Chapter 4

Solving a System of Linear Equations

Core Topics

(i) Gaussian Elimination Method (4.2)

(ii) Gaussian Elimination with Pivoting (4.3)

(iii) Gauss-Jordan Elimination Method (4.4)

(iv) LU Decomposition Method

(v) Inverse of a Matrix (4.6)

(vi) Iterative Methods (Jacobi, Gauss-Seidel) (4.7)

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

(viii) Tridiagonal Systems of Equations (4.9)

(ix) Error, Residual, Norms and Condition Number (4.10)

(x) Ill-Conditioned Systems

Methods for Solving Systems of Linear Algebraic Equations:

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

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

which can be written as:

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

Methods for Solving Systems of Linear Algebraic Equations:

These equations can be solved using analytic, direct or iterative methods. The problem with analytic methods (such as Cramer's Rule) is that it becomes practically impossible to compute when the number of equations is large. Orders of thousands are common for many applications.

With direct methods the solution is calculated using an efficient numerical algorithm which performs arithmetic operations on the equations.

With iterative methods an initial approximate solution is assumed and is then used in an iterative process to obtain successively more accurate solutions.

Direct Methods:

In direct methods the initial system of equations is manipulated to an equivalent system of equations that can be easily solved. Three systems of linear equations that can be easily solved are the upper triangular, lower triangular and diagonal forms.

The upper triangular form is this:

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

Direct Methods:

This equation is easily solved by back substitution. We start with the last equation and solve for x_n. Knowing x_n we can then solve for x_{n-1} using the second last equation and so on.

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

The general form for the solution using back substitution is:

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

Direct Methods:

Likewise a lower triangular form can be solved using forward substitution:

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

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

Direct Methods:

We solve by first solving for x_1 in the first equation and substituting into the second which allows to solve for x_2 and so on.

The general formula (for forward substitution) is:

$$x_1 = \frac{b_1}{a_{11}}$$

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

Direct Methods:

The diagonal form is thus:

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

or

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

(there's a mistake here – spot it!)

The diagonal form is easily solved. Simply:

$$x_i = \frac{b_i}{a_{ii}} \quad \forall i$$

Gaussian Elimination:

Gaussian Elimination:

Here the full set of linear equations is manipulated to be in upper triangular form – which can then be solved using back substitution. Given:

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

or

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

Gaussian Elimination:

We then manipulate the system of equations into an equivalent system which takes the form:

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

or

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

Gaussian Elimination:

We take the first equation (4.10a), called, in this instance, the 'pivot equation' and a_{11} is called the 'pivot coefficient' or 'pivot element'. The pivot element is always a diagonal element. We multiply the first equation (4.10a) by $m_{21} = \frac{a_{21}}{a_{11}}$. The resulting equation is subtracted from the second equation (4.10b) thus eliminating x_1. The pivot equation is unchanged and the new second equation (the result of the subtraction) now has new coefficients. The same process, using the first equation as the pivot equation and a_{11} as the pivot element, is used to eliminate x_1 from all the subsequent equations using $m_{i1} = \frac{a_{i1}}{a_{11}}$ for the i^{th} equation giving:

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

or

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

Gaussian Elimination:

The process is now repeated operating on the third and subsequent equations using the second as the pivot equation and a_{22} as the pivot element thus eliminating x_2 from these equations.

Continuing in this fashion yields an upper triangular system which can be solved using back substitution.

Problems with zero or near-zero pivoting elements can be remedied by changing the order of the system of equations such that no zero or near-zero elements appear on the diagonal.. This process is known as ‘pivoting’.

Example:

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:

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

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

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

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

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

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:

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

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

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

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:

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

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:

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

Example:

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:

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

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:

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

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
```

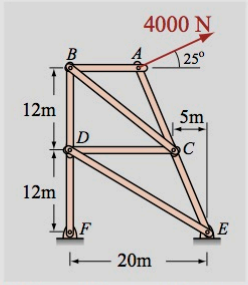
Solution for part (*a*).

Solution for part (*b*).

Example:

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):

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

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
>>
```

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

Gauss-Jordan Elimination:

The Gauss-Jordan Elimination Method is similar to Gaussian Elimination except that:

1. The pivot equation is normalised by dividing all the terms in the pivot equation by the pivot coefficient. This makes the pivot coefficient equal to 1.
2. The pivot equation is used to eliminate the off-diagonal terms in **ALL** equations.

