{13}, and U_{14} are calculated by:

$$U_{12} = \frac{a_{12}}{L_{11}} \quad U_{13} = \frac{a_{13}}{L_{11}} \quad \text{and} \quad U_{14} = \frac{a_{14}}{L_{11}} \tag{4.28}$$

Moving on to the second row, the value of L_{21} is calculated from $L_{21} = a_{21}$ and the value of L_{22} is calculated from:

$$L_{22} = a_{22} - L_{21}U_{12} \tag{4.29}$$

With the values of L_{21} and L_{22} known, the values of U_{23} and U_{24} are determined from:

$$U_{23} = \frac{a_{23} - L_{21}U_{13}}{L_{22}} \quad \text{and} \quad U_{24} = \frac{a_{24} - L_{21}U_{14}}{L_{22}} \tag{4.30}$$

In the third row:

$$L_{31} = a_{31}, \quad L_{32} = a_{32} - L_{31}U_{12}, \quad \text{and} \quad L_{33} = a_{33} - L_{31}U_{13} - L_{32}U_{23} \tag{4.31}$$

Once the values of L_{31}, L_{32}, and L_{33} are known, the value of U_{34} is calculated by:

$$U_{34} = \frac{a_{34} - L_{31}U_{14} - L_{32}U_{24}}{L_{33}} \tag{4.32}$$

In the fourth row, the values of L_{41}, L_{42}, L_{43}, and L_{44} are calculated by:

$$L_{41} = a_{41}, \quad L_{42} = a_{42} - L_{41}U_{12}, \quad L_{43} = a_{43} - L_{41}U_{13} - L_{42}U_{23}, \quad \text{and}$$
$$L_{44} = a_{44} - L_{41}U_{14} - L_{42}U_{24} - L_{43}U_{34} \tag{4.33}$$

A procedure for determining the elements of the matrices $[L]$ and $[U]$ can be written by following the calculations in Eqs. (4.28) through (4.33). If $[a]$ is an $(n \times n)$ matrix, the elements of $[L]$ and $[U]$ are given by:

Step 1: Calculating the first column of $[L]$:

$$\text{for} \quad i = 1, 2, \ldots, n \qquad L_{i1} = a_{i1} \tag{4.34}$$

Step 2: Substituting 1s in the diagonal of $[U]$:

$$\text{for} \quad i = 1, 2, \ldots, n \qquad U_{ii} = 1 \tag{4.35}$$

Step 3: Calculating the elements in the first row of $[U]$ (except U_{11} which was already calculated):

$$\text{for} \quad j = 2, 3, \ldots, n \qquad U_{1j} = \frac{a_{1j}}{L_{11}} \tag{4.36}$$

Step 4: Calculating the rest of the elements row after row (i is the row number and j is the column number). The elements of $[L]$ are calculated first because they are used for calculating the elements of $[U]$:

for $i = 2, 3, \ldots, n$

$$\text{for} \quad j = 2, 3, \ldots, i \qquad L_{ij} = a_{ij} - \sum_{k=1}^{k=j-1} L_{ik}U_{kj} \tag{4.37}$$

$$\text{for} \quad j = (i+1), (i+2), \ldots, n \qquad U_{ij} = \frac{a_{ij} - \sum_{k=1}^{k=i-1} L_{ik}U_{kj}}{L_{ii}} \tag{4.38}$$

Examples 4-5 and 4-6 show how the LU decomposition with Crout's method is used for solving systems of equations. In Example 4-5 the calculations are done manually, and in Example 4-6 the decomposition is done with a user-defined MATLAB program.

Example 4-5: Solving a set of four equations using LU decomposition with Crout's method.

Solve the following set of four equations (the same as in Example 4-1) using LU decomposition with Crout's method.

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ -6x_1 + 7x_2 + 6.5x_3 - 6x_4 &= -6.5 \\ x_1 + 7.5x_2 + 6.25x_3 + 5.5x_4 &= 16 \\ -12x_1 + 22x_2 + 15.5x_3 - x_4 &= 17 \end{aligned}$$

SOLUTION

First, the equations are written in matrix form:

$$\begin{bmatrix} 4 & -2 & -3 & 6 \\ -6 & 7 & 6.5 & -6 \\ 1 & 7.5 & 6.25 & 5.5 \\ -12 & 22 & 15.5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ -6.5 \\ 16 \\ 17 \end{bmatrix} \tag{4.39}$$

Next, the matrix of coefficients $[a]$ is decomposed into the product $[L][U]$, as shown in Eq. (4.27). The decomposition is done by following the steps listed on the previous page:

Step 1: Calculating the first column of $[L]$:

for $i = 1, 2, 3, 4 \quad L_{i1} = a_{i1}$: $L_{11} = 4, \quad L_{21} = -6, \quad L_{31} = 1, \quad L_{41} = -12$

Step 2: Substituting 1s in the diagonal of $[U]$:

for $i = 1, 2, 3, 4 \quad U_{ii} = 1$: $U_{11} = 1, \quad U_{22} = 1, \quad U_{33} = 1, \quad U_{44} = 1.$

Step 3: Calculating the elements in the first row of $[U]$ (except U_{11} which was already calculated):

for $j = 2, 3, 4 \quad U_{1j} = \dfrac{a_{1j}}{L_{11}}$: $U_{12} = \dfrac{a_{12}}{L_{11}} = \dfrac{-2}{4} = -0.5, \quad U_{13} = \dfrac{a_{13}}{L_{11}} = \dfrac{-3}{4} = -0.75,$

$$U_{14} = \frac{a_{14}}{L_{11}} = \frac{6}{4} = 1.5$$

Step 4: Calculating the rest of the elements row after row, starting with the second row (i is the row number and j is the column number). In the present problem there are four rows, so i starts at 2 and ends with 4. For each value of i (each row), the elements of L are calculated first, and the elements of U are calculated subsequently. The general form of the equations is (Eqs. (4.37) and (4.38)):

for $i = 2, 3, 4,$

$$\text{for } j = 2, 3, \ldots, i \qquad L_{ij} = a_{ij} - \sum_{k=1}^{k=j-1} L_{ik} U_{kj} \tag{4.40}$$

$$\text{for } j = (i+1), (i+2), \ldots, n \qquad U_{ij} = \frac{a_{ij} - \sum_{k=1}^{k=j-1} L_{ik} U_{kj}}{L_{ii}} \tag{4.41}$$

Starting with the second row, $i = 2$,

for $j = 2$: $L_{22} = a_{22} - \sum_{k=1}^{k=1} L_{2k}U_{k2} = a_{22} - L_{21}U_{12} = 7 - (-6 \cdot -0.5) = 4$

for $j = 3, 4$: $U_{23} = \dfrac{a_{23} - \sum_{k=1}^{k=1} L_{2k}U_{k3}}{L_{22}} = \dfrac{a_{23} - (L_{21}U_{13})}{L_{22}} = \dfrac{6.5 - (-6 \cdot -0.75)}{4} = 0.5$

$$U_{24} = \frac{a_{24} - \sum_{k=1}^{k=1} L_{2k}U_{k4}}{L_{22}} = \frac{a_{24} - (L_{21}U_{14})}{L_{22}} = \frac{-6 - (-6 \cdot 1.5)}{4} = 0.75$$

Next, for the third row, $i = 3$,

for $j = 2, 3$: $L_{32} = a_{32} - \sum_{k=1}^{k=1} L_{3k}U_{k2} = a_{32} - L_{31}U_{12} = 7.5 - (1 \cdot -0.5) = 8$

$$L_{33} = a_{33} - \sum_{k=1}^{k=2} L_{3k}U_{k3} = a_{33} - (L_{31}U_{13} + L_{32}U_{23}) = 6.25 - (1 \cdot -0.75 + 8 \cdot 0.5) = 3$$

for $j = 4$: $U_{34} = \dfrac{a_{34} - \sum_{k=1}^{k=2} L_{3k}U_{k4}}{L_{33}} = \dfrac{a_{34} - (L_{31}U_{14} + L_{32}U_{24})}{L_{33}} = \dfrac{5.5 - (1 \cdot 1.5 + 8 \cdot 0.75)}{3} = -0.6667$

For the last row, $i = 4$,

for $j = 2, 3, 4$: $L_{42} = a_{42} - \sum_{k=1}^{k=1} L_{4k}U_{k2} = a_{42} - L_{41}U_{12} = 22 - (-12 \cdot -0.5) = 16$

$$L_{43} = a_{43} - \sum_{k=1}^{k=2} L_{4k}U_{k3} = a_{43} - (L_{41}U_{13} + L_{42}U_{23}) = 15.5 - (-12 \cdot -0.75 + 16 \cdot 0.5) = -1.5$$

$$L_{44} = a_{44} - \sum_{k=1}^{k=4} L_{4k}U_{k4} = a_{44} - (L_{41}U_{14} + L_{42}U_{24} + L_{43}U_{34}) = -1 - (-12 \cdot 1.5 + 16 \cdot 0.75 + -1.5 \cdot -0.6667) = 4$$

Writing the matrices $[L]$ and $[U]$ in a matrix form,

$$L = \begin{bmatrix} 4 & 0 & 0 & 0 \\ -6 & 4 & 0 & 0 \\ 1 & 8 & 3 & 0 \\ -12 & 16 & -1.5 & 4 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 1 & -0.5 & -0.75 & 1.5 \\ 0 & 1 & 0.5 & 0.75 \\ 0 & 0 & 1 & -0.6667 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To verify that the two matrices are correct, they are multiplied by using MATLAB:

```
>> L = [4 0 0 0; -6, 4 0 0; 1 8 3 0; -12 16 -1.5 4];
>> U = [1 -0.5 -0.75 1.5; 0 1 0.5 0.75; 0 0 1 -0.6667; 0 0 0 1];
>> L*U
ans =
   4.0000   -2.0000   -3.0000    6.0000
  -6.0000    7.0000    6.5000   -6.0000
   1.0000    7.5000    6.2500    5.4999
 -12.0000   22.0000   15.5000   -1.0000
```

This matrix is the same as the matrix of coefficients in Eq. (4.39) (except for round-off errors).

Once the decomposition is complete, a solution is obtained by using Eqs. (4.22) and (4.23). First, the matrix $[L]$ and the vector $[b]$ are used in Eq. (4.23), $[L][y] = [b]$, to solve for $[y]$:

$$\begin{bmatrix} 4 & 0 & 0 & 0 \\ -6 & 4 & 0 & 0 \\ 1 & 8 & 3 & 0 \\ -12 & 16 & -1.5 & 4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 12 \\ -6.5 \\ 16 \\ 17 \end{bmatrix} \tag{4.42}$$

Using forward substitution, the solution is:

$y_1 = \dfrac{12}{4} = 3$, $y_2 = \dfrac{-6.5 + 6y_1}{4} = 2.875$, $y_3 = \dfrac{16 - y_1 - 8y_2}{3} = -3.333$, and

$$y_4 = \frac{17 + 12y_1 - 16y_2 + 1.5y_3}{4} = 0.5$$

Next, the matrix $[U]$ and the vector $[y]$ are used in Eq. (4.22), $[U][x] = [y]$, to solve for $[x]$:

$$\begin{bmatrix} 1 & -0.5 & -0.75 & 1.5 \\ 0 & 1 & 0.5 & 0.75 \\ 0 & 0 & 1 & -0.6667 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 2.875 \\ -3.333 \\ 0.5 \end{bmatrix} \tag{4.43}$$

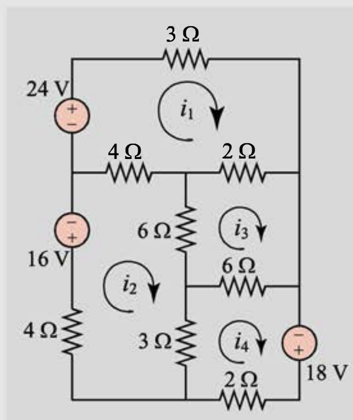
Using back substitution, the solution is:

$x_4 = \dfrac{0.5}{1} = 0.5$, $x_3 = -3.333 + 0.6667x_4 = -3$, $x_2 = 2.875 - 0.5x_3 - 0.75x_4 = 4$, and

$$x_1 = 3 + 0.5x_2 + 0.75x_3 - 1.5x_4 = 2$$

Example 4-6: MATLAB user-defined function for solving a system of equations using *LU* decomposition with Crout's method.

Determine the currents i_1, i_2, i_3, and i_4 in the circuit shown in the figure (same as in Fig. 4-1). Write the system of equations that has to be solved in the form $[a][i] = [b]$. Solve the system by using the *LU* decomposition method, and use Crout's method for doing the decomposition.



SOLUTION

The currents are determined from the set of four equations, Eq. (4.1). The equations are derived by using Kirchhoff's law. In matrix form, $[a][i] = [b]$, the equations are:

$$\begin{bmatrix} 9 & -4 & -2 & 0 \\ -4 & 17 & -6 & -3 \\ -2 & -6 & 14 & -6 \\ 0 & -3 & -6 & 11 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} 24 \\ -16 \\ 0 \\ 18 \end{bmatrix} \tag{4.44}$$

To solve the system of equations, three user-defined functions are created. The functions are as follows:

`[L U] = LUdecompCrout(A)` This function decomposes the matrix `A` into lower triangular and upper triangular matrices `L` and `U`, respectively.

`y = ForwardSub(L,b)` This function solves a system of equations that is given in lower triangular form.

`x = BackwardSub(L,b)` This function solves a system of equations that is given in upper triangular form.

