4 COMPUTER PROGRAM FOR ANALYSIS OF PLANE TRUSSES

Truss Bridge
(Capricornis Photographic Inc. / Shutterstock)

In the previous chapter, we studied the basic principles of the analysis of plane trusses by the matrix stiffness method. In this chapter, we consider the computer implementation of the foregoing method of analysis. Our objective is to develop a general computer program that can be used to analyze any statically determinate or indeterminate plane truss, of any arbitrary configuration, subjected to any system of joint loads.

From a programming viewpoint, it is generally convenient to divide a structural analysis program into two parts or modules: (a) *input module*, and (b) *analysis module* (Fig. 4.1). The input module reads, and stores into the computer's memory, the structural and loading data necessary for the analysis; the analysis module uses the input data to perform the analysis, and communicates the results back to the user via an output device, such as a printer or a monitor. The development of a relatively simple input module is presented in Section 4.1; in the following five sections (4.2 through 4.6), we consider programming of the five analysis steps discussed in Chapter 3 (Fig. 3.20). The topics covered in these sections are as follows: assignment of the degree-of-freedom and restrained coordinate numbers for plane trusses (Section 4.2); generation of the structure stiffness matrix by assembling the elements of the member stiffness matrices (Section 4.3); formation of the joint load vector (Section 4.4); solution of the structure stiffness equations to obtain joint displacements (Section 4.5); and, finally, evaluation of the member axial forces and support reactions (Section 4.6).

The entire programming process is described by means of detailed flowcharts, so that readers can write this computer program in any programming language. It is important to realize that the programming process presented in this chapter represents only one of many ways in which the matrix stiffness method of analysis can be implemented on computers. Readers are strongly encouraged to conceive, and attempt, alternative strategies that can make the computer implementation (and/or application of the method) more efficient. One such strategy, which takes advantage of the banded form of the structure stiffness matrix, will be discussed in Chapter 9.

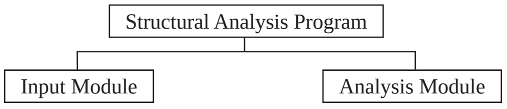


Fig. 4.1

4.1 DATA INPUT

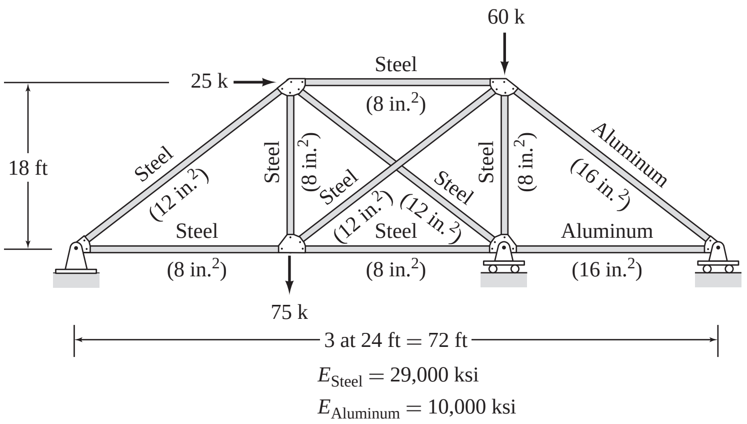
In this section, we focus our attention on the input module of our computer program. As stated previously, *the input module of a structural analysis program reads the structural and loading data necessary for analysis from a file or another type of input device, and stores it in the computer's memory so that it can be processed conveniently by the program for structural analysis.*

When structural analysis is carried out by hand calculations (e.g., as in Chapter 3), the information needed for the analysis is obtained by visually inspecting the analytical model of the structure (as represented by the line diagram). In computerized structural analysis, however, all of the data necessary for analysis must be specified in the form of numbers, and must be organized in the computer's memory in the form of matrices (arrays), in such a way that it can be used for analysis without any reference to a visual image (or line diagram) of the structure. This data in numerical form must completely and uniquely define the analytical model of the structure. In other words, a person with no knowledge of the actual structure or its analytical model should be able to reconstruct the visual analytical model of the structure, using only the numerical data and the knowledge of how this data is organized.