The result is a ***diagonal matrix*** that can easily be solved.

Pivoting applies as with Gaussian Elimination.

Example:

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:

Example:

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

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

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

The solution is:

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

LU Decomposition:

In this method we begin with the equation to be solved:

$$[a][x] = [b]$$

Then:

$$[a] = [L][U] \qquad (4.20)$$

where $[L]$ and $[U]$ are lower and upper diagonal matrices respectively.

LU Decomposition:

We then write:

$$[L][U][x] = [b] \qquad (4.21)$$

Then we say:

$$[U][x] = [y] \qquad (4.22)$$

and

$$[L][y] = [b] \qquad (4.23)$$

Equation (4.23) is then first solved for $[y]$ thus enabling Equation (4.22) to be solved for $[x]$. The advantage of this method is that Equation (4.21) can be easily solved (by back and forward-substitution) for multiple $[b]$s once $[L]$ and $[U]$ have been determined.

LU Decomposition using the Gaussian Elimination Procedure:

When the Gaussian Elimination procedure is applied to matrix $[a]$, the elements of the matrices $[L]$ and $[U]$ are actually calculated.

The upper triangular matrix, $[U]$, is that obtained at the end of the Gaussian Elimination procedure.

The lower triangular matrix, $[L]$ is a matrix with 1s as its diagonal elements and the multipliers m_{ij} that multiply the pivot equation as its off-diagonal elements.

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

LU Decomposition using Crout's Method:

Crout's method is particularly advantageous since we do not have to use vector $[b]$ at all. In Crout's Method the LU decomposition method 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 gives:

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

Direct Methods:

Equation (4.27) is then solved by equating the elements of the product with that of matrix $[a]$. In general:

1. Calculate the first column of $[L]$ using:

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

2. Substitute 1s in the diagonal of $[U]$:

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

3. Calculate the elements in the first row of $[U]$ except for U_{11}

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

4. Then calculate:

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

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

Example:

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

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).

Example:

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:

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

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

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:

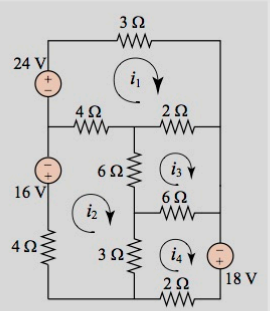
$$x_4 = \frac{0.5}{1} = 0.5, \quad x_3 = -3.333+0.6667x_4 = -3, \quad x_2 = 2.875-0.5x_3-0.75x_4 = 4, \quad \text{and}$$

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

Example:

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:

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

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                                   Step 4 (page 122).
     U(i,j)=(A(i,j)-L(i,1:i-1)*U(1:i-1,j))/L(i,i); Eq. (4.38).
   end
end
```

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:
```

Example:

```
% 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 $[b]$.

The Inverse of a Matrix:

This is most advantageous since it can easily be reused.

Consider:

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

For $[x] = [a]^{-1}$ we have:

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

The Inverse of a Matrix:

We then have:

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

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

Solving these four systems of equations gives the four columns of $[a]^{-1}$. Among other methods, this can be done by Gauss-Jordan elimination and by LU decomposition.

The Inverse of a Matrix by LU Decomposition and Gauss-Jordan Elimination:

We obtain each column of the inverse matrix by solving the system of equations in (4.48).

This can be done by direct application of LU decomposition or Gauss-Jordan elimination on each of these four equations

Example:

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);
[L U]= LUdecompCrout(A);
for j=1:nC
    y=ForwardSub(L,I(:,j));
    invA(:,j)=BackwardSub(U,y);
end
```

Create an identity matrix of the same size as $[A]$.