Listing of the user-defined function `LUdecompCrout`:

Program 4-3: User-defined function. *LU* decomposition using Crout's method.

```
function [L, U] = LUdecompCrout(A)
% The function decomposes the matrix A into a lower triangular matrix L
% and an upper triangular matrix U, using Crout's method, such that A = LU.
% Input variables:
% A  The matrix of coefficients.
% b  Right-hand-side column vector of constants.
% Output variable:
% L  Lower triangular matrix.
% U  Upper triangular matrix.

[R, C]=size(A);
for i=1:R
   L(i,1)=A(i,1);                    Eq. (4.34).
   U(i,i)=1;                         Eq. (4.35).
end                                                    Steps 1 and 2 (page 122).
for j=2:R
   U(1,j)=A(1,j)/L(1,1);             Eq. (4.36).
end                                                    Step 3 (page 122).
for i=2:R
   for j=2:i
     L(i,j)=A(i,j)-L(i,1:j-1)*U(1:j-1,j);       Eq. (4.37).
   end
   for j=i + 1:R
     U(i,j)=(A(i,j)-L(i,1:i-1)*U(1:i-1,j))/L(i,i);   Eq. (4.38).
   end
end                                                    Step 4 (page 122).
```

Listing of the user-defined function `ForwardSub`:

Program 4-4: User-defined function. Forward substitution.

```
function y = ForwardSub(a,b)
% The function solves a system of linear equations ax = b
% where a is lower triangular matrix by using forward substitution.
% Input variables:
% a  The matrix of coefficients.
% b  A column vector of constants.
% Output variable:
```

```
% y  A column vector with the solution.

n=length(b);
y(1,1)=b(1)/a(1,1);
for i=2:n
    y(i,1)=(b(i)-a(i,1:i-1)*y(1:i-1,1))./a(i,i);
end
```

Eq. (4.8).

Listing of the user-defined function `BackwardSub`:

Program 4-5: User-defined function. Back substitution.

```
function y = BackwardSub(a,b)
% The function solves a system of linear equations ax = b
% where a is an upper triangular matrix by using back substitution.
% Input variables:
% a  The matrix of coefficients.
% b  A column vector of constants.
% Output variable:
% y  A column vector with the solution.

n=length(b);
y(n,1)=b(n)/a(n,n);
for i=n-1:-1:1
   y(i,1)=(b(i)-a(i,i+1:n)*y(i+1:n,1))./a(i,i);
end
```

Eq. (4.5).

The functions are then used in a MATLAB computer program (script file) that is used for solving the problem by following these steps:

- The matrix of coefficients $[a]$ is decomposed into upper $[U]$ and lower $[L]$ triangular matrices (using the `LUdecompCrout` function).
- The matrix $[L]$ and the vector $[b]$ are used in Eq. (4.23), $[L][y] = [b]$, to solve for $[y]$, (using the `ForwardSub` function).
- The solution $[y]$ and the matrix $[U]$ are used in Eq. (4.22), $[U][i] = [y]$, to solve for $[i]$ (using the `BackwardSub` function).

Script file:

Program 4-6: Script file. Solving a system with Crout's *LU* decomposition.

```
% This script file solves a system of equations by using
% the Crout's LU decomposition method.
a = [9 -4 -2 0; -4 17 -6 -3; -2 -6 14 -6; 0 -3 -6 11];
b = [24; -16; 0; 18];
[L, U]=LUdecompCrout(a);
y=ForwardSub(L,b);
i=BackwardSub(U,y)
```

When the script file is executed, the following solution is displayed in the Command Window:

```
i =
    4.0343
    1.6545
    2.8452
    3.6395
```

The script file can be easily modified for solving the systems of equations $[a][i] = [b]$ for the same matrix $[a]$, but different values of $[b]$. The LU decomposition is done once, and only the last two steps have to be executed for each $[b]$.

4.5.3 LU Decomposition with Pivoting

Decomposition of a matrix $[a]$ into the matrices $[L]$ and $[U]$ means that $[a] = [L][U]$. In the presentation of Gauss and Crout's decomposition methods in the previous two subsections, it is assumed that it is possible to carry out all the calculations without pivoting. In reality, as was discussed in Section 4.3, pivoting may be required for a successful execution of the Gauss elimination procedure. Pivoting might also be needed with Crout's method. If pivoting is used, then the matrices $[L]$ and $[U]$ that are obtained are not the decomposition of the original matrix $[a]$. The product $[L][U]$ gives a matrix with rows that have the same elements as $[a]$, but due to the pivoting, the rows are in a different order. When pivoting is used in the decomposition procedure, the changes that are made have to be recorded and stored. This is done by creating a matrix $[P]$, called a permutation matrix, such that:

$$[P][a] = [L][U] \tag{4.45}$$

If the matrices $[L]$ and $[U]$ are used for solving a system of equations $[a][x] = [b]$ (by using Eqs. (4.23) and (4.22)), then the order of the rows of $[b]$ have to be changed such that it is consistent with the pivoting. This is done by multiplying $[b]$ by the permutation matrix, $[P][b]$. Use of the permutation matrix is shown in Section 4.8.3, where the decomposition is done with MATLAB's built-in function.

4.6 INVERSE OF A MATRIX

The inverse of a square matrix $[a]$ is the matrix $[a]^{-1}$ such that the product of the two matrices gives the identity matrix $[I]$.

$$[a][a]^{-1} = [a]^{-1}[a] = [I] \tag{4.46}$$

The process of calculating the inverse of a matrix is essentially the same as the process of solving a system of linear equations. This is illustrated for the case of a (4×4) matrix. If $[a]$ is a given matrix and $[x]$ is the unknown inverse of $[a]$, then:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.47}$$

Equation (4.47) can be rewritten as four separate systems of equations, where in each system one column of the matrix $[x]$ is the unknown:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.48}$$

Solving the four systems of equations in Eqs. (4.48) gives the four columns of the inverse of $[a]$. The systems of equations can be solved by using any of the methods that have been introduced earlier in this chapter (or other methods). Two of the methods, the *LU* decomposition method and the Gauss–Jordan elimination method, are described in more detail next.

4.6.1 Calculating the Inverse with the LU Decomposition Method

The *LU* decomposition method is especially suitable for calculating the inverse of a matrix. As shown in Eqs. (4.48), the matrix of coefficients in all four systems of equations is the same. Consequently, the *LU* decomposition of the matrix $[A]$ is calculated only once. Then, each of the systems is solved by first using Eq. (4.23) (forward substitution) and then Eq. (4.22) (back substitution). This is illustrated, by using MATLAB, in Example 4-7.

Example 4-7: Determining the inverse of a matrix using the *LU* decomposition method.

Determine the inverse of the matrix $[a]$ by using the LU decomposition method.

$$[a] = \begin{bmatrix} 0.2 & -5 & 3 & 0.4 & 0 \\ -0.5 & 1 & 7 & -2 & 0.3 \\ 0.6 & 2 & -4 & 3 & 0.1 \\ 3 & 0.8 & 2 & -0.4 & 3 \\ 0.5 & 3 & 2 & 0.4 & 1 \end{bmatrix} \tag{4.49}$$

Do the calculations by writing a MATLAB user-defined function. Name the function `invA = InverseLU(A)`, where `A` is the matrix to be inverted, and `invA` is the inverse. In the function, use the functions `LUdecompCrout`, `ForwardSub`, and `BackwardSub` that were written in Example 4-6.

SOLUTION

If the inverse of $[a]$ is $[x]$ ($[x] = [a]^{-1}$), then $[a][x] = [I]$, which are the following five sets of five systems of equations that have to be solved. In each set of equations, one column of the inverse is calculated.

$$\begin{bmatrix} 0.2 & -5 & 3 & 0.4 & 0 \\ -0.5 & 1 & 7 & -2 & 0.3 \\ 0.6 & 2 & -4 & 3 & 0.1 \\ 3 & 0.8 & 2 & -0.4 & 3 \\ 0.5 & 3 & 2 & 0.4 & 1 \end{bmatrix}\begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \\ x_{51} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0.2 & -5 & 3 & 0.4 & 0 \\ -0.5 & 1 & 7 & -2 & 0.3 \\ 0.6 & 2 & -4 & 3 & 0.1 \\ 3 & 0.8 & 2 & -0.4 & 3 \\ 0.5 & 3 & 2 & 0.4 & 1 \end{bmatrix}\begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \\ x_{42} \\ x_{52} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0.2 & -5 & 3 & 0.4 & 0 \\ -0.5 & 1 & 7 & -2 & 0.3 \\ 0.6 & 2 & -4 & 3 & 0.1 \\ 3 & 0.8 & 2 & -0.4 & 3 \\ 0.5 & 3 & 2 & 0.4 & 1 \end{bmatrix}\begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \\ x_{53} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0.2 & -5 & 3 & 0.4 & 0 \\ -0.5 & 1 & 7 & -2 & 0.3 \\ 0.6 & 2 & -4 & 3 & 0.1 \\ 3 & 0.8 & 2 & -0.4 & 3 \\ 0.5 & 3 & 2 & 0.4 & 1 \end{bmatrix}\begin{bmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \\ x_{54} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0.2 & -5 & 3 & 0.4 & 0 \\ -0.5 & 1 & 7 & -2 & 0.3 \\ 0.6 & 2 & -4 & 3 & 0.1 \\ 3 & 0.8 & 2 & -0.4 & 3 \\ 0.5 & 3 & 2 & 0.4 & 1 \end{bmatrix}\begin{bmatrix} x_{15} \\ x_{25} \\ x_{35} \\ x_{45} \\ x_{55} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.50}$$

The solution is obtained with the user-defined function `InverseLU` that is listed below. The function can be used for calculating the inverse of any sized square matrix.

The function executes the following operations:

- The matrix $[a]$ is decomposed into matrices $[L]$ and $[U]$ by applying Crout's method. This is done by using the function `LUdecompCrout` that was written in Example 4-6.
- Each system of equations in Eqs. (4.50) is solved by using Eqs. (4.23) and (4.22). This is done by first using the function `ForwardSub` and subsequently the function `BackwardSub` (see Example 4-6).

Program 4-7: User-defined function. Inverse of a matrix.

```
function invA = InverseLU(A)
% The function calculates the inverse of a matrix
% Input variables:
% A  The matrix to be inverted.
% Output variable:
% invA  The inverse of A.
```

```
[nR nC] = size(A);
I=eye(nR);                          Create an identity matrix of the same size as [A].
[L U]= LUdecompCrout(A);            Decomposition of [A] into [L] and [U].
for j=1:nC
    y=ForwardSub(L,I(:,j));
    invA(:,j)=BackwardSub(U,y);
end
```

In each pass of the loop, one set of the equations in Eqs. (4.50) is solved. Each solution is one column in the inverse of the matrix.

The function is then used in the Command Window for solving the problem.

```
>> F=[0.2 -5 3 0.4 0; -0.5 1 7 -2 0.3; 0.6 2 -4 3 0.1; 3 0.8 2 -0.4 3; 0.5 3 2 0.4 1];
>> invF = InverseLU(F)
invF =
  -0.7079    2.5314    2.4312    0.9666   -3.9023
  -0.1934    0.3101    0.2795    0.0577   -0.2941
   0.0217    0.3655    0.2861    0.0506   -0.2899
   0.2734   -0.1299    0.1316   -0.1410    0.4489
   0.7815   -2.8751   -2.6789   -0.7011    4.2338
>> invF*F
ans =
   1.0000   -0.0000    0.0000   -0.0000   -0.0000
   0.0000    1.0000    0.0000   -0.0          0
        0   -0.0000    1.0000   -0.0000   -0.0000
  -0.0000    0.0000   -0.0000    1.0000   -0.0000
  -0.0000    0.0000   -0.0000   -0.0000    1.0000
```

The solution $[F]^{-1}$.

Check if $[F][F]^{-1} = [I]$.

4.6.2 Calculating the Inverse Using the Gauss–Jordan Method

The Gauss–Jordan method is easily adapted for calculating the inverse of a square $(n \times n)$ matrix $[a]$. This is done by first appending an identity matrix $[I]$ of the same size as the matrix $[a]$ to $[a]$ itself. This is shown schematically for a (4×4) matrix in Fig. 4-20*a*. Then, the Gauss–Jordan procedure is applied such that the elements of the matrix $[a]$ (the left half of the augmented matrix) are converted to 1s along the diagonal and 0s elsewhere. During this process, the terms of the identity matrix in Fig. 4-20*a* (the right half of the augmented matrix) are changed and become the elements $[a']$ in Fig. 4-20*b*, which constitute the inverse of $[a]$.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & 1 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 1 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 & 0 & 1 & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} & 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{Gauss–Jordan procedure}} \begin{bmatrix} 1 & 0 & 0 & 0 & a'_{11} & a'_{12} & a'_{13} & a'_{14} \\ 0 & 1 & 0 & 0 & a'_{21} & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & 1 & 0 & a'_{31} & a'_{32} & a'_{33} & a'_{34} \\ 0 & 0 & 0 & 1 & a'_{41} & a'_{42} & a'_{43} & a'_{44} \end{bmatrix}$$

(*a*) (*b*)

Figure 4-20: Calculating the inverse with the Gauss–Jordan method.

4.7 ITERATIVE METHODS

A system of linear equations can also be solved by using an iterative approach. The process, in principle, is the same as in the fixed-point iteration method used for solving a single nonlinear equation (see Section 3.7). In an iterative process for solving a system of equations, the equations are written in an explicit form in which each unknown is written in terms of the other unknown. The explicit form for a system of four equations is illustrated in Fig. 4-21.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= b_4 \end{aligned}$$

(*a*)

Writing the equations in an explicit form.

$$\begin{aligned} x_1 &= [b_1 - (a_{12}x_2 + a_{13}x_3 + a_{14}x_4)]/a_{11} \\ x_2 &= [b_2 - (a_{21}x_1 + a_{23}x_3 + a_{24}x_4)]/a_{22} \\ x_3 &= [b_3 - (a_{31}x_1 + a_{32}x_2 + a_{34}x_4)]/a_{33} \\ x_4 &= [b_4 - (a_{21}x_1 + a_{42}x_2 + a_{43}x_3)]/a_{44} \end{aligned}$$

(*b*)

Figure 4-21: Standard (*a*) and explicit (*b*) forms of a system of four equations.