The input data necessary for the analysis of plane trusses can be divided into the following six categories:

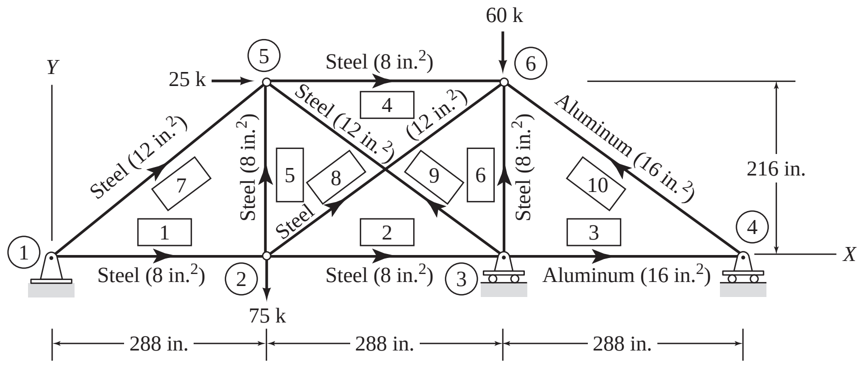
- joint data
- support data
- material property data
- cross-sectional property data
- member data
- load data

In the following, we discuss procedures for inputting data belonging to each of the foregoing categories, using the truss of Fig. 4.2(a) as an example. The analytical model of this truss is depicted in Fig. 4.2(b). Note that all the information in this figure is given in units of kips and inches. This is because we plan to design a computer program that can work with any *consistent* set of units. Thus, all the data must be converted into a consistent set of units before being input into the program.



(a) Actual Truss

Fig. 4.2



$E_{\text{Steel}} = 29{,}000$ ksi

$E_{\text{Aluminum}} = 10{,}000$ ksi

(b) Analytical Model

$$\mathbf{COORD} = \begin{array}{cc|l} X\text{ coordinate} & Y\text{ coordinate} & \\ \hline 0 & 0 & \leftarrow \text{Joint 1} \\ 288 & 0 & \leftarrow \text{Joint 2} \\ 576 & 0 & \leftarrow \text{Joint 3} \\ 864 & 0 & \leftarrow \text{Joint 4} \\ 288 & 216 & \leftarrow \text{Joint 5} \\ 576 & 216 & \leftarrow \text{Joint 6} \end{array}$$

$NJ \times 2$

(c) Joint Coordinate Matrix

$$\mathbf{CP} = \begin{array}{c|l} 8 & \leftarrow \text{Cross-section type no. 1} \\ 12 & \leftarrow \text{Cross-section type no. 2} \\ 16 & \leftarrow \text{Cross-section type no. 3} \end{array}$$

$NCP \times 1$

(f) Cross-sectional Property Vector

$$\mathbf{MPRP} = \begin{array}{cccc|l} \text{Beginning joint} & \text{End joint} & \text{Material no.} & \text{Cross-section type no.} & \\ \hline 1 & 2 & 1 & 1 & \leftarrow \text{Member 1} \\ 2 & 3 & 1 & 1 & \leftarrow \text{Member 2} \\ 3 & 4 & 2 & 3 & \leftarrow \text{Member 3} \\ 5 & 6 & 1 & 1 & \leftarrow \text{Member 4} \\ 2 & 5 & 1 & 1 & \leftarrow \text{Member 5} \\ 3 & 6 & 1 & 1 & \leftarrow \text{Member 6} \\ 1 & 5 & 1 & 2 & \leftarrow \text{Member 7} \\ 2 & 6 & 1 & 2 & \leftarrow \text{Member 8} \\ 3 & 5 & 1 & 2 & \leftarrow \text{Member 9} \\ 4 & 6 & 2 & 3 & \leftarrow \text{Member 10} \end{array}$$

$NM \times 4$

(g) Member Data Matrix

$$\mathbf{MSUP} = \begin{array}{ccc} \text{Joint number} & \begin{array}{c}\text{Restraint in } X \text{ direction}\\ (0 = \text{free, } 1 = \text{restrained})\end{array} & \begin{array}{c}\text{Restraint in } Y \text{ direction}\\ (0 = \text{free, } 1 = \text{restrained})\end{array} \\ \hline 1 & 1 & 1 \\ 3 & 0 & 1 \\ 4 & 0 & 1 \end{array}$$