Decomposition of $[A]$ into $[L]$ and $[U]$.

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

Iterative Methods:

We examine the Jacobi and Gauss-Seidel iterative methods here. The system of linear equations is written in explicit form:

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

Writing the equations in an explicit form. →

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

The idea is to assume an initial solution for $[x]$ and then recursively substitute these values into the r.h.s. of the above explicit form and so obtain new estimates for $[x]$.

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

A sufficient, but not necessary, condition for convergence is:

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

The Jacobi Iterative Method:

An initial solution for $[x]$ is chosen. If no information is available then $[x] = 0$. The values for x are substituted into the r.h.s. of the explicit representation of the system of equations and a new estimate for $[x]$ is found. This process is repeated recursively. That is:

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

The process is continued until the absolute value of the estimated relative error of all unknowns is less than some predetermined value:

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

The Gauss-Seidel Iterative Method:

To begin with, all unknowns x_j except for x_1 are initialised with some approximate values. These can be zero if approximate values are unknown.
Using these values, with $i = 1$ in Equation (4.51), we calculate x_1.
With $i = 2$ in Equation (4.51), we calculate a new value for x_2 and so on. This completes the first iteration.
The second iteration starts with $i = 1$ in Equation (4.51) where a new value of x_1 is calculated. With $i = 2$ a new value of x_2 is calculated and so on.

The iterations are repeated until:

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

The Gauss-Seidel Iterative Method:

Mathematically the Gauss-Seidel algorithm can be written:

$$x_1^{(k+1)} = \frac{1}{a_{11}}\left[b_1 - \sum_{j=2}^{j=n} a_{i1}x_j^{(k)}\right]$$

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

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

The Gauss-Seidel algorithm converges faster than the Jacobi method and like the Jacobi method the sufficient but not necessary condition for convergence is that $[a]$ is diagonally dominant.

Example:

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):

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

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
```

Example:

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 $x_4 = -1$.

Tridiagonal Systems of Equations:

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

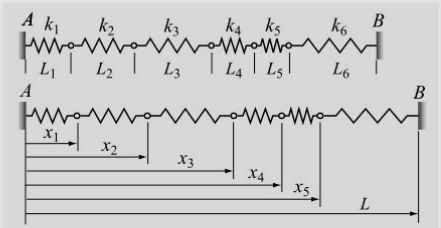
These can be readily solved using previous methods but many elements of $[a]$ are zero. This means more storage space and many redundant zero multiplications and additions.

The Thomas algorithm is a commonsense approach to the problem. Instead of performing Gaussian elimination on $[a]$ as a whole we store the diagonal and off-diagonal entries as separate vectors. We then apply Gaussian elimination as before.

Example:

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);
end
for i=1:nR-1
   ad(i)=A(i,i+1);
end
for i=2:nR
   bd(i)=A(i,i-1);
end
ad(1)=ad(1)/d(1);
B(1)=B(1)/d(1);
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));
end
B(nR)=(B(nR)-bd(nR)*B(nR-1))/(d(nR)-bd(nR)*ad(nR - 1));
x(nR,1)=B(nR);
for i=nR-1:-1:1
   x(i,1)=B(i)-ad(i)*x(i+1);
end
```

Define the vector `d` with the elements of the diagonal.

Define the vector `ad` with the above diagonal elements.

Define the vector `bd` with the below diagonal elements.

Step 1.

Step 2.

Step 3.

Step 4.

Step 5.