The solution process starts by assuming initial values for the unknowns (first estimated solution). In the first iteration, the first assumed solution is substituted on the right-hand side of the equations, and the new values that are calculated for the unknowns are the second estimated solution. In the second iteration, the second solution is substituted back in the equations to give new values for the unknowns, which are the third estimated solution. The iterations continue in the same manner, and when the method does work, the solutions that are obtained as successive iterations converge toward the actual solution. For a system with n equations, the explicit equations for the $[x_i]$ unknowns are:

$$x_i = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1,\, j \neq i}^{j=n} a_{ij}x_j\right] \qquad i = 1, 2, \ldots, n \tag{4.51}$$

Condition for convergence

For a system of n equations $[a][x] = [b]$, a sufficient condition for convergence is that in each row of the matrix of coefficients $[a]$ the absolute value of the diagonal element is greater than the sum of the absolute values of the off-diagonal elements.

$$|a_{ii}| > \sum_{j=1,\, j \neq i}^{j=n} |a_{ij}| \tag{4.52}$$

This condition is sufficient but not necessary for convergence when the iteration method is used. When condition (4.52) is satisfied, the matrix $[a]$ is classified as ***diagonally dominant***, and the iteration process converges toward the solution. The solution, however, might converge even when Eq. (4.52) is not satisfied.

Two specific iterative methods for executing the iterations, the Jacobi and Gauss–Seidel methods, are presented next. The difference between the two methods is in the way that the new calculated values of the unknowns are used. In the Jacobi method, the estimated values of the unknowns that are used on the right-hand side of Eq. (4.51) are updated all at once at the end of each iteration. In the Gauss–Seidel method, the value of each unknown is updated (and used in the calculation of the new estimate of the rest of the unknowns in the same iteration) when a new estimate for this unknown is calculated.

4.7.1 Jacobi Iterative Method

In the Jacobi method, an initial (first) value is assumed for each of the unknowns, $x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}$. If no information is available regarding the approximate values of the unknown, the initial value of all the unknowns can be assumed to be zero. The second estimate of the solution $x_1^{(2)}, x_2^{(2)}, \ldots, x_n^{(2)}$ is calculated by substituting the first estimate in the right-hand side of Eqs. (4.51):

$$x_i^{(2)} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1,\, j \neq i}^{j=n} a_{ij}x_j^{(1)}\right] \qquad i = 1, 2, \ldots, n \tag{4.53}$$

In general, the $(k+1)$th estimate of the solution is calculated from the (k)th estimate by:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1,\, j \neq i}^{j=n} a_{ij}x_j^{(k)}\right] \qquad i = 1, 2, \ldots, n \tag{4.54}$$

The iterations continue until the differences between the values that are obtained in successive iterations are small. The iterations can be stopped when the absolute value of the estimated relative error (see Section 3.2) of all the unknowns is smaller than some predetermined value:

$$\left|\frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k)}}\right| < \varepsilon \qquad i = 1, 2, \ldots, n \tag{4.55}$$

4.7.2 Gauss-Seidel Iterative Method

In the Gauss–Seidel method, initial (first) values are assumed for the unknowns $x_2, x_3, \ldots, x_n$ (all of the unknowns except x_1). If no information is available regarding the approximate value of the unknowns, the initial value of all the unknowns can be assumed to be zero. The first assumed values of the unknowns are substituted in Eq. (4.51) with $i = 1$ to calculate the value of x_1. Next, Eq. (4.51) with $i = 2$ is used for calculating a new value for x_2. This is followed by using Eq. (4.51) with $i = 3$ for calculating a new value for x_3. The process continues until $i = n$, which is the end of the first iteration. Then, the second iteration starts with $i = 1$ where a new value for x_1 is calculated, and so on. In the Gauss–Seidel method, the current values of the unknowns are used for calculating the new value of the next unknown. In other words, as a new value of an unknown is calculated, it is immediately used for the next application of Eq. (4.51). (In the Jacobi method, the values of the unknowns obtained in one iteration are used as a complete set for calculating the new values of the unknowns in the next iteration. The values of the unknowns are not updated in the middle of the iteration.)

Applying Eq. (4.51) to the Gauss–Seidel method gives the iteration formula:

$$x_1^{(k+1)} = \frac{1}{a_{11}}\left[b_1 - \sum_{j=2}^{j=n} a_{1j}x_j^{(k)}\right]$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left[b_i - \left(\sum_{j=1}^{j=i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{j=n} a_{ij}x_j^{(k)}\right)\right] \quad i = 2, 3, \ldots, n-1 \tag{4.56}$$

$$x_n^{(k+1)} = \frac{1}{a_{nn}}\left[b_n - \sum_{j=1}^{j=n-1} a_{nj}x_j^{(k+1)}\right]$$

Notice that the values of the unknowns in the $k+1$ iteration, $x_i^{(k+1)}$, are calculated by using the values $x_j^{(k+1)}$ obtained in the $k+1$ iteration for $j < i$ and using the values $x_j^{(k)}$ for $j > i$. The criterion for stopping the iterations is the same as in the Jacobi method, Eq. (4.55). The Gauss–Seidel method converges faster than the Jacobi method and requires less computer memory when programmed. The method is illustrated for a system of four equations in Example 4-8.

Example 4-8: Solving a set of four linear equations using Gauss–Seidel method.

Solve the following set of four linear equations using the Gauss–Seidel iteration method.

$$\begin{aligned} 9x_1 - 2x_2 + 3x_3 + 2x_4 &= 54.5 \\ 2x_1 + 8x_2 - 2x_3 + 3x_4 &= -14 \\ -3x_1 + 2x_2 + 11x_3 - 4x_4 &= 12.5 \\ -2x_1 + 3x_2 + 2x_3 + 10x_4 &= -21 \end{aligned}$$

SOLUTION

First, the equations are written in an explicit form (see Fig. 4.21):

$$\begin{aligned} x_1 &= [54.5 - (-2x_2 + 3x_3 + 2x_4)]/9 \\ x_2 &= [-14 - (2x_1 - 2x_3 + 3x_4)]/8 \\ x_3 &= [12.5 - (-3x_1 + 2x_2 - 4x_4)]/11 \\ x_4 &= [-21 - (-2x_1 + 3x_2 + 2x_3)]/10 \end{aligned} \tag{4.57}$$

As a starting point, the initial value of all the unknowns, $x_1^{(1)}, x_2^{(1)}, x_3^{(1)}$, and $x_4^{(1)}$, is assumed to be zero. The first two iterations are calculated manually, and then a MATLAB program is used for calculating the values of the unknowns in seven iterations.

Manual calculation of the first two iterations:

The second estimate of the solution ($k = 2$) is calculated in the first iteration by using Eqs. (4.57). The values that are substituted for x_i in the right-hand side of the equations are the most recent known values. This means that when the first equation is used to calculate $x_1^{(2)}$, all the x_i values are zero. Then, when the second equation is used to calculate $x_2^{(2)}$, the new value $x_1^{(2)}$ is substituted for x_1, but the older values $x_3^{(1)}$ and $x_4^{(1)}$ are substituted for x_3 and x_4, and so on:

$$x_1^{(2)} = [54.5 - (-2 \cdot 0 + 3 \cdot 0 + 2 \cdot 0)]/9 = 6.056$$

$$x_2^{(2)} = [-14 - (2 \cdot 6.056 - (2 \cdot 0) + 3 \cdot 0)]/8 = -3.264$$

$$x_3^{(2)} = [12.5 - (-3 \cdot 6.056 + 2 \cdot -3.264 - (4 \cdot 0))]/11 = 3.381$$

$$x_4^{(2)} = [-21 - (-2 \cdot 6.056 + 3 \cdot -3.264 + 2 \cdot 3.381)]/10 = -0.5860$$

The third estimate of the solution ($k = 3$) is calculated in the second iteration:

$$x_1^{(3)} = [54.5 - (-2 \cdot -3.264 + 3 \cdot 3.381 + 2 \cdot -0.5860)]/9 = 4.333$$

$$x_2^{(3)} = [-14 - (2 \cdot 4.333 - (2 \cdot 3.381) + 3 \cdot -0.5860)]/8 = -1.768$$

$$x_3^{(3)} = [12.5 - (-3 \cdot 4.333 + 2 \cdot -1.768 - (4 \cdot -0.5860))]/11 = 2.427$$

$$x_4^{(3)} = [-21 - (-2 \cdot 4.333 + 3 \cdot -1.768 + 2 \cdot 2.427)]/10 = -1.188$$

MATLAB program that calculates the first seven iterations:

The following is a MATLAB program in a script file that calculates the first seven iterations of the solution by using Eqs. (4.57):

Program 4-8: Script file. Gauss–Seidel iteration.

```
k = 1; x1 = 0; x2 = 0; x3 = 0; x4 = 0;
disp('  k          x1         x2         x3          x4')
fprintf(' %2.0f      %-8.5f  %-8.5f  %-8.5f  %-8.5f \n', k, x1, x2, x3, x4)
for k = 2 : 8
   x1=(54.5-(-2*x2+3*x3+2*x4))/9;
   x2=(-14-(2*x1-2*x3+3*x4))/8;
   x3=(12.5-(-3*x1+2*x2-4*x4))/11;
   x4=(-21-(-2*x1+3*x2+2*x3))/10;
   fprintf(' %2.0f     %-8.5f  %-8.5f  %-8.5f  %-8.5f \n', k, x1, x2, x3, x4)
end
```

When the program is executed, the following results are displayed in the Command Window.

```
  k         x1         x2        x3         x4
  1       0.00000   0.00000   0.00000    0.00000
  2       6.05556  -3.26389   3.38131   -0.58598
  3       4.33336  -1.76827   2.42661   -1.18817
```

```
4       5.11778  -1.97723  2.45956  -0.97519
5       5.01303  -2.02267  2.51670  -0.99393
6       4.98805  -1.99511  2.49806  -1.00347
7       5.00250  -1.99981  2.49939  -0.99943
8       5.00012  -2.00040  2.50031  -0.99992
```

The results show that the solution converges toward the exact solution, which is $x_1 = 5$, $x_2 = -2$, $x_3 = 2.5$, and $x_4 = -1$.

4.8 USE OF MATLAB BUILT-IN FUNCTIONS FOR SOLVING A SYSTEM OF LINEAR EQUATIONS

MATLAB has mathematical operations and built-in functions that can be used for solving a system of linear equations and for carrying out other matrix operations that are described in this chapter.

4.8.1 Solving a System of Equations Using MATLAB's Left and Right Division

***Left division* \ :** Left division can be used to solve a system of n equations written in matrix form $[a][x] = [b]$, where $[a]$ is the $(n \times n)$ matrix of coefficients, $[x]$ is an $(n \times 1)$ column vector of the unknowns, and $[b]$ is an $(n \times 1)$ column vector of constants.

```
x = a\b
```

For example, the solution of the system of equations in Examples 4-1 and 4-2 is calculated by (Command Window):

```
>> a=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
>> b=[12; -6.5; 16; 17];
>> x=a\b
x =
    2.0000
    4.0000
   -3.0000
    0.5000
```

***Right division* / :** Right division is used to solve a system of n equations written in matrix form $[x][a] = [b]$, where $[a]$ is the $(n \times n)$ matrix of coefficients, $[x]$ is a $(1 \times n)$ row vector of the unknowns, and $[b]$ is a $(1 \times n)$ row vector of constants.

```
x = b/a
```

For example, the solution of the system of equations in Examples 4-1 and 4-2 is calculated by (Command Window):

```
>> a=[4 -6 1 -12; -2 7 7.5 22; -3 6.5 6.25 15.5; 6 -6 5.5 -1];
>> b=[12 -6.5 16 17];
>> x=b/a
x =
   2.0000    4.0000   -3.0000    0.5000
```

Notice that the matrix $[a]$ used in the right division calculation is the transpose of the matrix used in the left division calculation.

4.8.2 Solving a System of Equations Using MATLAB's Inverse Operation

In matrix form, the system of equations $[a][x]=[b]$ can be solved for $[x]$. Multiplying both sides from the left by $[a]^{-1}$ (the inverse of $[a]$) gives:

$$[a]^{-1}[a][x] = [a]^{-1}[b] \tag{4.58}$$

Since $[a]^{-1}[a] = [I]$ (identity matrix), and $[I][x] = [x]$, Eq. (4.58) reduces to:

$$[x] = [a]^{-1}[b] \tag{4.59}$$

In MATLAB, the inverse of a matrix $[a]$ can be calculated either by raising the matrix to the power of –1 or by using the `inv(a)` function. Once the inverse is calculated, the solution is obtained by multiplying the vector $[b]$ by the inverse. This is demonstrated for the solution of the system in Examples 4-1 and 4-2.

```
>> a=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
>> b=[12; -6.5; 16; 17];
>> x=a^-1*b
x =
   2.0000
   4.0000
  -3.0000
   0.5000
```

The same result is obtained by typing >> x = inv(a)*b.

4.8.3 MATLAB's Built-In Function for LU Decomposition

MATLAB has a built-in function, called `lu`, that decomposes a matrix $[a]$ into the product $[L][U]$, such that $[a] = [L][U]$ where $[L]$ is a lower triangular matrix and $[U]$ is an upper triangular matrix. One form of the function is:

```
[L, U, P] = lu(a)
```

`L` is a lower triangular matrix.
`U` is an upper triangular matrix.
`P` is a permutation matrix.

`a` is the matrix to be decomposed.

MATLAB uses partial pivoting when determining the factorization. Consequently, the matrices $[L]$ and $[U]$ that are determined by MATLAB are the factorization of a matrix with rows that may be in a different order than in $[a]$. The permutation matrix $[P]$ (a matrix with 1s and 0s) contains the information about the pivoting. Multiplying $[a]$ by the matrix $[P]$ gives the matrix whose decomposition is given by $[L]$ and $[U]$ (see Section 4.5.3):

$$[L][U] = [P][a] \tag{4.60}$$

The matrix $[P][a]$ has the same rows as $[a]$ but in a different order. If MATLAB does not use partial pivoting when the function `lu` is used, then the permutation matrix $[P]$ is the identity matrix.

