

# Overview of Today's Topics

- Sparse Matrix Data Structures, Linked-List Storage
- LU Factorization
  - Basic Approach:  $[A][X]=[B] \rightarrow [L][U][X]=[B]$
  - Crout's method, row-based example in class
- Introduction to MatLab
  - Interactive commands
  - SAVE-ing and LOAD-ing .mat files
  - Useful functions related to sparse matrices
  - Editing and running .m files
  - MatLab exercise

# Sparse Matrix Data Structures

- Matrices having very few non-zero entries are termed “sparse.”
- Sparsity = numb. zero entries/matrix size
- Density = numb. nonzero entries/matrix size
- Power system admittance matrices [Y] tend to be quite sparse.
  - Example: 10 000-bus system, avg of 3 lines/bus
  - Sparsity is:  $(10^8 - 40\,000) / 10^8 = 99,96\%$

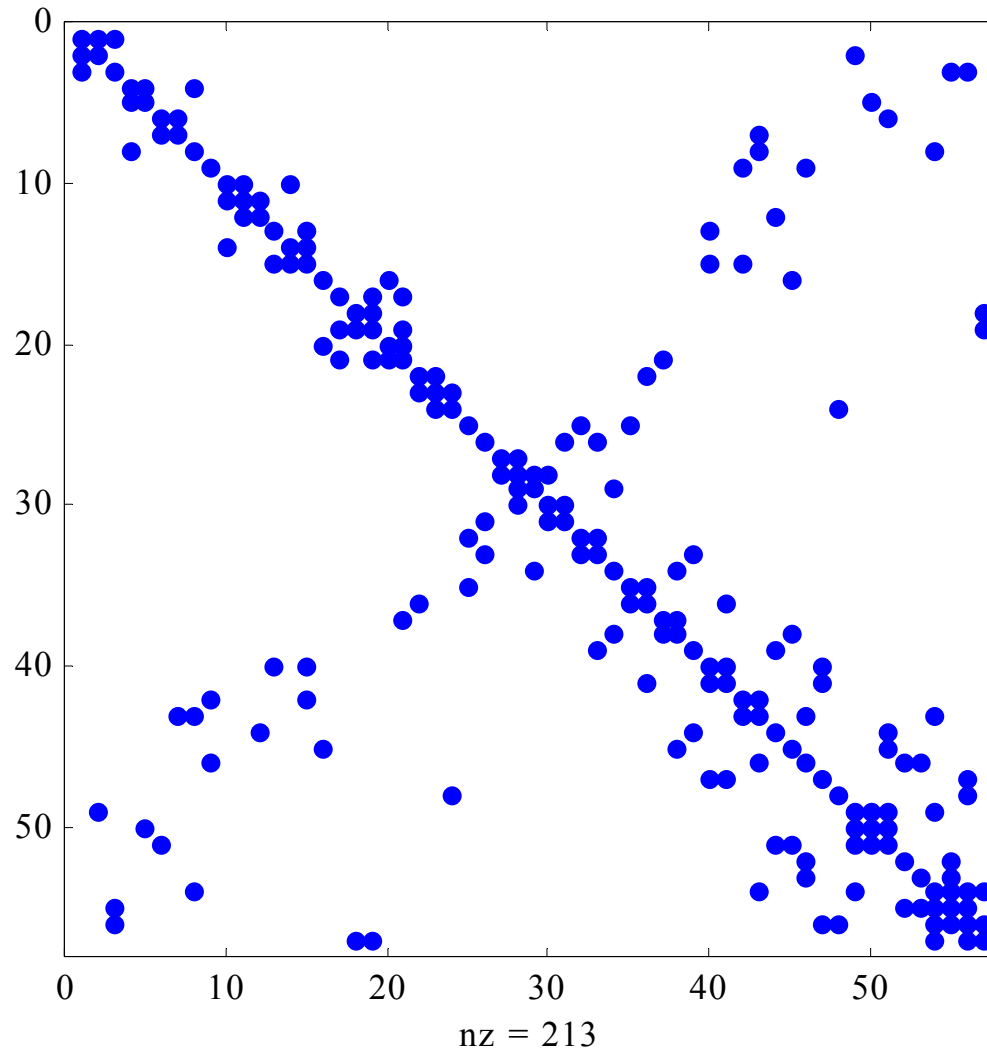
# Helpful MatLab Commands

- General
  - save, load
  - who – lists variables
  - clear a, b, ..
  - inv(A) – inverts matrix
  - zeros, ones
  - find
  - **help xxx** to find help on a specific command
  - HTML help desk
- LU, sparse matrices
  - issparse, sparse, full
  - nnz, nonzeros
  - spy – shows topology
  - [L,U] = lu(A)
  - Reordering: **colmd, symmmd, symrcm, colperm, randperm, dmperm**