$NS \times (NCJT + 1)$

(d) Support Data Matrix

$$\mathbf{EM} = \begin{array}{c|l} 29000 & \leftarrow \text{Material no. 1} \\ 10000 & \leftarrow \text{Material no. 2} \end{array}$$

$NMP \times 1$

(e) Elastic Modulus Vector

$$\mathbf{JP} = \begin{array}{c} \text{Joint number} \\ \hline 2 \\ 5 \\ 6 \end{array} \qquad \mathbf{PJ} = \begin{array}{cc} \text{Force in } X \text{ direction} & \text{Force in } Y \text{ direction} \\ \hline 0 & -75 \\ 25 & 0 \\ 0 & -60 \end{array}$$

$NJL \times 1$ $\qquad NJL \times NCJT$

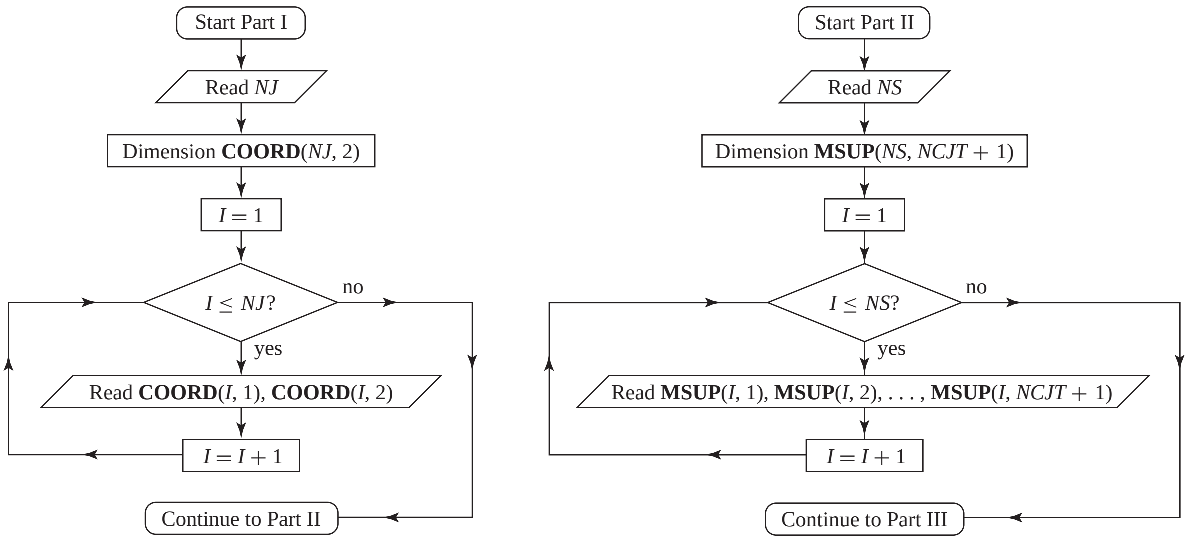
(h) Load Data Matrices

Fig. 4.2 (*continued*)

Joint Data

The joint data consists of: (a) the total number of joints (*NJ*) of the truss, and (b) the global (*X* and *Y*) coordinates of each joint. The relative positions of the joints of the truss are specified by means of the global (*X* and *Y*) coordinates of the joints. These joint coordinates are usually stored in the computer's memory in the form of a matrix, so that they can be accessed easily by the computer program for analysis. In our program, we store the joint coordinates in a matrix **COORD** of the order $NJ \times 2$ (Fig. 4.2(c)). The matrix, which is referred to as the *joint coordinate matrix*, has two columns, and its number of rows equals the total number of joints (*NJ*) of the structure. The *X* and *Y* coordinates of a joint *i* are stored in the first and second columns, respectively, of the *i*th row of the matrix **COORD**. Thus, for the truss of Fig. 4.2(b) (which has six joints), the joint coordinate matrix is a 6×2 matrix, as shown in Fig. 4.2(c). Note that the joint coordinates are stored in the sequential order of joint numbers. Thus, by comparing Figs. 4.2(b) and (c), we can see that the *X* and *Y* coordinates of joint 1 (i.e., 0 and 0) are stored in the first and second columns, respectively, of the first row of **COORD**. Similarly, the *X* and *Y* coordinates of joint 5 (288 and 216) are stored in the first and second columns, respectively, of the fifth row of **COORD**, and so on.