If the matrices $[L]$ and $[U]$ that are determined by the `lu` function are subsequently used for solving a system of equations $[a][x] = [b]$ (by using Eqs. (4.23) and (4.22)), then the vector $[b]$ has to be multiplied by the permutation matrix $[P]$. This pivots the rows in $[b]$ to be consistent with the pivoting in $[a]$. The following shows a MATLAB solution of the system of equations from Examples 4-1 and 4-2 using the function `lu`.

```
>> a=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
>> b=[12; -6.5; 16; 17];
>> [L, U, P]=lu(a)
L =
    1.0000         0         0         0
   -0.0833    1.0000         0         0
   -0.3333    0.5714    1.0000         0
    0.5000   -0.4286   -0.9250    1.0000
U =
  -12.0000   22.0000   15.5000   -1.0000
         0    9.3333    7.5417    5.4167
         0         0   -2.1429    2.5714
         0         0         0   -0.800
```

Decomposition of $[a]$ using MATLAB's `lu` function.

```
P =
   0     0     0     1
   0     0     1     0
   1     0     0     0
   0     1     0     0
>> y=L\(P*b)
```

Multiplying $[P][a]$ gives the pivoted $[a]$

```
>> P*a
ans =
  -12.0000   22.0000   15.5000   -1.0000
    1.0000    7.5000    6.2500    5.5000
    4.0000   -2.0000   -3.0000    6.0000
   -6.0000    7.0000    6.5000   -6.0000
```

Solve for y in Eq. (4.23).

Vector $[b]$ is multiplied by the permutation matrix.

```
y =
   17.0000
   17.4167
    7.7143
   -0.4000
>> x=U\y
x =
    2.0000
    4.0000
   -3.0000
    0.5000
```

Solve for x in Eq. (4.22).

4.8.4 Additional MATLAB Built-In Functions

MATLAB has many built-in functions that can be useful in the analysis of systems of equations. Several of these functions are presented in Table 4-1. Note that the operations that are related to some of the functions in the table are discussed in Section 4.9.

Table 4-1: Built-in MATLAB functions for matrix operations and analysis.

Function	Description	Example

`inv(A)`	Inverse of a matrix.
	`A` is a square matrix. Returns the inverse of `A`.
		`>> A=[-3 1 0.6; 0.2 -4 3; 0.1 0.5 2];`
		`>> Ain=inv(A)`
		`Ain =`
		`  -0.3310   -0.0592    0.1882`
		`  -0.0035   -0.2111    0.3178`
		`   0.0174    0.0557    0.4111`

Table 4-1: Built-in MATLAB functions for matrix operations and analysis. (Continued)

Function	Description	Example

`d=det(A)`	Determinant of a matrix
	`A` is a square matrix, `d` is the determinant of `A`.
		`>> A=[-3 1 0.6; 0.2 -4 3; 0.1 0.5 2];`
		`>> d=det(A)`
		`d =`
		`28.7000`

`n=norm(A)`
 	
`n=norm(A,p)`
	Vector and matrix norm
	`A` is a vector or a matrix, `n` is its norm.
	***When A is a vector*:**
	`norm(A,p)` returns:
	`sum(abs(A.^p)^(1/p).`
	`p=inf` The infinity norm (see Eq. (4.70)).
	`norm(A)` Returns the Euclidean 2-norm (see Eq. (4.72)), same as `norm(A,2)`.
		`>> A=[2 0 7 -9];`
		`>> n=norm(A,1)`
		`n =`
		`18`
		`>> n=norm(A,inf)`
		`n =`
		`9`
		`>> n=norm(A,2)`
		`n =`
		`11.5758`

	***When A is a matrix*:**
	`norm(A,p)` returns:
	`p=1` The 1-norm (largest column sum of `A` (see Eq. (4.74)).
	`p=2` The largest singular value, same as `norm(A)` (see Eq. (4.75)). This is not the Euclidean norm (see Eq. (4.76)).
	`p=inf` The infinity norm (see Eq. (4.73)).
		`>> A=[1 3 -2; 0 -1 4; 5 2 3];`
		`>> n=norm(A,1)`
		`n =`
		`9`
		`>> n=norm(A,2)`
		`n =`
		`6.4818`
		`>> n=norm(A,inf)`
		`n =`
		`10`

`c=cond(A)`
 	
`c=cond(A,p)`
	Condition number (see Eq. (4.86))
	`A` is a square matrix, `c` is the condition number of `A`.
	`cond(A)` The same as `p=2`.
	`p=1` The 1-norm condition number.
	`p=2` The 2-norm condition number.
	`p=inf` The infinity norm condition number.
		`>> a=[9 -2 3 2; 2 8 -2 3; -3 2 11 -4; -2 3 2 10];`
		`>> cond(a,inf)`
		`ans =`
		`3.8039`
		See the end of Example 4-10.

4.9 TRIDIAGONAL SYSTEMS OF EQUATIONS

Tridiagonal systems of linear equations have a matrix of coefficients with zero as their entries except along the diagonal, above-diagonal, and below-diagonal elements. A tridiagonal system of n equations in matrix form is shown in Eq. (4.61) and is illustrated for a system of five equations in Fig. 4-22.

$$\begin{bmatrix} A_{11} & A_{12} & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 \\ 0 & A_{32} & A_{33} & A_{34} & 0 \\ 0 & 0 & A_{43} & A_{44} & A_{45} \\ 0 & 0 & 0 & A_{54} & A_{55} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \end{bmatrix}$$

Figure 4-22: Tridiagonal system of five equations.

$$\begin{bmatrix} A_{11} & A_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 & 0 & 0 & 0 \\ 0 & A_{32} & A_{33} & A_{34} & 0 & 0 & 0 & 0 \\ & & \cdots & \cdots & \cdots & & & \\ & & & \cdots & \cdots & \cdots & & \\ 0 & 0 & 0 & 0 & A_{n-2,n-3} & A_{n-2,n-2} & A_{n-2,n-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & A_{n-1,n-2} & A_{n-1,n-1} & A_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & 0 & A_{n,n-1} & A_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ \cdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \cdots \\ \cdots \\ B_{n-2} \\ B_{n-1} \\ B_n \end{bmatrix} \tag{4.61}$$

The matrix of coefficients of tridiagonal systems has many elements that are zero (especially when the system contains a large number of equations). The system can be solved with the standard methods (Gauss, Gauss–Jordan, *LU* decomposition), but then a large number of zero elements are stored and a large number of needless operations (with zeros) are executed. To save computer memory and computing time, special numerical methods have been developed for solving tridiagonal systems of equations. One of these methods, the Thomas algorithm, is described in this section.

Many applications in engineering and science require the solution of tridiagonal systems of equations. Some numerical methods for solving differential equations also involve the solution of such systems.

Thomas algorithm for solving tridiagonal systems

The Thomas algorithm is a procedure for solving tridiagonal systems of equations. The method of solution in the Thomas algorithm is similar to the Gaussian elimination method in which the system is first changed to upper triangular form and then solved using back substitution. The Thomas algorithm, however, is much more efficient because only the nonzero elements of the matrix of coefficients are stored, and only the necessary operations are executed. (Unnecessary operations on the zero elements are eliminated.)

The Thomas algorithm starts by assigning the nonzero elements of the tridiagonal matrix of coefficients $[A]$ to three vectors. The diagonal elements A_{ii} are assigned to vector d (d stands for diagonal) such that $d_i = A_{ii}$. The above diagonal elements $A_{i,\,i+1}$ are assigned to vector a (a stands for above diagonal) such that $a_i = A_{i,\,i+1}$, and the below diagonal elements $A_{i-1,\,i}$ are assigned to vector b (b stands for below diagonal), such that $b_i = A_{i-1,\,i}$. With the nonzero elements in the matrix of coefficients stored as vectors, the system of equations has the form:

$$\begin{bmatrix} d_1 & a_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_2 & d_2 & a_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & b_3 & d_3 & a_3 & 0 & 0 & 0 & 0 \\ & & \cdots & \cdots & \cdots & & & \\ & & & \cdots & \cdots & \cdots & & \\ 0 & 0 & 0 & 0 & b_{n-2} & d_{n-2} & a_{n-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{n-1} & d_{n-1} & a_{n-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & b_n & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ \cdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \cdots \\ \cdots \\ B_{n-2} \\ B_{n-1} \\ B_n \end{bmatrix} \tag{4.62}$$

It should be emphasized here that in Eq. (4.62) the matrix of coefficients is displayed as a matrix (with the 0s), but in the Thomas algorithm only the vectors b, d, and a are stored.

Next, the first row is normalized by dividing the row by d_1. This makes the element d_1 (to be used as the pivot element) equal to 1:

$$\begin{bmatrix} 1 & a'_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_2 & d_2 & a_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & b_3 & d_3 & a_3 & 0 & 0 & 0 & 0 \\ & & \cdots & \cdots & \cdots & & & \\ & & & \cdots & \cdots & \cdots & & \\ 0 & 0 & 0 & 0 & b_{n-2} & d_{n-2} & a_{n-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{n-1} & d_{n-1} & a_{n-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & b_n & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ \cdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} B'_1 \\ B_2 \\ B_3 \\ \cdots \\ \cdots \\ B_{n-2} \\ B_{n-1} \\ B_n \end{bmatrix} \tag{4.63}$$

where $a'_1 = a_1/d_1$ and $B'_1 = B_1/d_1$.

Now the element b_2 is eliminated. The first row (the pivot row) is multiplied by b_2 and then is subtracted from the second row:

$$\begin{bmatrix} 1 & a'_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & d'_2 & a_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & b_3 & d_3 & a_3 & 0 & 0 & 0 & 0 \\ & & \cdots & \cdots & \cdots & & & \\ & & & \cdots & \cdots & \cdots & & \\ 0 & 0 & 0 & 0 & b_{n-2} & d_{n-2} & a_{n-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{n-1} & d_{n-1} & a_{n-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & b_n & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ \cdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} B'_1 \\ B'_2 \\ B_3 \\ \cdots \\ \cdots \\ B_{n-2} \\ B_{n-1} \\ B_n \end{bmatrix} \tag{4.64}$$

where $d'_2 = d_2 - b_2 a'_1$, and $B'_2 = B_2 - B_1 b_2$.

The operations performed with the first and second row are repeated with the second and third rows. The second row is normalized by dividing the row by d'_2. This makes the element d'_2 (to be used as the pivot element) equal to 1. The second row is then used to eliminate b_3 in the third row.

This process continues row after row until the system of equations is transformed to be upper triangular with 1s along the diagonal:

$$\begin{bmatrix} 1 & a'_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & a'_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & a'_3 & 0 & 0 & 0 & 0 \\ & & \cdots & \cdots & \cdots & & & \\ & & & \cdots & \cdots & \cdots & & \\ 0 & 0 & 0 & 0 & 0 & 1 & a'_{n-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & a'_{n-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ \cdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} B'_1 \\ B''_2 \\ B''_3 \\ \cdots \\ \cdots \\ B''_{n-2} \\ B''_{n-1} \\ B''_n \end{bmatrix} \tag{4.65}$$

Once the matrix of coefficients is in upper triangular form, the values of the unknowns are calculated by using back substitution.

In mathematical form, the Thomas algorithm can be summarized in the following steps:

Step 1: Define the vectors $b = [0, b_2, b_3, \ldots, b_n]$, $d = [d_1, d_2, \ldots, d_n]$, $a = [a_1, a_2, \ldots, a_{n-1}]$, and $B = [B_1, B_2, \ldots, B_n]$.

Step 2: Calculate: $a_1 = \dfrac{a_1}{d_1}$ and $B_1 = \dfrac{B_1}{d_1}$.

Step 3: For $i = 2, 3, \ldots, n-1$, calculate:

$$a_i = \frac{a_i}{d_i - b_i a_{i-1}} \text{ and } B_i = \frac{B_i - b_i B_{i-1}}{d_i - b_i a_{i-1}}$$

Step 4: Calculate: $B_n = \dfrac{B_n - b_n B_{n-1}}{d_n - b_n a_{n-1}}$

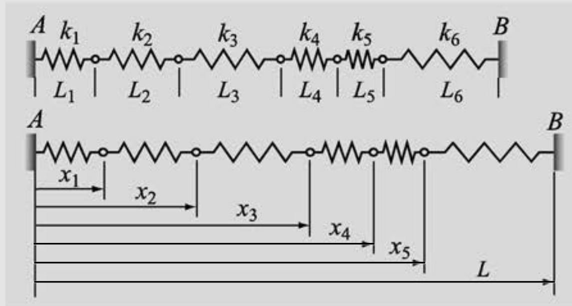
Step 5: Calculate the solution using back substitution:

$x_n = B_n$ and for $i = n-1, n-2, n-3, \ldots, 2, 1$, $x_i = B_i - a_i x_{i+1}$

A solution of a tridiagonal system of equations, using a user-defined MATLAB function, is shown in Example 4-9.

Example 4-9: Solving a tridiagonal system of equations using the Thomas algorithm.

Six springs with different spring constants k_i and unstretched lengths L_i are attached to each other in series. The endpoint B is then displaced such that the distance between points A and B is $L = 1.5$ m. Determine the positions $x_1, x_2, \ldots, x_5$ of the endpoints of the springs.



The spring constants and the unstretched lengths of the springs are:

spring	1	2	3	4	5	6
k (kN/m)	8	9	15	12	10	18
L (m)	0.18	0.22	0.26	0.19	0.15	0.30

SOLUTION

The force, F, in a spring is given by:

$$F = k\delta$$

where k is the spring constant and δ is the extension of the spring beyond its unstretched length. Since the springs are connected in series, the force in all of the springs is the same. Consequently, it is possible to write five equations that equate the force in every two adjacent springs. For example, the condition that the force in the first spring is equal to the force in the second spring gives:

$$k_1(x_1 - L_1) = k_2[(x_2 - x_1) - L_2]$$

Similarly, four additional equations can be written:

$$k_2[(x_2 - x_1) - L_2] = k_3[(x_3 - x_2) - L_3]$$

$$k_3[(x_3 - x_2) - L_3] = k_4[(x_4 - x_3) - L_4]$$

$$k_4[(x_4 - x_3) - L_4] = k_5[(x_5 - x_4) - L_5]$$

$$k_5[(x_5 - x_4) - L_5] = k_6[(L - x_5) - L_6]$$

The five equations form a system that is tridiagonal. In matrix form the system is:

$$\begin{bmatrix} k_1+k_2 & -k_2 & 0 & 0 & 0 \\ -k_2 & k_2+k_3 & -k_3 & 0 & 0 \\ 0 & -k_3 & k_3+k_4 & -k_4 & 0 \\ 0 & 0 & -k_4 & k_4+k_5 & -k_5 \\ 0 & 0 & 0 & -k_5 & k_5+k_6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} k_1L_1 - k_2L_2 \\ k_2L_2 - k_3L_3 \\ k_3L_3 - k_4L_4 \\ k_4L_4 - k_5L_5 \\ k_5L_5 + k_6L - k_6L_6 \end{bmatrix} \tag{4.66}$$

The system of equations (4.66) is solved with a user-defined MATLAB function `Tridiagonal`, which is listed next.