# Use of the spy function

57-Bus  
IEEE System

Sparsity:  
93.44%



# Algorithms and Sparse Systems

- Typical situation: Given  $[Y][V] = [I]$  where  $[Y]$  and  $[I]$  are known, solve for  $[V]$ ....
- Direct inversion of  $[Y]$  will result in a **full** matrix  $[Y]^{-1}$
- Therefore, “**in situ**” factorization methods are applied directly to  $[Y]$ .
- $[Y]$  is stored in a **linked-list data structure**, and **dynamically modified** as the factorization progresses.
- We must understand the data structures, for I/O of the data and to pass variables between program modules of C++, Fortran, Pascal, Java, Delphi, etc.

# Full Matrix Storage

- Example: 5x5 coefficient matrix

$$\begin{bmatrix} 3 & 0 & 0 & 2 & 0 \\ 0 & 5 & 1 & 0 & 0 \\ 0 & 1 & 2 & 7 & 9 \\ 2 & 0 & 7 & 1 & 0 \\ 0 & 0 & 9 & 0 & 4 \end{bmatrix}$$

Full storage requirements:  
-25 single-precision values  
-200 bytes if complex values

Data Types:  
-Single precision Real: 4 bytes  
-Double precision Real: 8 bytes  
-Complex: 8 bytes  
-Integer can be 2-byte or 4-byte

# Linked Lists – Several Variations

- Data structure uses only column vectors.
- Only the nonzero matrix values are stored.
- Use integer vector of **pointers** to start of each row.
- Use integer vector of column positions for each value.
- Use integer vector of **links** between the nonzero values of a row.
- Integer values can be unsigned 2-byte integers and still address 65 536 positions in a vector.
- [Switch over to Overhead Projector for example...](#)

# Crout's LU Algorithm

- Algorithm can progress by rows or by columns
- Procedure is straight-forward. Row-method:
  - Copy column one
  - Normalize Row 1: Divide Row 1 off-diagonal entries by diagonal row term of Row 1.
  - For each element  $(i,j)$  where  $i > 1$  and  $j > 1$ , subtract from it the product of  $a_{i1} \cdot a_{1j}$  ie.  $y(i,j) = y(i,j) - y(i,1) \cdot y(1,j)$
  - If the resulting submatrix is of order  $\geq 2$ , repeat the same operations on the submatrix.



# LU Factorization Example

- Solve by LU Factorization
- Use in situ method, Crout's method by row

$$\begin{bmatrix} 2 & 4 & 4 & 2 \\ 3 & 3 & 12 & 6 \\ 2 & 4 & -1 & 2 \\ 4 & 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 8 \\ 10 \end{bmatrix}$$

MatLab Variable Names  $\rightarrow$  **A**  $\rightarrow$  **B**

# Step 1: LU Factorization

Perform LU Factorization  
on the coefficient matrix:

$$\begin{bmatrix} 2 & 4 & 4 & 2 \\ 3 & 3 & 12 & 6 \\ 2 & 4 & -1 & 2 \\ 4 & 2 & 1 & 1 \end{bmatrix} \Rightarrow [L][U]$$

Perform steps on transparency in class...

Result:

How many “fills?”

How many positions

were “eliminated?”

$$\begin{bmatrix} 2 & 2 & 2 & 1 \\ 3 & -3 & -2 & -1 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -19 & -9 \end{bmatrix} = [L \cdot \cdot \cdot U]$$

# L and U

- There are many possible L & U pairs, i.e. the factorization is not unique. In this case, applying the Crout method by row and splitting apart the result to get [L] and [U]:

$$\begin{bmatrix} 2 & 2 & 2 & 1 \\ 3 & -3 & -2 & -1 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -19 & -9 \end{bmatrix} \Rightarrow [L][U] = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & -3 & 0 & 0 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -19 & -9 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 & 1 \\ 0 & 1 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A

LA

UA

Matlab Variable Names


# Solving the Intermediate Equations

- $[A][x]=[B]$
- $[LA][UA][x]=[B]$   
Substituting  $[z]=[U][x]$  , solve
- $[LA][z]=[B]$  by forward substitution  
After solving for  $[z]$  , then solve
- $[U][x]=[z]$  by backward substitution
- No matrix inversion required!