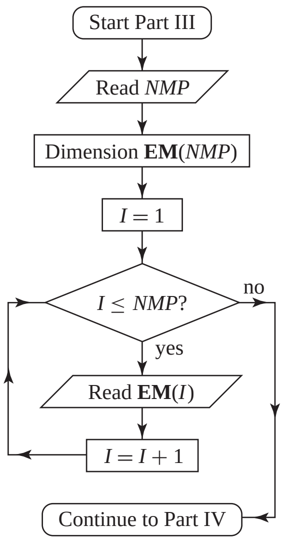
A flowchart for programming the reading and storing of the joint data for plane trusses is given in Fig. 4.3(a). As shown there, the program first reads the value of the integer variable *NJ*, which represents the total number of joints of the truss. Then, using a *Do Loop* command, the *X* and *Y* coordinates of each joint are read, and stored in the first and second columns, respectively, of the matrix **COORD**. The *Do Loop* starts with joint number 1 and ends with joint


Start Part I
Read NJ
Dimension **COORD**(NJ, 2)
$I = 1$
$I \le NJ$?
no
yes
Read **COORD**(I, 1), **COORD**(I, 2)
$I = I + 1$
Continue to Part II
Start Part II
Read NS
Dimension **MSUP**(NS, $NCJT + 1$)
$I = 1$
$I \le NS$?
no
yes
Read **MSUP**(I, 1), **MSUP**(I, 2), . . . , **MSUP**(I, $NCJT + 1$)
$I = I + 1$
Continue to Part III

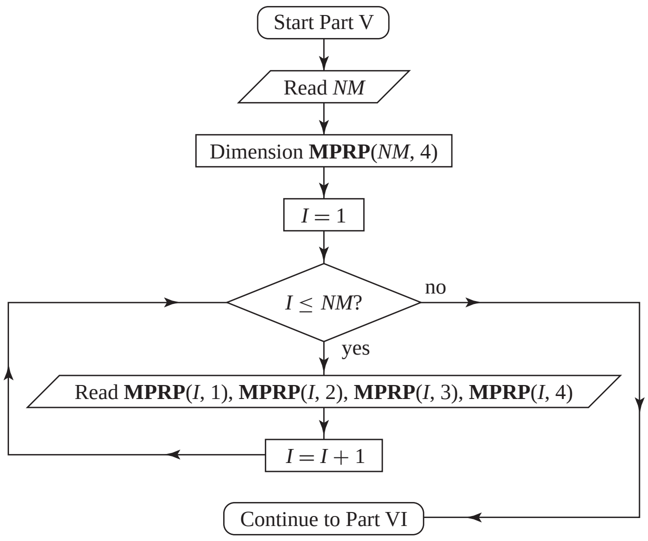
(a) Flowchart for Reading and Storing Joint Data