Program 4-9: User-defined function. Solving a tridiagonal system of equations.

```
function x = Tridiagonal(A,B)
% The function solves a tridiagonal system of linear equations [a][x]=[b]
% using the Thomas algorithm.
% Input variables:
```

```
% A   The matrix of coefficients.
% B   Right-hand-side column vector of constants.
% Output variable:
% x   A column vector with the solution.

[nR, nC]=size(A);
for i=1:nR
   d(i)=A(i,i);            Define the vector d with the elements of the diagonal.
end
for i=1:nR-1
   ad(i)=A(i,i+1);         Define the vector ad with the above diagonal elements.     Step 1.
end
for i=2:nR
   bd(i)=A(i,i-1);         Define the vector bd with the below diagonal elements.
end
ad(1)=ad(1)/d(1);
B(1)=B(1)/d(1);                                                                        Step 2.
for i=2:nR-1
   ad(i)=ad(i)/(d(i)-bd(i)*ad(i-1));
   B(i)=(B(i)-bd(i)*B(i-1))/(d(i)-bd(i)*ad(i-1));                                      Step 3.
end
B(nR)=(B(nR)-bd(nR)*B(nR-1))/(d(nR)-bd(nR)*ad(nR - 1));                                 Step 4.
x(nR,1)=B(nR);
for i=nR-1:-1:1
   x(i,1)=B(i)-ad(i)*x(i+1);                                                           Step 5.
end
```

The user-defined function `Tridiagonal` is next used in a script file program to solve the system in Eq. (4.66).

```
% Example 4-9
k1=8000; k2=9000; k3=15000; k4=12000; k5=10000; k6=18000;
L=1.5; L1=0.18; L2=0.22; L3=0.26; L4=0.19; L5=0.15; L6=0.30;
a=[k1 + k2, -k2, 0, 0, 0; -k2, k2+k3, -k3, 0, 0; 0, -k3, k3+k4, -k4, 0
    0, 0, -k4, k4+k5, -k5; 0, 0, 0, -k5, k5+k6];
b=[k1*L1-k2*L2; k2*L2-k3*L3; k3*L3-k4*L4; k4*L4-k5*L5; k5*L5+k6*L-k6*L6];
Xs=Tridiagonal(a,b)
```

When the script file is executed, the following solution is displayed in the Command Window.

```
Xs =
   0.2262
   0.4872
   0.7718
   0.9926
   1.1795
>>
```

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

4.10 ERROR, RESIDUAL, NORMS, AND CONDITION NUMBER

A numerical solution of a system of equations is seldom an exact solution. Even though direct methods (Gauss, Gauss–Jordan, *LU* decomposition) can be exact, they are still susceptible to round-off errors when implemented on a computer. This is especially true with large systems and with ill-conditioned systems (see Section 4.11). Solutions that are obtained with iterative methods are approximate by nature. This section describes measures that can be used for quantifying the accuracy, or estimating the magnitude of the error, of a numerical solution.

4.10.1 Error and Residual

If $[x_{NS}]$ is a computed approximate numerical solution of a system of n equations $[a][x]=[b]$ and $[x_{TS}]$ is the true (exact) solution, then the true error is the vector:

$$[e] = [x_{TS}]-[x_{NS}] \tag{4.67}$$

The true error, however, cannot in general be calculated because the true solution is not known.

An alternative measure of the accuracy of a solution is the residual $[r]$, which is defined by:

$$[r] = [a][x_{TS}]-[a][x_{NS}] = [b]-[a][x_{NS}] \tag{4.68}$$

In words, $[r]$ measures how well the system of equations is satisfied when $[x_{NS}]$ is substituted for $[x]$. (This is equivalent to the tolerance in $f(x)$ when the solution of a single equation is considered. See Eq. (3.5) in Section 3.2.) The vector $[r]$ has n elements, and if the numerical solution is close to the true solution, then all the elements of $[r]$ are small. It should be remembered that $[r]$ does not really indicate how small the error is in the solution $[x]$. $[r]$ only shows how well the right-hand side of the equations is satisfied when $[x_{NS}]$ is substituted for $[x]$ in the original equations. This depends on the magnitude of the elements of the matrix $[a]$. As shown next in Example 4-10, it is possible to have an approximate numerical solution that has a large true error but gives a small residual.

A more accurate estimate of the error in a numerical solution can be obtained by using quantities that measure the size, or magnitude, of vectors and matrices. For numbers, it is easy to determine which one is large or small by comparing their absolute values. It is more difficult to measure the magnitude (size) of vectors and matrices. This is done by a quantity called ***norm***, which is introduced next.

Example 4-10: Error and residual.

The true (exact) solution of the system of equations:

$$1.02x_1 + 0.98x_2 = 2$$
$$0.98x_1 + 1.02x_2 = 2$$

is $x_1 = x_2 = 1$.

Calculate the true error and the residual for the following two approximate solutions:

(*a*) $x_1 = 1.02$, $x_2 = 1.02$.

(b) $x_1 = 2$, $x_2 = 0$.

SOLUTION

In matrix form, the given system of equations is $[a][x] = [b]$, where $[a] = \begin{bmatrix} 1.02 & 0.98 \\ 0.98 & 1.02 \end{bmatrix}$ and $[b] = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$.

The true solution is $[x_{TS}] = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

The true error and the residual are given by Eqs. (4.67) and (4.68), respectively. Applying these equations to the two approximate solutions gives:

(*a*) In this case $[x_{NS}] = \begin{bmatrix} 1.02 \\ 1.02 \end{bmatrix}$. Consequently, the error and residual are:

$$[e] = [x_{TS}] - [x_{NS}] = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1.02 \\ 1.02 \end{bmatrix} = \begin{bmatrix} -0.02 \\ -0.02 \end{bmatrix} \quad \text{and}$$

$$[r] = [b] - [a][x_{NS}] = [b] - [a][x_{NS}] = \begin{bmatrix} 2 \\ 2 \end{bmatrix} - \begin{bmatrix} 1.02 & 0.98 \\ 0.98 & 1.02 \end{bmatrix}\begin{bmatrix} 1.02 \\ 1.02 \end{bmatrix} = \begin{bmatrix} -0.04 \\ -0.04 \end{bmatrix}$$

In this case, both the error and the residual are small.

(*b*) In this case, $[x_{NS}] = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$. Consequently, the error and residual are:

$$[e] = [x_{TS}] - [x_{NS}] = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \text{and}$$

$$[r] = [b] - [a][x_{NS}] = [b] - [a][x_{NS}] = \begin{bmatrix} 2 \\ 2 \end{bmatrix} - \begin{bmatrix} 1.02 & 0.98 \\ 0.98 & 1.02 \end{bmatrix}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.04 \\ 0.04 \end{bmatrix}$$

In this case, the error is large but the residual is small.

This example shows that a small residual does not necessarily guarantee a small error. Whether or not a small residual implies a small error depends on the "magnitude" of the matrix $[a]$.

4.10.2 Norms and Condition Number

A ***norm*** is a real number assigned to a matrix or vector that satisfies the following four properties:

(*i*) The norm of a vector or matrix denoted by $\|[a]\|$ is a positive quantity. It is equal to zero only if the object $[a]$ itself is zero. In other words, $\|[a]\| \ge 0$ and $\|[a]\| = 0$ only if $[a] = 0$. This statement means that all vectors or matrices except for the zero vector or zero matrix have a positive magnitude.

(*ii*) For all numbers α, $\|\alpha[a]\| = |\alpha|\|[a]\|$. This statement means that the two objects $[a]$ and $[-a]$ have the same "magnitude" and that the magnitude of $[10a]$ is 10 times the magnitude of $[a]$.

(*iii*) For matrices and vectors, $\|[a][x]\| \le \|[a]\|\|[x]\|$, which means that the norm of a product of two matrices is equal to or smaller than the product of the norms of each matrix.

(*iv*) For any two vectors or matrices $[a]$ and $[b]$,

$$\|[a+b]\| \le \|[a]\| + \|[b]\| \tag{4.69}$$

This statement is known as the ***triangle inequality*** because for vectors $[a]$ and $[b]$ it states that the sum of the lengths of two sides of a triangle can never be smaller than the length of the third side.

Any norm of a vector or a matrix must satisfy the four properties listed above in order to qualify as a legitimate measure of its "magnitude." Different ways of calculating norms for vectors and matrices are described next.

Vector norms

For a given vector $[v]$ of n elements, the ***infinity norm*** written as $\|v\|_\infty$ is defined by:

$$\|v\|_\infty = \max_{1 \le i \le n} |v_i| \tag{4.70}$$

In words, $\|v\|_\infty$ is a number equal to the element v_i with the largest absolute value.

The ***1-norm*** written as $\|v\|_1$ is defined by:

$$\|v\|_1 = \sum_{i=1}^{n} |v_i| \tag{4.71}$$

In words, $\|v\|_1$ is the sum of the absolute values of the elements of the vector.

The ***Euclidean 2-norm*** written as $\|v\|_2$ is defined by:

$$\|v\|_2 = \left(\sum_{i=1}^{n} v_i^2\right)^{1/2} \tag{4.72}$$

In words, $\|v\|_2$ is the square root of the sum of the square of the elements. It is also called the magnitude of the vector $[v]$.

Matrix norms

The matrix ***infinity norm*** is given by:

$$\|[a]\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| \tag{4.73}$$

In words, the absolute values of the elements in each row of the matrix are added. The value of the largest sum is assigned to $\|a\|_\infty$.

The matrix ***1-norm*** is calculated by:

$$\|[a]\|_1 = \max_{1 \le j \le n} \sum_{j=1}^{n} |a_{ij}| \tag{4.74}$$

It is similar to the infinity norm, except that the summation of the absolute values of the elements is done for each column, and the value of the largest sum is assigned to $\|a\|_1$.

The 2-norm of a matrix is evaluated as the spectral norm:

$$\|[a]\|_2 = max\left(\frac{\|[a][v]\|}{\|[v]\|}\right) \tag{4.75}$$

where $[v]$ is an eigenvector of the matrix $[a]$ corresponding to an eigenvalue λ. (Eigenvalues and eigenvectors are covered in Chapter 5.) The 2-norm of a matrix is calculated by MATLAB using a technique called singular value decomposition, where the matrix $[a]$ is factored into $[a] = [u][d][v]$, where $[u]$ and $[v]$ are orthogonal matrices (special matrices with the property $[u]^{-1} = [u]^T$), and where $[d]$ is a diagonal matrix. The largest value of the diagonal elements of $[d]$ is used as the 2-norm of the matrix $[a]$.

The Euclidean norm for an $m \times n$ matrix $[a]$ (which is different from the 2-norm of a matrix) is given by:

$$\|[a]\|_{Euclidean} = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2} \tag{4.76}$$

Using norms to determine bounds on the error of numerical solutions

From Eqs. (4.67) and (4.68), the residual can be written in terms of the error $[e]$ as:

$$[r] = [a][x_{TS}] - [a][x_{NS}] = [a]([x_{TS}] - [x_{NS}]) = [a][e] \tag{4.77}$$

If the matrix $[a]$ is invertible (otherwise the system of equations does not have a solution), the error can be expressed as:

$$[e] = [a]^{-1}[r] \tag{4.78}$$

Applying property (*iii*) of the matrix norm to Eq. (4.78) gives:

$$\|[e]\| = \|[a]^{-1}[r]\| \le \|[a]^{-1}\|\|[r]\| \tag{4.79}$$

From Eq. (4.77), the residual $[r]$ is:

$$[r] = [a][e] \tag{4.80}$$

Applying property (*iii*) of the matrix norm to Eq. (4.80) gives:

$$\|[r]\| = \|[a][e]\| \le \|[a]\|\|[e]\| \tag{4.81}$$

The last equation can be rewritten as:

$$\frac{\|[r]\|}{\|[a]\|} \le \|[e]\| \tag{4.82}$$

Equations (4.79) and (4.82) can be combined and written in the form:

$$\frac{\|[r]\|}{\|[a]\|} \le \|[e]\| = \|[a]^{-1}[r]\| \le \|[a]^{-1}\|\|[r]\| \tag{4.83}$$

To use Eq. (4.83), two new quantities are defined. One is the ***relative error*** defined by $\|[e]\|/\|[x_{TS}]\|$, and the second is the ***relative residual*** defined by $\|[r]\|/\|[b]\|$. For an approximate numerical solution, the residual can be calculated from Eq. (4.68). With the residual known, Eq. (4.83) can be used for obtaining an upper bound and a lower bound on the relative error in terms of the relative residual. This is done by dividing Eq. (4.83) by $\|[x_{TS}]\|$, and rewriting the equation in the form:

$$\frac{1}{\|[a]\|}\frac{\|b\|}{\|[x_{TS}]\|}\frac{\|[r]\|}{\|[b]\|} \le \frac{\|[e]\|}{\|[x_{TS}]\|} \le \|[a]^{-1}\|\frac{\|[b]\|}{\|[x_{TS}]\|}\frac{\|[r]\|}{\|[b]\|} \tag{4.84}$$

Since $[a][x_{TS}] = [b]$, property (*iii*) of matrix norms gives: $\|[b]\| \le \|[a]\|\|[x_{TS}]\|$ or $\frac{\|[b]\|}{\|[x_{TS}]\|} \le \|[a]\|$, and this means that $\|[a]\|$ can be substituted for $\frac{\|[b]\|}{\|[x_{TS}]\|}$ in the right-hand side of Eq. (4.84). Similarly, since $[x_{TS}] = [a]^{-1}[b]$, property (*iii*) of matrix norms gives $\|[x_{TS}]\| \le \|[a]^{-1}\|\|[b]\|$ or $\frac{1}{\|[a]^{-1}\|} \le \frac{\|[b]\|}{\|[x_{TS}]\|}$, and this means that $\frac{1}{\|[a]^{-1}\|}$ can be substituted for $\frac{\|[b]\|}{\|[x_{TS}]\|}$ in the left-hand side of Eq. (4.84). With these substitutions, Eq. (4.84) becomes:

$$\frac{1}{\|[a]\|\|[a]^{-1}\|}\frac{\|[r]\|}{\|[b]\|} \le \frac{\|[e]\|}{\|[x_{TS}]\|} \le \|[a]^{-1}\|\|[a]\|\frac{\|[r]\|}{\|[b]\|} \tag{4.85}$$

Equation (4.85) is the main result of this section. It provides a means for bounding the error in a numerical solution of a system of equations. Equation (4.85) states that the true relative error, $\frac{\|[e]\|}{\|[x_{TS}]\|}$ (which is not known), is bounded between $\dfrac{1}{\|[a]\|\|[a]^{-1}\|}$ times the relative residual, $\dfrac{\|[r]\|}{\|[b]\|}$ (lower bound), and $\|[a]^{-1}\|\|[a]\|$ times the relative residual (upper bound). The relative residual can be calculated from the approximate numerical solution so that the true relative error can be bounded if the quantity $\|[a]\|\|[a]^{-1}\|$ (called condition number) can be calculated.