Solve for  $[z]$  by forward substitution

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & -3 & 0 & 0 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -19 & -9 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 8 \\ 10 \end{bmatrix}$$

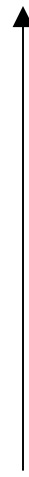
LA B



Solve for  $[x]$  by backward substitution

$$\begin{bmatrix} 1 & 2 & 2 & 1 \\ 0 & 1 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4.00 \\ 2.67 \\ 0.00 \\ -1.11 \end{bmatrix}$$

UA [z]



Step 2

$$\begin{bmatrix} \underline{\underline{2}} & 4 & 4 & 2 \\ 3 & 3 & 12 & 6 \\ 2 & 4 & -1 & 2 \\ 4 & 2 & 1 & 1 \end{bmatrix} \Rightarrow [L][U]$$

①

②

$$\begin{bmatrix} 2 & 2 & 2 & 1 \\ 3 & -3 & 6 & 3 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -7 & -3 \end{bmatrix}$$

$$U = \begin{bmatrix} \phi & 2 & 2 & 1 \\ 0 & \phi & -2 & -1 \\ 0 & 0 & \phi & 0 \\ 0 & 0 & 0 & \phi \end{bmatrix}$$

③

$$\begin{bmatrix} 2 & 2 & 2 & 1 \\ 3 & -3 & -2 & -1 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -19 & -9 \end{bmatrix} \Rightarrow$$

$$L = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & -3 & 0 & 0 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -19 & -9 \end{bmatrix}$$

CHECK!  
 $[L][U] \stackrel{?}{=} [A]$

$$\begin{bmatrix}
 2 & 0 & 0 & 6 \\
 0 & 4 & 8 & 0 \\
 0 & 0 & 2 & 4 \\
 4 & 0 & 0 & 6
 \end{bmatrix}
 \begin{bmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4
 \end{bmatrix}
 =
 \begin{bmatrix}
 8 \\
 4 \\
 0 \\
 10
 \end{bmatrix}$$

"A Fill" = changing a zero to non zero entry.

Key: Re-order the equations to minimize the fills.



# Implications of Sparsity

- What happens to efficiency of LU factorization as sparsity increases?
- Zeros in rows don't have to be normalized!
- A row-column product of zero does not change anything!

$$\begin{bmatrix} 2 & 0 & 0 & 6 \\ 0 & 4 & 0 & 8 \\ 0 & 0 & 2 & 4 \\ 4 & 0 & 0 & 6 \end{bmatrix}$$

elimination without row interchange is given by

$$\begin{aligned} 0.000125x_1 + 1.25x_2 &= 6.25 \\ -1.25 \times 10^5 x_2 &= -6.25 \times 10^5 \end{aligned}$$

Consequently,  $x_2 = 5.00$  and  $x_1 = 0$ , which is clearly unacceptable. Let us now see what happens if we interchange the rows in Eq. (4-47):

$$\begin{bmatrix} 12.5 & 12.5 \\ 0.000125 & 1.25 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 75 \\ 6.25 \end{bmatrix} \quad (4-48)$$

Applying the Gaussian elimination again, we obtain

$$\begin{aligned} 12.5x_1 + 12.5x_2 &= 75 \\ 1.25x_2 &= 6.25 \end{aligned}$$

Hence,  $x_2 = 5.00$  and  $x_1 = 1.00$ , which is obviously the correct solution.

Our example suggests that too small a pivot is likely to cause severe numerical error. It has been suggested [1] that for most matrices a good strategy is to choose the pivot at the  $k$ th stage to be the element with the largest absolute value in column  $k$  of all rows from  $k$  through  $n$ . Testing for size to find the optimum pivot and the subsequent row-interchange operation required to move the desired pivot to the proper position would no doubt require additional computer time. However, the improvement in accuracy is usually worth the extra time. Even further refinement is possible by interchanging not only the rows but also the columns. This strategy is sometimes referred to as *complete pivoting*. For further details concerning this topic, see [3].