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
>>
```

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

Error and Residual:

Consider the equation $[a][x] = [b]$.

The true error is the vector:

$$[e] = [x_{TS}] - [x_{NS}] \qquad (4.67)$$

But because $[x_{TS}]$ cannot in general be calculated we instead use the residual error $[r]$:

$$[r] = [a][x_{TS}] - [a][x_{NS}] = [b] - [a][x_{NS}] \qquad (4.68)$$

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

The smaller the elements in $[r]$ the better $[x_{NS}]$ satisfies the equation. However this does not necessarily tell us how close $[x_{NS}]$ is to $[x_{TS}]$ (see Equation (4.68) – that depends on $[a]^{-1}$).

Example:

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*) $x_1 = 1.02, \quad x_2 = 1.02.$

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

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 $[a]$.

Norms:

The norm of a vector or matrix is a number that in some way indicates collectively the size of the vector or matrix elements. There are various definitions for the norm, each with their own uses, of a vector $[v]$ or a matrix $[a]$. We will concern ourselves with the Euclidean 2-norm in this course:

Vector Norms:

The infinity Norm: $$\|v\|_\infty = \max_{1 \le i \le n} |v_i| \qquad (4.70)$$

The 1-norm: $$\|v\|_1 = \sum_{i=1}^{n} |v_i| \qquad (4.71)$$

The Euclidean 2-norm: $$\|v\|_2 = \left(\sum_{i=1}^{n} v_i^2 \right)^{1/2} \qquad (4.72)$$

This latter one is also referred to as the magnitude of the vector $[v]$ with which you are already familiar.

Norms:

Matrix Norms:

The infinity norm: $$\|[a]\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| \qquad (4.73)$$

The 1-norm: $$\|[a]\|_1 = \max_{1 \le j \le n} \sum_{j=1}^{n} |a_{ij}| \qquad (4.74)$$

The 2-norm: $$\|[a]\|_2 = max\left(\frac{\|[a][v]\|}{\|[v]\|}\right) \qquad (4.75)$$

where $[v]$ is an eigenvector of $[a]$.

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

Properties of Norms:

1. The norm of a vector or matrix $[a]$, denoted $\|a\|$, is a positive quantity and is equal to zero iff $[a] = 0$.

2. $\|\alpha[a]\| = |\alpha|\|[a]\| \; \forall \alpha$. This means $\|[-a]\| = \|[a]\| \geq 0$.

3. $\|[a][x]\| \leq \|[a]\|\|x\|$.

4. $\|[a + b]\| \leq \|[a]\| + \|b\|$ - the triangle inequality.

Using Norms to Determine Error Bounds:

It can be shown that:

$$\frac{1}{\|[a]\|\|[a]^{-1}\|}\frac{\|[r]\|}{\|[b]\|} \leq \frac{\|[e]\|}{\|[x_{TS}]\|} \leq \|[a]^{-1}\|\,\|[a]\|\frac{\|[r]\|}{\|[b]\|} \qquad (4.85)$$

This useful result gives an upper and lower bound to error relative to the true solution.

The Condition Number of a Matrix:

This tells us how stable the solution to a system of equations is.

This number shows how a small perturbation in $[a]$ will affect the solution. If the condition number is much greater than 1 then a small perturbation in $[a]$ will greatly affect the solution. In this case the matrix $[a]$ is said to be ill-conditioned.

By contrast, if a small perturbation in $[a]$ does not affect the solution greatly then $[a]$ is said to be well-conditioned.

The condition number of a matrix $[a]$ is given by:

$$Cond[a] = \|[a]\|\|[a]^{-1}\| \qquad (4.86)$$

The Condition Number of a Matrix:

Take, for example the following system of equations:

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

The solution of this system is:

$$x_1 = 45,\ x_2 = 130$$

If we now change a_{22} slightly to 3.84 then:

$$x_1 = 110,\ x_2 = 325$$

It is clear that $[a] = \begin{pmatrix} 6 & 2 \\ 11.5 & 3.85 \end{pmatrix}$ is ill-conditioned. In fact it has a condition number greater that ~1500 regardless of what norm is used.

Example:

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): $\|v\|_\infty = \max_{1 \le i \le n} |v_i|$. 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), $\|[a]\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$:

Condition Number of a Matrix:

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

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

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

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

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

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

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

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

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

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

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

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).

Recommended Problems:

Problems to be solved by hand (do at least 2):
4.2, 4.13, 4.19

Problems to be programmed in MATLAB:
4.25

Problems in math, science and engineering:
4.34

You should pick out problems that you find interesting/challenging and do these too.