Condition number

The number $\|[a]\|\|[a]^{-1}\|$ is called the ***condition number*** of the matrix $[a]$. It is written as:

$$Cond[a] = \|[a]\|\|[a]^{-1}\| \tag{4.86}$$

- The condition number of the identity matrix is 1. The condition number of any other matrix is 1 or greater.
- If the condition number is approximately 1, then the true relative error is of the same order of magnitude as the relative residual.
- If the condition number is much larger than 1, then a small relative residual does not necessarily imply a small true relative error.
- For a given matrix, the value of the condition number depends on the matrix norm that is used.
- The inverse of a matrix has to be known in order to calculate the condition number of the matrix.

Example 4-11 illustrates the calculation of error, residual, norms, and condition number.

Example 4-11: Calculating error, residual, norm and condition number.

Consider the following set of four equations (the same that was solved in Example 4-8).

$$\begin{aligned}
9x_1 - 2x_2 + 3x_3 + 2x_4 &= 54.5\\
2x_1 + 8x_2 - 2x_3 + 3x_4 &= -14\\
-3x_1 + 2x_2 + 11x_3 - 4x_4 &= 12.5\\
-2x_1 + 3x_2 + 2x_3 + 10x_4 &= -21
\end{aligned}$$

The true solution of this system is $x_1 = 5$, $x_2 = -2$, $x_3 = 2.5$, and $x_4 = -1$. When this system was solved in Example 4-8 with the Gauss–Seidel iteration method, the numerical solution in the sixth iteration was $x_1 = 4.98805$, $x_2 = -1.99511$, $x_3 = 2.49806$, and $x_4 = -1.00347$.

(*a*) Determine the true error, $[e]$, and the residual, $[r]$.

(*b*) Determine the infinity norms of the true solution, $[x_{TS}]$, the error, $[e]$, the residual, $[r]$, and the vector $[b]$.

(*c*) Determine the inverse of $[a]$, the infinity norm of $[a]$ and $[a]^{-1}$, and the condition number of the matrix $[a]$.

(*d*) Substitute the quantities from parts (*b*) and (*c*) in Eq. (4.85) and discuss the results.

SOLUTION

First, the equations are written in matrix form:

$$\begin{bmatrix} 9 & -2 & 3 & 2 \\ 2 & 8 & -2 & 3 \\ -3 & 2 & 11 & -4 \\ -2 & 3 & 2 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 54.5 \\ -14 \\ 12.5 \\ -21 \end{bmatrix}$$

(*a*) The true solution is $x_{TS} = \begin{bmatrix} 5 \\ -2 \\ 2.5 \\ -1 \end{bmatrix}$, and the approximate numerical solution is $x_{NS} = \begin{bmatrix} 4.98805 \\ -1.99511 \\ 2.49806 \\ -1.00347 \end{bmatrix}$.

The error is then: $[e] = [x_{TS}] - [x_{NS}] = \begin{bmatrix} 5 \\ -2 \\ 2.5 \\ -1 \end{bmatrix} - \begin{bmatrix} 4.98805 \\ -1.99511 \\ 2.49806 \\ -1.00347 \end{bmatrix} = \begin{bmatrix} 0.0119 \\ -0.0049 \\ 0.0019 \\ 0.0035 \end{bmatrix}$.

The residual is given by Eq. (4.77) $[r] = [a][e]$. It is calculated with MATLAB (Command Window):

```
>> a=[9 -2 3 2; 2 8 -2 3; -3 2 11 -4; -2 3 2 10];
>> e=[0.0119; -0.0049; 0.0019; 0.0035];
>> r=a*e
r =
   0.13009000000000
  -0.00869000000000
  -0.03817000000000
   0.00001000000000
```

(*b*) The infinity norm of a vector is defined in Eq. (4.70): $\|v\|_\infty = \max\limits_{1 \le i \le n} |v_i|$. Using this equation to calculate infinity norm of the true solution, the residual, and the vector $[b]$ gives:

$$\|x_{TS}\|_\infty = \max_{1 \le i \le 4} |x_{TS\,i}| = \max[|5|, |-2|, |2.5|, |-1|] = 5$$

$$\|e\|_\infty = \max_{1 \le i \le 4} |e_i| = \max[|0.0119|, |-0.0049|, |0.0019|, |0.0035|] = 0.0119$$

$$\|r\|_\infty = \max_{1 \le i \le 4} |r_i| = \max[|0.13009|, |-0.00869|, |-0.03817|, |0.00001|] = 0.13009$$

$$\|b\|_\infty = \max_{1 \le i \le 4} |b_i| = \max[|54.5|, |-14|, |12.5|, |-21|] = 54.5$$

(*c*) The inverse of $[a]$ is calculated by using MATLAB's `inv` function (Command Window):

```
>> aINV=inv(a)
aINV =
   0.0910    0.0386   -0.0116   -0.0344
  -0.0206    0.1194    0.0308   -0.0194
   0.0349   -0.0200    0.0727    0.0281
   0.0174   -0.0241   -0.0261    0.0933
```

The infinity norms of $[a]$ and $[a]^{-1}$ are calculated by using Eq. (4.73), $\|[a]\|_\infty = \max\limits_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$:

$$\|[a]\|_\infty = \max_{1 \le i \le 4} \sum_{j=1}^{n} |a_{ij}| = max[|9|+|-2|+|3|+|2|,\ |2|+|8|+|-2|+|3|,\ |-3|+|2|+|11|+|-4|,\ |9|+|-2|+|3|+|2|]$$

$$\|[a]\|_\infty = max[16, 15, 20, 16] = 20$$

$$\|[a]^{-1}\|_\infty = \max_{1 \le i \le 4} \sum_{j=1}^{n} |a_{ij}^{-1}| = max[|0.091| + |0.0386| + |-0.0116| + |-0.0344|,\ |-0.0206| + |0.1194| +$$

$$|0.0308| + |-0.0194|,\ |0.0349| + |-0.02| + |0.0727| + |0.0281|,\ |0.0174| + |-0.0241| + |-0.0261| + |0.0933|]$$

$$\|[a]^{-1}\|_\infty = max[0.1756, 0.1902, 0.1557, 0.1609] = 0.1902$$

The condition number of the matrix $[a]$ is calculated by using Eq. (4.86):

$$Cond[a] = \|[a]\|\|[a]^{-1}\| = 20 \cdot 0.1902 = 3.804$$

(*d*) Substituting all the variables calculated in parts (*b*) and (*c*) in Eq.(4.85) gives:

$$\frac{1}{\|[a]\|\|[a]^{-1}\|}\frac{\|[r]\|}{\|[b]\|} \le \frac{\|[e]\|}{\|[x_{TS}]\|} \le \|[a]^{-1}\|\,\|[a]\|\frac{\|[r]\|}{\|[b]\|}$$

$$\frac{1}{3.804}\frac{0.13009}{54.5} \le \frac{\|[e]\|}{\|[x_{TS}]\|} \le 3.804\frac{0.13009}{54.5}$$

$$\frac{1}{3.804}0.002387 \le \frac{\|[e]\|}{\|[x_{TS}]\|} \le 3.804 \cdot 0.002387, \quad \text{or} \quad 6.275 \times 10^{-4} \le \frac{\|[e]\|}{\|[x_{TS}]\|} \le 0.00908$$

These results indicate that the magnitude of the true relative error is between 6.275×10^{-4} and 0.00908. In this problem, the magnitude of the true relative error can be calculated because the true solution is known.

The magnitude of the true relative error is:

$$\frac{\|[e]\|}{\|[x_{TS}]\|} = \frac{0.0119}{5} = 0.00238$$, which is within the bounds calculated by Eq. (4.85).

4.11 ILL-CONDITIONED SYSTEMS

An ill-conditioned system of equations is one in which small variations in the coefficients cause large changes in the solution. The matrix of coefficients of ill-conditioned systems generally has a condition number that is significantly greater than 1. As an example, consider the system:

$$\begin{aligned} 6x_1 - 2x_2 &= 10 \\ 11.5x_1 - 3.85x_2 &= 17 \end{aligned} \qquad (4.87)$$

The solution of this system is:

$$x_1 = \frac{a_{12}b_2 - a_{22}b_1}{a_{12}a_{21} - a_{11}a_{22}} = \frac{-2 \cdot 17 - (-3.85 \cdot 10)}{-2 \cdot 11.5 - (6 \cdot -3.85)} = \frac{4.5}{0.1} = 45$$

$$x_2 = \frac{a_{21}b_1 - a_{11}b_2}{a_{12}a_{21} - a_{11}a_{22}} = \frac{11.5 \cdot 10 - (6 \cdot 17)}{-2 \cdot 11.5 - (6 \cdot -3.85)} = \frac{13}{0.1} = 130$$

If a small change is made in the system by changing a_{22} to 3.84,

$$\begin{aligned} 6x_1 - 2x_2 &= 10 \\ 11.5x_1 - 3.84x_2 &= 17 \end{aligned} \qquad (4.88)$$

then the solution is:

$$x_1 = \frac{a_{12}b_2 - a_{22}b_1}{a_{12}a_{21} - a_{11}a_{22}} = \frac{-2 \cdot 17 - (-3.84 \cdot 10)}{-2 \cdot 11.5 - (6 \cdot -3.84)} = \frac{4.4}{0.04} = 110$$

$$x_2 = \frac{a_{21}b_1 - a_{11}b_2}{a_{12}a_{21} - a_{11}a_{22}} = \frac{11.5 \cdot 10 - (6 \cdot 17)}{-2 \cdot 11.5 - (6 \cdot -3.84)} = \frac{13}{0.04} = 325$$

It can be observed that there is a very large difference between the solutions of the two systems. A careful examination of the solutions of Eqs. (4.87) and (4.88) shows that the numerator of the equation for x_2 in both solutions is the same and that there is only a small difference in the numerator of the equation for x_1. At the same time, there is a large difference (a factor of 2.5) between the denominators of the two equations. The denominators of both equations are the determinants of the matrices of coefficients $[a]$.

The fact that the system in Eqs. (4.87) is ill-conditioned is evident from the value of the condition number. For this system:

$$[a] = \begin{bmatrix} 6 & -2 \\ 11.5 & -3.85 \end{bmatrix} \quad \text{and} \quad [a]^{-1} = \begin{bmatrix} 38.5 & -20 \\ 115 & -60 \end{bmatrix}$$

Using the infinity norm, Eq. (4.73), the condition number for the system is:

$$Cond[a] = \|[a]\|\|[a]^{-1}\| = 15.35 \cdot 175 = 2686.25$$

Using the 1-norm, Eq. (4.74), the condition number for the system is:

$$Cond[a] = \|[a]\|\|[a]^{-1}\| = 17.5 \cdot 153.5 = 2686.25$$

Using the 2-norm, the condition number for the system is (the norms were calculated with MATLAB built-in `norm(a,2)` function):

$$Cond[a] = \|[a]\|\|[a]^{-1}\| = 13.6774 \cdot 136.774 = 1870.7$$

These results show that with any norm used, the condition number of the matrix of coefficients of the system in Eqs. (4.87) is much larger than 1. This means that the system is likely ill-conditioned.

When an ill-conditioned system of equations is being solved numerically, there is a high probability that the solution obtained will have a large error or that a solution will not be obtained at all. In general, it is difficult to quantify the value of the condition number that can precisely identify an ill-conditioned system. This depends on the precision of the computer used and other factors. Thus, in practice, one needs to worry only about whether or not the condition number is much larger than 1, and not about its exact value. Furthermore, it might not be possible to calculate the determinant and the condition number for an ill-conditioned system anyway because the mathematical operations done in these calculations are similar to the operations required in solving the system.

4.12 PROBLEMS

Problems to be solved by hand

Solve the following problems by hand. When needed, use a calculator, or write a MATLAB script file to carry out the calculations. If using MATLAB, do not use built-in functions for operations with matrices.

4.1 Solve the following system of equations using the Gauss elimination method:

$$\begin{aligned} 2x_1 + x_2 - x_3 &= 1 \\ x_1 + 2x_2 + x_3 &= 8 \\ -x_1 + x_2 - x_3 &= -5 \end{aligned}$$

4.2 Given the system of equations $[a][x] = [b]$, where $a = \begin{bmatrix} 2 & -2 & 1 \\ 3 & 2 & -5 \\ -1 & 2 & 3 \end{bmatrix}$, $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, and $b = \begin{bmatrix} 10 \\ -16 \\ 8 \end{bmatrix}$, determine the solution using the Gauss elimination method.

4.3 Consider the following system of two linear equations: $\begin{aligned} 0.0003x_1 + 1.566x_2 &= 1.569 \\ 0.3454x_1 - 2.436x_2 &= 1.018 \end{aligned}$.