 \*4-4 THE LU FACTORIZATION<sup>7</sup>

4-4-1 A Theorem on Factorization

In Section 4-3, we discussed the Gaussian elimination algorithm. It was shown that any nonsingular matrix  $A$  can be reduced to an upper triangular matrix  $U$ , with unit diagonal elements, by a proper sequence of elementary row operations. Or equivalently,

$$\mathbf{\epsilon}_m \mathbf{\epsilon}_{m-1} \dots \mathbf{\epsilon}_2 \mathbf{\epsilon}_1 A = U \quad (4-49)$$

where  $u_{11} = u_{22} = \dots = u_{nn} = 1$ , and  $\mathbf{\epsilon}_k$  may be any of the three types of elementary matrices.

The type 3 elementary matrices in Eq. (4-49) are lower triangular matrices,<sup>8</sup> because we always add  $c$  times one row to a later row. Any type 2 elementary matrix is a lower triangular matrix, but this is not so for a type 1 elementary matrix.

If the pivot elements  $a_{11}, a_{22}^{(2)}, \dots$  at all stages are nonzero, then type 1 elementary matrices are not needed in Eq. (4-49). Since the product of any two lower triangular matrices is also a lower triangular matrix, it follows that the product

<sup>7</sup>The results of this section are used in Chapter 16 in conjunction with sparse-matrix techniques.

<sup>8</sup>A lower triangular matrix  $L$  is a square matrix with zero elements above the principal diagonal.

Chua & Lin  
©1975

$$(\mathbf{\epsilon}_m \mathbf{\epsilon}_{m-1} \dots \mathbf{\epsilon}_2 \mathbf{\epsilon}_1)$$

in Eq. (4-49) is lower triangular. It can be easily shown that the inverse of any nonsingular lower triangular matrix is also lower triangular. Then we can solve for  $A$  in Eq. (4-49) to give

$$\begin{aligned} A &= (\mathbf{\epsilon}_m \mathbf{\epsilon}_{m-1} \dots \mathbf{\epsilon}_2 \mathbf{\epsilon}_1)^{-1} U \\ &= LU \end{aligned} \quad (4-50)$$

where  $L = \mathbf{\epsilon}_1^{-1} \mathbf{\epsilon}_2^{-1} \dots \mathbf{\epsilon}_m^{-1}$  is a lower triangular matrix.

On the other hand, if at any stage the pivot element  $a_{kk}^{(k)}$  is zero, then a row interchange is needed for the elimination procedure to continue. In such a case, some of the elementary matrices in Eq. (4-49) will be of type 1. To explain the effect of row interchange, let us introduce the following subscript notations for the three types of elementary matrices:

- $\mathbf{\epsilon}_{ij}$ : Type 1 elementary matrix obtained from  $\mathbf{1}$  by interchanging the  $i$ th and  $j$ th rows.
- $\mathbf{\epsilon}_i(c)$ : Type 2 elementary matrix obtained from  $\mathbf{1}$  by multiplying the  $i$ th row by  $c$  ( $c \neq 0$ ).
- $\mathbf{\epsilon}_{ij}(c)$ : Type 3 elementary matrix obtained from  $\mathbf{1}$  by adding  $c$  times the  $i$ th row to the  $j$ th row.
- $P$ : Type 1 elementary matrix, or the product of some type 1 elementary matrices.

Then, for the worst case when a row interchange is needed in every stage of the forward-elimination step, Eq. (4-49) for the case  $n = 4$  is of the form<sup>9</sup>

$$\underbrace{\mathbf{\epsilon}_4(c_{10})\mathbf{\epsilon}_3(c_9)\mathbf{\epsilon}_2(c_8)\mathbf{\epsilon}_1(c_7)}_{\text{normalizing diagonal elements}} \underbrace{\mathbf{\epsilon}_{34}(c_6)P_3}_{\text{third stage}} \underbrace{\mathbf{\epsilon}_{24}(c_5)\mathbf{\epsilon}_{23}(c_4)P_2}_{\text{second stage}} \underbrace{\mathbf{\epsilon}_{14}(c_3)\mathbf{\epsilon}_{13}(c_2)\mathbf{\epsilon}_{12}(c_1)P_1}_{\text{first stage}} A = U \quad (4-51)$$

where each  $P_k$  is an  $\mathbf{\epsilon}_{ij}$ . Each  $P_k$  in Eq. (4-51) can be moved to the right of a neighboring  $\mathbf{\epsilon}_j$  matrix provided that "proper changes" are made on the elements of  $\mathbf{\epsilon}_j$  to obtain a new elementary  $\hat{\mathbf{\epsilon}}_j$ . For example, if  $P_k = \mathbf{\epsilon}_{12}$  and  $\mathbf{\epsilon}_j = \mathbf{\epsilon}_{13}(\alpha)$ , then

$$\begin{aligned} P_k \mathbf{\epsilon}_j &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ \alpha & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \alpha & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \hat{\mathbf{\epsilon}}_j P_k \end{aligned}$$

<sup>9</sup>Similar arguments hold for the case  $n > 4$ . But the expression is too lengthy to be given here.

Hence

$$\hat{\epsilon}_j = \epsilon_{2j}(\alpha)$$

The general rules for moving  $P_k$  one step to the right are as follows (the proof is straightforward). Assuming that  $i \neq j \neq k \neq l$ , then

$$\begin{aligned} \epsilon_{ij}\epsilon_{kl}(c) &= \epsilon_{kl}(c)\epsilon_{ij} \\ \epsilon_{ij}\epsilon_{jk}(c) &= \epsilon_{ik}(c)\epsilon_{ij}, & \epsilon_{ij}\epsilon_{ik}(c) &= \epsilon_{jk}(c)\epsilon_{ij} \\ \epsilon_{ij}\epsilon_{kj}(c) &= \epsilon_{ki}(c)\epsilon_{ij}, & \epsilon_{ij}\epsilon_{ki}(c) &= \epsilon_{kj}(c)\epsilon_{ij} \\ \epsilon_{ij}\epsilon_{ij}(c) &= \epsilon_{ji}(c)\epsilon_{ij} \end{aligned} \quad (4-52)$$

In other words, the subscripts of  $\hat{\epsilon}$  are obtained from the subscripts of  $\epsilon$  by making interchanges (possibly none) of the subscripts  $j \leftrightarrow i$ .

Now consider a typical  $P_k$ , say  $P_2$ , in Eq. (4-51). Its corresponding row operation is to interchange the second row with some later row, say the fourth row, so that a nonzero element is brought to the (2, 2) position. Then  $P_2 = \epsilon_{24}$ . Applying the rules of Eq. (4-52) to Eq. (4-51), we have

$$\begin{aligned} P_2\epsilon_{14}(c_3)\epsilon_{13}(c_2)\epsilon_{12}(c_1) &= \epsilon_{24}\epsilon_{14}(c_3)\epsilon_{13}(c_2)\epsilon_{12}(c_1) = \epsilon_{12}(c_3)\epsilon_{24}\epsilon_{13}(c_2)\epsilon_{12}(c_1) \\ &= \epsilon_{12}(c_3)\epsilon_{13}(c_2)\epsilon_{24}\epsilon_{12}(c_1) = \epsilon_{12}(c_3)\epsilon_{13}(c_2)\epsilon_{14}(c_1)\epsilon_{24} \\ &= \epsilon_{12}(c_3)\epsilon_{13}(c_2)\epsilon_{14}(c_1)P_2 \end{aligned}$$

Thus, we have moved  $P_2$  successively to the rightmost position. Although the contents of the other  $\epsilon$  matrices change, note that they all remain lower triangular matrices!

From our discussion we see that, after moving all  $P_k$ 's to the right, Eq. (4-49) may be written as

$$(\hat{\epsilon}_m \dots \hat{\epsilon}_2 \hat{\epsilon}_1)(P_{n-1} \dots P_2 P_1)A = U \quad (4-53)$$

where all  $\hat{\epsilon}_k$  are lower triangular matrices. Equation (4-53) directly leads to

$$LU = PA$$

where

$$P = P_{n-1}P_{n-2} \dots P_2P_1$$

is a permutation matrix, and

$$L = (\hat{\epsilon}_m \dots \hat{\epsilon}_2 \hat{\epsilon}_1)^{-1}$$

is a lower triangular matrix. We have, in fact, proved Theorem 4-1.