(*a*) Solve the system with the Gauss elimination method using rounding with four significant figures.

(*b*) Switch the order of the equations, and solve the system with the Gauss elimination method using rounding with four significant figures.

Check the answers by substituting the solution back in the equations.

4.4 Solve the following system of equations using the Gauss elimination method.

$$\begin{aligned} 2x_1 + x_2 - x_3 + 2x_4 &= 0 \\ x_1 - 2x_2 + x_3 - 4x_4 &= 3 \\ 3x_1 - x_2 - 2x_3 - x_4 &= -3 \\ -x_1 + 2x_2 + x_3 - 2x_4 &= 13 \end{aligned}$$

4.5 Solve the following system of equations with the Gauss elimination method.

$$\begin{aligned} 2x_1 + x_2 - x_3 + 4x_4 &= 19 \\ -x_1 - 2x_2 + x_3 + 2x_4 &= -3 \\ 2x_1 + 4x_2 + 2x_3 + x_4 &= 25 \\ -x_1 + x_2 - x_3 - 2x_4 &= -5 \end{aligned}$$

4.6 Solve the following system of equations using the Gauss–Jordan method.

$$\begin{aligned} 4x_1 + x_2 + 2x_3 &= 21 \\ 2x_1 - 2x_2 + 2x_3 &= 8 \\ x_1 - 2x_2 + 4x_3 &= 16 \end{aligned}$$

4.7 Solve the system of equations given in Problem 4.2 using the Gauss–Jordan method.

4.8 Given the system of equations $[a][x] = [b]$, where $a = \begin{bmatrix} 4 & 5 & -2 \\ 2 & -5 & 2 \\ 6 & 2 & 4 \end{bmatrix}$, $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, and $b = \begin{bmatrix} -6 \\ 24 \\ 30 \end{bmatrix}$, determine the solution using the Gauss–Jordan method.

4.9 Solve the following system of equations with the Gauss–Jordan elimination method.

$$\begin{aligned} 4x_1 + 3x_2 + 2x_3 + x_4 &= 17 \\ 2x_1 - x_2 + 2x_3 - 4x_4 &= 11 \\ x_1 + 2x_2 - 2x_3 - x_4 &= 8 \\ -2x_1 + 4x_2 + 5x_3 - x_4 &= 15 \end{aligned}$$

4.10 Determine the LU decomposition of the matrix $a = \begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 1 \\ 6 & -2 & 2 \end{bmatrix}$ using the Gauss elimination procedure.

4.11 Determine the LU decomposition of the matrix $a = \begin{bmatrix} 6 & 12 & 24 \\ 2 & 11 & 29 \\ 4 & 10 & 24 \end{bmatrix}$ using Crout's method.

4.12 Solve the following system with LU decomposition using Crout's method.

$$\begin{bmatrix} 2 & -6 & 6 \\ 3 & -7 & 13 \\ -2 & 2 & -11 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ -13 \\ 21 \end{bmatrix}$$

4.13 Find the inverse of the matrix $\begin{bmatrix} 10 & 12 & 0 \\ 0 & 2 & 8 \\ 2 & 4 & 8 \end{bmatrix}$ using the Gauss–Jordan method.

4.14 Given the matrix $a = \begin{bmatrix} -1 & 2 & 2 \\ 0 & 2 & -0.5 \\ 0.5 & 1 & -2 \end{bmatrix}$, determine the inverse of $[a]$ using the Gauss–Jordan method.

4.15 Carry out the first three iterations of the solution of the following system of equations using the Gauss–Seidel iterative method. For the first guess of the solution, take the value of all the unknowns to be zero.

$$\begin{aligned} 8x_1 + 2x_2 + 3x_3 &= 51 \\ 2x_1 + 5x_2 + x_3 &= 23 \\ -3x_1 + x_2 + 6x_3 &= 20 \end{aligned}$$

4.16 Carry out the first three iterations of the solution of the following system of equations using the Gauss–Seidel iterative method. For the first guess of the solution, take the value of all the unknowns to be zero.

$$\begin{bmatrix} 4 & 0 & 1 & 0 & 1 \\ 2 & 5 & -1 & 1 & 0 \\ 1 & 0 & 3 & -1 & 0 \\ 0 & 1 & 0 & 4 & -2 \\ 1 & 0 & -1 & 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 32 \\ 19 \\ 14 \\ -2 \\ 41 \end{bmatrix}$$

4.17 Find the condition number of the matrix in Problem 4.13 using the infinity norm.

4.18 Find the condition number of the matrix in Problem 4.14 using the infinity norm.

4.19 Find the condition number of the matrix in Problem 4.13 using the 1-norm.

4.20 Find the condition number of the matrix in Problem 4.14 using the 1-norm.

Problems to be programmed in MATLAB

Solve the following problems using the MATLAB environment. Do not use MATLAB's built-in functions for operations with matrices.

4.21 Modify the user-defined function `GaussPivot` in Program 4-2 (Example 4-3) such that in each step of the elimination the pivot row is switched with the row that has a pivot element with the largest absolute numerical value. For the function name and arguments use `x = GaussPivotLarge(a,b)`, where `a` is the matrix of coefficients, `b` is the right-hand-side column of constants, and `x` is the solution.
(*a*) Use the `GaussPivotLarge` function to solve the system of linear equations in Eq. (4.17).
(*b*) Use the `GaussPivotLarge` function to solve the system:

$$\begin{bmatrix} 0 & 3 & 8 & -5 & -1 & 6 \\ 3 & 12 & -4 & 8 & 5 & -2 \\ 8 & 0 & 0 & 10 & -3 & 7 \\ 3 & 1 & 0 & 0 & 0 & 4 \\ 0 & 0 & 4 & -6 & 0 & 2 \\ 3 & 0 & 5 & 0 & 0 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 34 \\ 20 \\ 45 \\ 36 \\ 60 \\ 28 \end{bmatrix}$$

4.22 Write a user-defined MATLAB function that solves a system of n linear equations, $[a][x] = [b]$, with the Gauss–Jordan method. The program should include pivoting in which the pivot row is switched with the row that has a pivot element with the largest absolute numerical value. For the function name and arguments use `x = GaussJordan(a,b)`, where `a` is the matrix of coefficients, `b` is the right-hand-side column of constants, and `x` is the solution.
(*a*) Use the `GaussJordan` function to solve the system:

$$2x_1 + x_2 + 4x_3 - 2x_4 = 19$$
$$-3x_1 + 4x_2 + 2x_3 - x_4 = 1$$
$$3x_1 + 5x_2 - 2x_3 + x_4 = 8$$
$$-2x_1 + 3x_2 + 2x_3 + 4x_4 = 13$$

(*b*) Use the `GaussJordan` function to solve the system:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & -3 & 2 & 5 & -4 & 6 \\ 6 & 1 & -2 & 4 & 3 & 5 \\ 3 & 2 & -1 & 4 & 5 & 6 \\ 4 & -2 & -1 & 3 & 6 & 5 \\ 5 & -6 & -3 & 4 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 91 \\ 37 \\ 63 \\ 81 \\ 69 \\ -4 \end{bmatrix}$$

4.23 Write a user-defined MATLAB function that decomposes an $n \times n$ matrix $[A]$ into a lower triangular matrix $[L]$ and an upper triangular matrix $[U]$ (such that $[A] = [L][U]$) using the Gauss elimination method (without pivoting). For the function name and arguments, use `[L,U] = LUdecompGauss(A)`, where the input argument `A` is the matrix to be decomposed and the output arguments `L` and `U` are the corresponding upper and lower triangular matrices. Use `LUdecompGauss` to determine the LU decomposition of the following matrix:

$$\begin{bmatrix} 4 & -1 & 3 & 2 \\ -8 & 0 & -3 & -3.5 \\ 2 & -3.5 & 10 & 3.75 \\ -8 & -4 & 1 & -0.5 \end{bmatrix}$$

4.24 Write a user-defined MATLAB function that determines the inverse of a matrix using the Gauss–Jordan method. For the function name and arguments use `Ainv = Inverse(A)`, where `A` is the matrix to be inverted, and `Ainv` is the inverse of the matrix. Use the `Inverse` function to calculate the inverse of:

(*a*) The matrix $\begin{bmatrix} -1 & 2 & 1 \\ 2 & 2 & -4 \\ 0.2 & 1 & 0.5 \end{bmatrix}$. (*b*) The matrix $\begin{bmatrix} -1 & -2 & 1 & 2 \\ 1 & 1 & -4 & -2 \\ 1 & -2 & -4 & -2 \\ 2 & -4 & 1 & -2 \end{bmatrix}$.

4.25 Write a user-defined MATLAB function that calculates the 1-norm of any matrix. For the function name and arguments use `N = OneNorm(A)`, where `A` is the matrix and `N` is the value of the norm. Use the function for calculating the 1-norm of:

(*a*) The matrix $A = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1.5 \end{bmatrix}$. (*b*) The matrix $B = \begin{bmatrix} 4 & -1 & 0 & 1 & 0 \\ -1 & 4 & -1 & 0 & 1 \\ 0 & -1 & 4 & -1 & 0 \\ 1 & 0 & -1 & 4 & -1 \\ 0 & 1 & 0 & -1 & 4 \end{bmatrix}$.

4.26 Write a user-defined MATLAB function that calculates the infinity norm of any matrix. For the function name and arguments use `N = InfinityNorm(A)`, where `A` is the matrix, and `N` is the value of the norm. Use the function for calculating the infinity norm of:

(*a*) The matrix $A = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1.5 \end{bmatrix}$. (*b*) The matrix $B = \begin{bmatrix} 4 & -1 & 0 & 1 & 0 \\ -1 & 4 & -1 & 0 & 1 \\ 0 & -1 & 4 & -1 & 0 \\ 1 & 0 & -1 & 4 & -1 \\ 0 & 1 & 0 & -1 & 4 \end{bmatrix}$.

4.27 Write a user-defined MATLAB function that calculates the condition number of an $(n \times n)$ matrix by using the 1-norm. For the function name and arguments use `c = CondNumb_One(A)`, where `A` is the matrix and `c` is the value of the condition number. Within the function, use the user-defined functions `Inverse` from Problem 4.24 and `OneNorm` from Problem 4.25. Use the function `CondNumb_One` for calculating the condition number of the matrices in Problem 4.25.

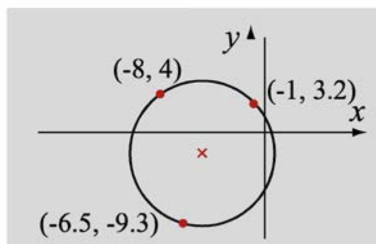
4.28 Write a user-defined MATLAB function that calculates the condition number of an $(n \times n)$ matrix by using the infinity norm. For the function name and arguments use `c = CondNumb_Inf(A)`, where `A` is the matrix and `c` is the value of the condition number. Within the function, use the user-defined functions `Inverse` from Problem 4.24 and `InfinityNorm` from Problem 4.26. Use the function `CondNumb_Inf` for calculating the condition number of the matrices in Problem 4.25.

Problems in math, science, and engineering

Solve the following problems using the MATLAB environment. As stated, use the MATLAB programs that are presented in the chapter, programs developed in previously solved problems, or MATLAB's built-in functions.

4.29 In a Cartesian coordinate system the equation of a circle with its center at point (a, b) and radius r is:

$$(x-a)^2 + (y-b)^2 = r^2$$



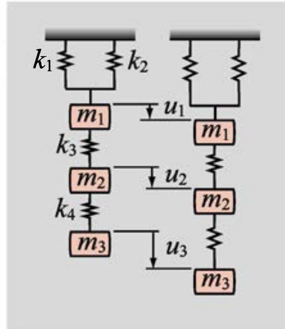
Given three points, $(-1, 3.2)$, $(-8, 4)$, and $(-6.5, -9.3)$, determine the equation of the circle that passes through the points.
Solve the problem by deriving a system of three linear equations (substitute the points in the equation) and solve the system.
(*a*) Use the user-defined function `GaussPivotLarge` developed in Problem 4.21.
(*b*) Solve the system of equations using MATLAB's left division operation.

4.30 In a 3D Cartesian coordinate system the equation of a plane is:

$$ax + by + cz = d$$

Given three points, $(2, -3, -2)$, $(5, 2, 1)$, and $(-1, 5, 4)$, determine the equation of the plane that passes through the points.

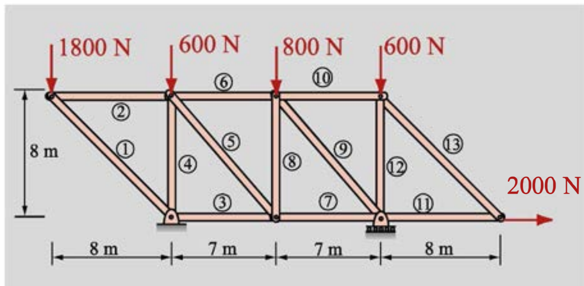
4.31 Three masses, $m_1 = 2$ kg, $m_2 = 3$ kg, and $m_3 = 1.5$ kg, are attached to springs, $k_1 = 30$ N/m, $k_2 = 25$ N/m, $k_3 = 20$ N/m, and $k_4 = 15$ N/m, as shown. Initially the masses are positioned such that the springs are in their natural length (not stretched or compressed); then the masses are slowly released and move downward to an equilibrium position as shown. The equilibrium equations of the three masses are:



$$(k_1 + k_2 + k_3)u_1 - k_3 u_2 = m_1 g$$
$$-k_3 u_1 + (k_3 + k_4)u_2 - k_4 u_3 = m_2 g$$
$$-k_4 u_2 + k_4 u_3 = m_3 g$$

where u_1, u_2, and u_3 are the relative displacement of each mass as shown. Determine the displacement of the three masses. ($g = 9.81\ m/s^2$)

4.32 The axial force F_i in each of the 13-member pin-connected truss, shown in the figure, can be calculated by solving the following system of 13 equations:



$F_2 + 0.707F_1 = 0$, $F_3 - 0.707F_1 - 2000 = 0$

$0.7071F_1 + F_4 + 6229 = 0$, $-F_2 + 0.659F_5 + F_6 = 0$

$-F_4 - 0.753F_5 - 600 = 0$, $-F_3 - 0.659F_5 + F_7 = 0$

$0.753F_5 + F_8 = 0$, $-F_6 + 0.659F_9 + F_{10} = 0$

$-F_8 - 0.753F_9 - 800 = 0$, $-F_7 - 0.659F_9 + F_{11} = 0$

$0.753F_9 + F_{12} - 2429 = 0$, $-F_{10} + 0.707F_{13} = 0$

$-F_{12} - 0.7071F_{13} - 600 = 0$

(*a*) Solve the system of equations using the user-defined function `GaussPivotLarge` developed in Problem 4.21.
(*b*) Solve the system of equations using Gauss–Seidel iteration. Does the solution converge for a starting (guess) vector whose elements are all zero?
(*c*) Solve the system of equations using MATLAB's left division operation.