**Theorem 4-1.** Given any nonsingular matrix  $A$ , there exists some permutation matrix  $P$  (possibly  $P = 1$ ) such that

$$\boxed{LU = PA} \quad (4-54)$$

where  $U$  is an upper triangular matrix with unit diagonal elements and  $L$  is a

lower triangular matrix with nonzero diagonal elements. Once a proper  $P$  is chosen, this factorization is unique.

The factorization of  $A$  or  $PA$  into the product  $LU$  is called *LU factorization*. By inserting the product of two diagonal matrices  $D_1$  and  $D_2$ , with  $D_1D_2 = 1$ , between  $L$  and  $U$ , the theorem can take on many different forms, as follows:

$$LU = LD_1D_2U = (LD_1)(D_2U) = \hat{L}\hat{U} = \hat{L}D_2U = LD_1\hat{U} \quad (4-55)$$

where  $\hat{L}$  and  $\hat{U}$  are lower and upper triangular matrices, respectively, one of which may have arbitrary nonzero diagonal elements.

Once the  $LU$  factorization is obtained, by whatever method, the equation

$$Ax = LUx = \mu \quad (4-56)$$

is solved by transforming Eq. (4-56) into

$$Ly = \mu \quad (4-57)$$

and

$$Ux = y \quad (4-58)$$

We first solve for  $y$  from Eq. (4-57) and next solve for  $x$  from Eq. (4-58). Since both equations are triangular systems of equations, the solutions are easily obtained by back substitutions.



#### 4-4-2 Crout's Algorithm without Row Interchange

We shall now describe a practical method for calculating the  $L$  and  $U$  factors, which is sometimes referred to as Crout's method [6].

In Gaussian elimination, the forward-elimination step yields  $U$  explicitly.<sup>10</sup> The elements of  $L$  can be shown to be given by

$$L = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22}^{(2)} & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2}^{(2)} & & a_{nn}^{(n)} \end{bmatrix} \quad (4-59)$$

Each element of  $L$  appears at a certain stage of the forward-elimination step and may be recorded (usually written over the original matrix to save storage).

Note that in Gaussian elimination many elements of the given matrix are written over quite a number of times before an element in the  $U$  matrix is obtained. For example, in Eq. (4-23), the element  $a_{44}$  is written over three times:

$$a_{44}, a_{44}^{(2)}, a_{44}^{(3)}, a_{44}^{(4)}$$

The last result is the (4, 4) element of  $L$  matrix.

<sup>10</sup>To obtain  $U$  with unit diagonal elements, further divide the  $k$ th equation in Eq. (4-30) by  $a_{kk}^{(k)}$ .

The Crout's method, as will be shown presently, is actually a method of calculating the elements of  $L$  and  $U$  recursively, without writing over previous results. This is a great advantage with desk calculators. With automatic digital computers, this advantage is not an important one. As far as the number of operations is concerned, both Gaussian elimination and Crout's method require approximately  $n^3/3$  operations for  $n$  large. The real advantage of Crout's method lies in smaller round-off error and in its suitability to incorporate sparse-matrix techniques [7]-[8] (see Chap. 16).

To see how Crout's method generates the elements of  $L$  and  $U$  recursively, again consider the case  $n = 4$ , as in Section 4-3. We assume that no row interchange is necessary in Eq. (4-54). Then  $P = 1$  and

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad (4-60)$$

We shall now show how each element  $l_{jk}$  of  $L$  and each element  $u_{jk}$  of  $U$  can be determined using the Crout's algorithm.

Define an auxiliary matrix, consisting of elements of  $L$  and  $U$  in the following manner:

$$Q \triangleq \begin{bmatrix} l_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & l_{22} & u_{23} & u_{24} \\ l_{31} & l_{32} & l_{33} & u_{34} \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \quad (4-61)$$

Elements of  $Q$  are to be calculated one by one in the illustrated order

$$\begin{bmatrix} \textcircled{1} & \textcircled{5} & \textcircled{6} & \textcircled{7} \\ \textcircled{2} & \textcircled{8} & \textcircled{11} & \textcircled{12} \\ \textcircled{3} & \textcircled{9} & \textcircled{13} & \textcircled{15} \\ \textcircled{4} & \textcircled{10} & \textcircled{14} & \textcircled{16} \end{bmatrix} \quad (4-62)$$

where  $\textcircled{k}$  indicates the  $k$ th element to be calculated. In other words, we calculate

First column of  $L$

First row of  $U$  (except  $u_{11} = 1$ )

Second column of  $L$

Second row of  $U$  (except  $u_{22} = 1$ )

etc.

The elements of  $L$  and  $U$  are calculated simply by equating  $a_{jk}$  successively, according to the order shown in Eq. (4-62), with the product of the  $j$ th row of  $L$  and the  $k$ th column of  $U$ . We have

$$\begin{aligned} a_{11} &= l_{11} \cdot 1 & \therefore l_{11} &= a_{11} \\ a_{21} &= l_{21} \cdot 1 & \therefore l_{21} &= a_{21} \\ a_{31} &= l_{31} \cdot 1 & \therefore l_{31} &= a_{31} \\ a_{41} &= l_{41} \cdot 1 & \therefore l_{41} &= a_{41} \\ a_{12} &= l_{11}u_{12} & \therefore u_{12} &= \frac{a_{12}}{l_{11}} \\ a_{13} &= l_{11}u_{13} & \therefore u_{13} &= \frac{a_{13}}{l_{11}} \\ a_{14} &= l_{11}u_{14} & \therefore u_{14} &= \frac{a_{14}}{l_{11}} \\ a_{22} &= l_{21}u_{12} + l_{22} & \therefore l_{22} &= a_{22} - l_{21}u_{12} \\ a_{32} &= l_{31}u_{12} + l_{32} & \therefore l_{32} &= a_{32} - l_{31}u_{12} \\ a_{42} &= l_{41}u_{12} + l_{42} & \therefore l_{42} &= a_{42} - l_{41}u_{12} \\ a_{23} &= l_{21}u_{13} + l_{22}u_{23} & \therefore u_{23} &= \frac{a_{23} - l_{21}u_{13}}{l_{22}} \\ a_{24} &= l_{21}u_{14} + l_{22}u_{24} & \therefore u_{24} &= \frac{a_{24} - l_{21}u_{14}}{l_{22}} \\ a_{33} &= l_{31}u_{13} + l_{32}u_{23} + l_{33} & \therefore l_{33} &= a_{33} - l_{31}u_{13} - l_{32}u_{23} \\ a_{43} &= l_{41}u_{13} + l_{42}u_{23} + l_{43} & \therefore l_{43} &= a_{43} - l_{41}u_{13} - l_{42}u_{23} \\ a_{34} &= l_{31}u_{14} + l_{32}u_{24} + l_{33}u_{34} & \therefore u_{34} &= \frac{a_{34} - l_{31}u_{14} - l_{32}u_{24}}{l_{33}} \\ a_{44} &= l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + l_{44} & \therefore l_{44} &= a_{44} - l_{41}u_{14} - l_{42}u_{24} - l_{43}u_{34} \end{aligned} \quad (4-63)$$

Note a very important feature of this calculation: the calculation of the element  $\textcircled{k}$  of the auxiliary matrix  $Q$ , which is an element of either  $L$  or  $U$ , involves only the element of  $A$  in the same position and some of those elements of  $Q$  that have already been calculated. As the element  $\textcircled{k}$  is obtained, it is recorded in the  $Q$  matrix. (In fact, it may be recorded in the corresponding position in the  $A$  matrix, if there is no need to keep the  $A$  matrix.) This calculated result need never be written over; it is already one of the elements of  $L$  or  $U$ .

Following the pattern displayed by Eq. (4-63), we can formulate Crout's algorithm for obtaining the auxiliary matrix  $Q$  (which contains the components  $L$  and  $U$ ) as given below. A rigorous proof of the algorithm may be found in [6].

**Crout's Algorithm without Row Interchange**

Given  $A$ , an  $n \times n$  matrix. Initially,  $Q = 0$  of dimension  $n \times n$ .