4.33 A particular dessert consists of 2 lb of bananas, 3 lb of strawberries, 3 lb of cherries, and 4 lb of frozen yogurt. If the cost of the entire batch of this dessert is to be no more than \$20 (in order to yield an acceptable profit), what must the cost of each ingredient be (per pound) if the strawberries cost twice as much as the cherries, and the cherries cost \$1 per pound less than the frozen yogurt, and the frozen yogurt costs as much as half a pound of cherries and 4 pounds of bananas? (Hint: Set up a system of four equations where the unknowns are the cost (per pound) of the bananas (x_1), the cost (per pound) of the strawberries (x_2), the cost (per pound) of the cherries (x_3), the cost (per pound) of the frozen yogurt (x_4), and use the fact that all the ingredient costs have to add up to \$20.)

4.34 A particular chemical substance is produced from three different ingredients A, B, and C, each of which have be dissolved in water first before they react to form the desired substance. Suppose that a solution containing ingredient A at a concentration of 2 g/cm^3 is combined with a solution containing ingredient B at a concentration of 3.6 g/cm^3 and with a solution containing ingredient C at a concentration of 6.3 g/cm^3 to form 25.4 g of the substance. If the concentrations of A, B, and C in these solutions are changed to 4 g/cm^3, 4.3 g/cm^3, and 5.4 g/cm^3, respectively (while the volumes remain the same), then 27.7 g of the substance is produced. Finally, if the concentrations are changed to 7.2, 5.5, and 2.3 g/cm^3, respectively, then 28.3 g of the chemical is produced. Find the volumes (in cubic centimeters) of the solutions containing A, B, and C.

4.35 Mass spectrometry of a sample gives a series of peaks that represent various masses of ions of constituents within the sample. For each peak, the height of the peak I_i is influenced by the amounts of the various constituents:

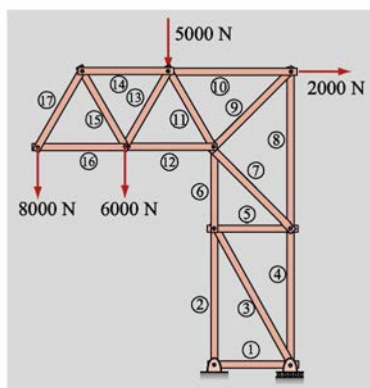
$$I_j = \sum_{i=1}^{N} C_{ij} n_j$$

where C_{ij} is the contribution of ions of species i to the height of peak j, and n_j is the amount of ions or concentration of species j. The coefficients C_{ij} for each peak are given by:

Peak identity	Species				
	CH_4	C_2H_4	C_2H_6	C_3H_6	C_3H_8
1	2	0.5	0	2.4	0.2
2	18	4	0.3	0.2	0.1
3		18	10	0	15
4			12	0	1
5				10	2
6					10

If a sample produces a mass spectrum with peak heights, $I_1 = 30.5$, $I_2 = 71.5$, $I_3 = 354.8$, $I_4 = 180$, $I_5 = 100$, and $I_6 = 36.9$, determine the concentrations of the different species in the sample.

4.36 The axial force F_i in each of the 21 member pin connected truss, shown in the figure, can be calculated by solving the following system of 21 equations:



$-F_1 - 0.342F_3 = 0$, $0.94F_3 + F_4 - 54000 = 0$

$F_5 + 0.342F_3 = 0$, $F_6 - F_2 - 0.94F_3 = 0$

$-F_5 - 0.7071F_7 = 0$, $F_8 + 0.707F_7 - F_4 = 0$

$0.707F_9 + 0.707F_5 - 0.5F_{11} - F_{12} = 0$

$-F_6 - 0.707F_7 + 0.7071F_9 + 0.866F_{11} = 0$

$-F_{10} - 0.707F_9 + 2000 = 0$, $-F_8 - 0.707F_9 = 0$

$F_{10} + 0.5F_{11} - 0.5F_{13} - F_{14} = 0$, $-0.866F_{11} - 0.866F_{13} - 5000 = 0$

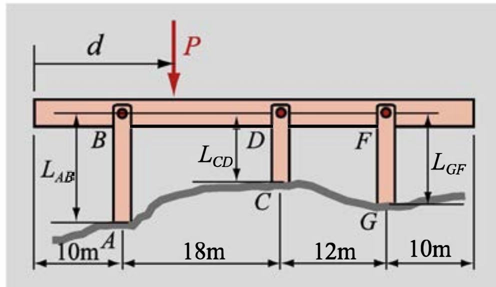
$F_{12} + 0.5F_{13} - 0.5F_{15} - F_{16} = 0$, $0.866F_{13} + 0.866F_{15} - 6000 = 0$

$F_{16} + 0.5F_{17} = 0$, $F_{14} + 0.5F_{15} - 0.5F_{17} = 0$, $-0.866F_{15} - 0.866F_{17} = 0$

(*a*) Solve the system of equations using the user-defined function `GaussJordan` developed in Problem 4.22.

(*b*) Solve the system of equations using MATLAB's left division operation.

4.37 A bridge is modeled by a rigid horizontal bar supported by three elastic vertical columns as shown. A force $P = 40\,\text{kN}$ applied to the rigid bar at a distance d from the end of the bar represents a car on the bridge. The forces in columns F_{AB}, F_{CD}, and F_{GF} can be determined from the solution of the following system of three equations:



$$F_{AB} + F_{CD} + F_{GF} = -P, \quad 10F_{AB} + 28F_{CD} + 40F_{GF} = -d \cdot P$$

$$12F_{AB}L_{AB} - 30F_{CD}L_{CD} + 18F_{GF}L_{GF} = 0$$

Once the force in each of the column is known, its elongation δ can be determined with the formula $\delta = \frac{FL}{EA}$, where E and A are the elastic modulus and the cross-sectional area of each of the columns. Write a MATLAB program in a script file that determines the forces in the three columns and their elongation for $0 \le d \le 50$ m. The program displays the three forces as a function of d in one plot, and the elongation of the three columns as a function of d in a second plot (two plots on the same page). Also given: $L_{AB} = 12$ m, $L_{CD} = 8$ m, $L_{GF} = 10$ m, $E = 70$ GPa, and $A = 25 \cdot 10^{-4}\,\text{m}^2$.

4.38 A food company manufactures the five types of 1.0 lb trail mix packages that have the following composition and cost:

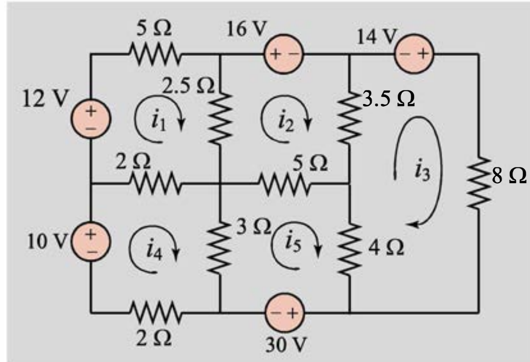
Mix	Peanuts (lb)	Raisins (lb)	Almonds (lb)	Chocolate Chips (lb)	Dried Plums (lb)	Total Cost of Ingredients ($)
A	0.2	0.2	0.2	0.2	0.2	1.44
B	0.35	0.15	0.35	0	0.15	1.16
C	0.1	0.3	0.1	0.1	0.4	1.38
D	0	0.3	0.1	0.4	0.2	1.78
E	0.15	0.3	0.2	0.35	0	1.61

Using the information in the table, determine the cost per pound of each of the ingredients. Write a system of linear equations and solve by using the following methods.

(*a*) Use the user-defined function `GaussJordan` that was developed in Problem 4.22.

(*b*) Use MATLAB's built-in functions.

4.39 The currents, i_1, i_2, i_3, i_4, i_5, in the circuit that is shown can be determined from the solution of the following system of equations. (Obtained by applying Kirchhoff's law.)



$$9.5i_1 - 2.5i_2 - 2i_4 = 12 \ ,$$
$$-2.5i_1 + 11i_2 - 3.5i_3 - 5i_5 = -16$$
$$-3.5i_2 + 15.5i_3 - 4i_5 = 14 \ , \quad -2i_1 + 7i_4 - 3i_5 = 10$$
$$-5i_2 - 4i_3 - 3i_4 + 12i_5 = -30$$

Solve the system using the following methods.

(*a*) Use the user-defined function `GaussJordan` that was developed in Problem 4.22.

(*b*) Use MATLAB's built-in functions.

4.40 When balancing the following chemical reaction by conserving the number of atoms of each element between reactants and products:

$$P_2I_4 + aP_4 + bH_2O \leftrightarrows cPH_4I + dH_3PO_4$$

the unknown stoichiometric coefficients *a*, *b*, *c*, and *d* are given by the solution of the following system of equations:

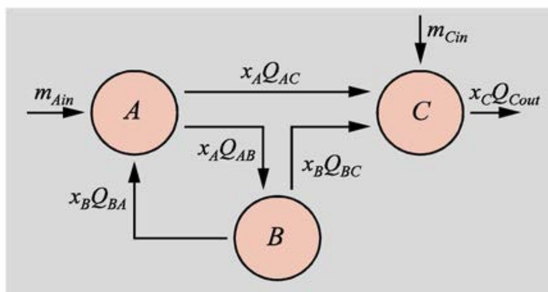
$$\begin{bmatrix} -4 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -2 & 4 & 3 \\ 0 & -1 & 0 & 4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \\ 0 \end{bmatrix}$$

Solve for the unknown stoichiometric coefficients using

(*a*) The user-defined function `GaussJordan` that was developed in Problem 4.22.

(*b*) MATLAB's left division operation.

4.41 A certain chemical engineering process application (see figure) involves three chemical reactors *A*, *B*, and *C*. At steady state, the concentrations of a particular species *n* in each reactor has the values x_A, x_B, and x_C in units of mg/m^3. If the flow rates from reactor *i* (*A*, *B*, or *C*) to reactor *j* (*A*, *B*, or *C*) is denoted as Q_{ij} (units of m^3/s), then the mass flow rate of species *n* from reactor *i* to reactor *j* is x_iQ_{ij} (units of mg/s). Since this chemical species is conserved (i.e., neither produced nor destroyed) conservation of mass (of the species) for each reactor must hold. For the process shown in the figure, Q_{AB} = 40 m^3/s, Q_{AC} = 80 m^3/s, Q_{BA} = 60 m^3/s, Q_{BC} = 20 m^3/s, Q_{Cout} = 150 m^3/s, m_{Cin} = 195 mg/s, and m_{Ain} = 1320 mg/s. Write down the mass continuity equations for each reactor and solve them to find the concentrations x_A, x_B, and x_C in each reactor.



4.42 When balancing the following chemical reaction by conserving the number of atoms of each element between reactants and products:

$$(Cr(N_2H_4CO)_6)_4(Cr(CN)_6)_3 + aKMnO_4 + bH_2SO_4 \rightarrow$$
$$cK_2Cr_2O_7 + dMnSO_4 + eCO_2 + fKNO_3 + gK_2SO_4 + hHO_2$$

the unknown stoichiometric coefficients a through h are given by the solution of the following system of equations:

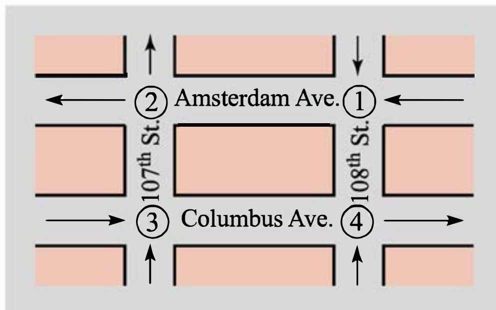
$$\begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -4 & -4 & 7 & 4 & 2 & 3 & 4 & 1 \\ -1 & 0 & 2 & 0 & 0 & 1 & 2 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} 7 \\ 66 \\ 96 \\ 42 \\ 24 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Solve for the unknown stoichiometric coefficients using

(a) The user-defined function `GaussJordan` that was developed in Problem 4.22.

(b) MATLAB's left division operation.

4.43 Traffic congestion is encountered at the intersections shown in the figure. All the streets are one-way and in the directions shown. In order for effective movement of traffic, it is necessary that for every car that arrives at a given corner, another car must leave so that the number of cars arriving per unit time must equal the number of cars leaving per unit time. Traffic engineers gather the following information:



- 600 cars per hour come down Amsterdam Ave. to intersection #1 and 300 cars per hour enter intersection #1 on 108th St.
- 650 cars per hour leave intersection #2 along Amsterdam Ave. and 50 cars per hour leave intersection #2 along 107th St.
- 350 cars per hour come up Columbus Ave. to intersection #3 and 50 cars per hour enter intersection #3 along 107th St.
- 400 cars per hour leave intersection #4 along Columbus Ave. and 300 cars per hour enter intersection #4 from 108th St.

Find n_1, n_2, n_3, and n_4, where n_1 denotes the number of cars traveling per hour along Amsterdam Ave. from intersection #1 to intersection #2, n_2 denotes the number of cars traveling per hour along 107th St. from intersection #3 to intersection #2, n_3 denotes the number of cars traveling per hour along Columbus Ave. from intersection #3 to intersection #4, and n_4 denotes the number of cars traveling per hour along 108th St. from intersection #1 to intersection #4.