- Step 1. Obtain the first column of  $Q$  by  $q_{k1} = a_{k1}$ , for  $k = 1, 2, \dots, n$ .  
 Step 2. Complete the first row of  $Q$  by  $q_{1k} = a_{1k}/q_{11}$ , for  $k = 2, 3, \dots, n$ .  
 Step 3. Set  $j \leftarrow 2$ .  
 Step 4. Complete the entries of the  $j$ th column of  $Q$  by  $q_{kj} = a_{kj} - q_{k1}q_{1j} - q_{k2}q_{2j} - \dots - q_{k(j-1)}q_{(j-1)j}$  for  $k = j, j+1, \dots, n$ .  
 Step 5. If  $j = n$ , terminate the algorithm. Otherwise go to step 6.  
 Step 6. Complete the entries of the  $j$ th row of  $Q$  by  $q_{jk} = (a_{jk} - q_{j1}q_{1k} - q_{j2}q_{2k} - \dots - q_{j(j-1)}q_{(j-1)k})/q_{jj}$  for  $k = j+1, j+2, \dots, n$ .  
 Step 7. Set  $j \leftarrow j+1$ ; go to step 4.

A numerical example will illustrate the application of the algorithm.

**EXAMPLE 4-2.**

$$A = \begin{bmatrix} 2 & 4 & 4 & 2 \\ 3 & 3 & 12 & 6 \\ 2 & 4 & -1 & 2 \\ 4 & 2 & 1 & 1 \end{bmatrix}$$

Following the algorithm, we have

$$Q = \begin{bmatrix} 2 & 2 & 2 & 1 \\ 3 & -3 & -2 & -1 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -19 & -9 \end{bmatrix}, \quad L = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & -3 & 0 & 0 \\ 2 & 0 & -5 & 0 \\ 4 & -6 & -19 & -9 \end{bmatrix},$$

$$U = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 0 & 1 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Elements of  $Q$  are calculated as follows:

First column:  $2 = 2, 3 = 3, 2 = 2, 4 = 4$ .

First row:  $2 = \frac{4}{2}, 2 = \frac{4}{2}, 1 = \frac{2}{2}$ .

Second column:  $-3 = 3 - 3 \times 2, 0 = 4 - 2 \times 2, -6 = 2 - 4 \times 2$ .

Second row:  $-2 = (12 - 3 \times 2)/(-3), -1 = (6 - 3 \times 1)/(-3)$ .

Third column:  $-5 = -1 - 2 \times 2 - 0 \times (-2), -19 = 1 - 4 \times 2 - (-6) \times (-2)$ .

Third row:  $0 = 2 - 2 \times 1 - 0 \times (-1)$ .

Fourth column:  $-9 = 1 - 4 \times 1 - (-6) \times (-1) - (-19) \times 0$ .

**4-5 SINUSOIDAL STEADY-STATE ANALYSIS OF LINEAR NETWORKS BY NODAL EQUATIONS**

The analysis of the behavior of a linear network in sinusoidal steady state is usually referred to as "ac analysis." A great deal of similarity exists between ac analysis and the analysis of linear resistive networks in Section 4-2. In fact, all the relationships discussed there carry over to ac analysis if we make the following transitions in notations and concepts.

ANALYSIS OF LINEAR RESISTIVE NETWORK	AC ANALYSIS
Resistance $R$	Impedance $Z$
Conductance $G$	Admittance $Y$
$v(t)$ , time function	$V(\omega)$ , phasor
$i(t)$ , time function	$I(\omega)$ , phasor
Values of $R, G, v, i$ are real numbers	Values of $Z, Y, V, I$ are complex numbers

Because of such correspondence, we shall not repeat the detailed derivation. Instead, we shall only give the definitions and the main results, and the reader should refer to Section 4-2 for details.

Define the voltage vectors as

$$\hat{V} \triangleq \begin{bmatrix} \hat{V}_1 \\ \hat{V}_2 \\ \vdots \\ \hat{V}_b \end{bmatrix}, \quad V \triangleq \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_b \end{bmatrix}, \quad E \triangleq \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_b \end{bmatrix} \quad (4-64)$$

and the current vectors as

$$\hat{I} \triangleq \begin{bmatrix} \hat{I}_1 \\ \hat{I}_2 \\ \vdots \\ \hat{I}_b \end{bmatrix}, \quad I \triangleq \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_b \end{bmatrix}, \quad J \triangleq \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_b \end{bmatrix} \quad (4-65)$$