(b) Flowchart for Reading and Storing Support Data

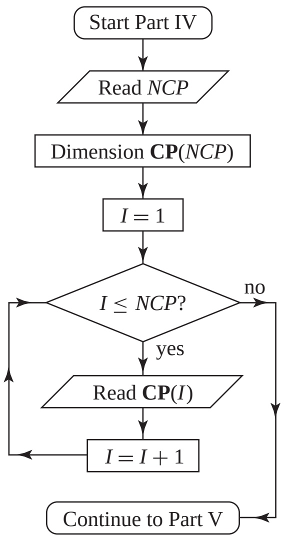
Fig. 4.3



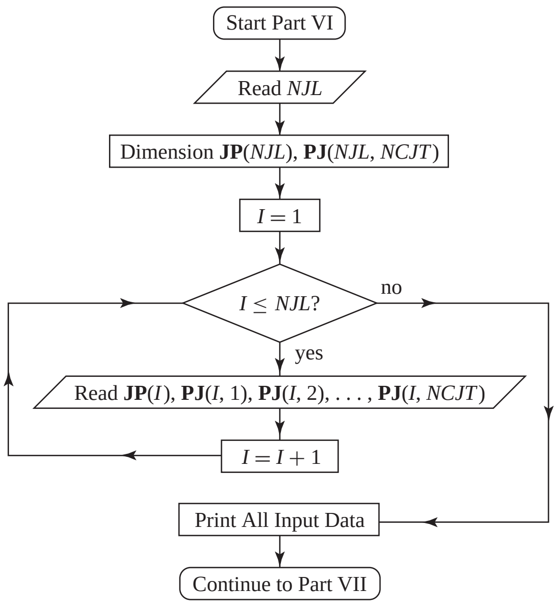
(c) Flowchart for Reading and Storing Material Property Data


Start Part V
Read *NM*
Dimension **MPRP**(*NM*, 4)
I = 1
I ≤ *NM*?
no
yes
Read **MPRP**(*I*, 1), **MPRP**(*I*, 2), **MPRP**(*I*, 3), **MPRP**(*I*, 4)
I = *I* + 1
Continue to Part VI

(e) Flowchart for Reading and Storing Member Data



(d) Flowchart for Reading and Storing Cross-sectional Property Data

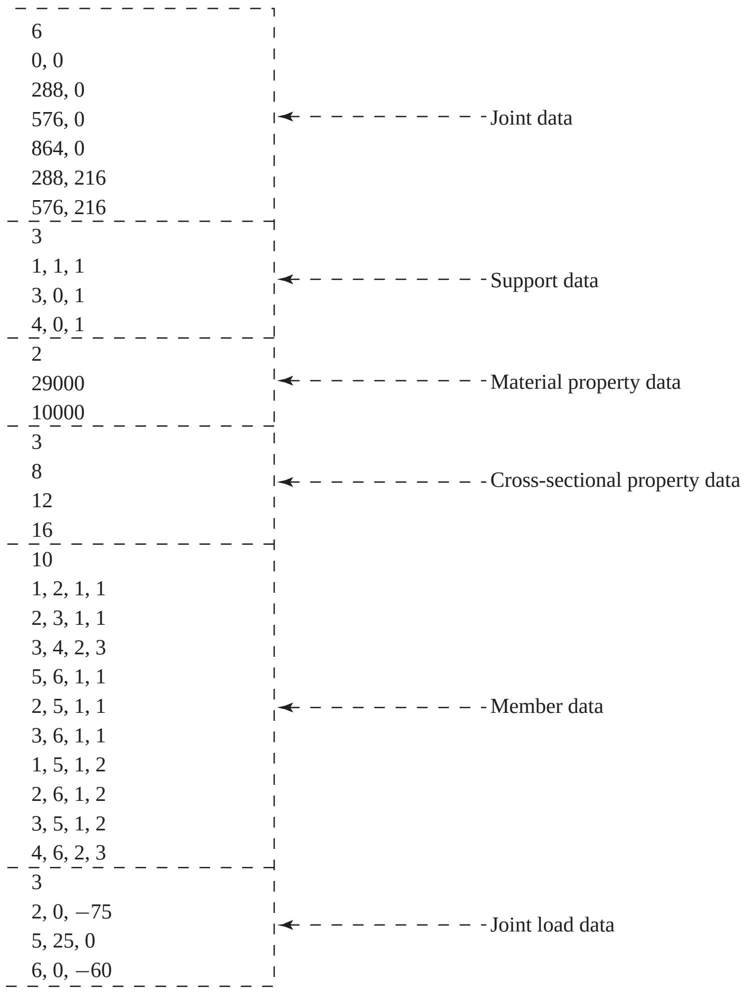

Start Part VI
Read *NJL*
Dimension **JP**(*NJL*), **PJ**(*NJL*, *NCJT*)
I = 1
I ≤ *NJL*?
no
yes
Read **JP**(*I*), **PJ**(*I*, 1), **PJ**(*I*, 2), . . . , **PJ**(*I*, *NCJT*)
I = *I* + 1
Print All Input Data
Continue to Part VII

(f) Flowchart for Reading and Storing Load Data

Fig. 4.3 (*continued*)

number *NJ*. It should be noted that, depending upon the type of programming language and/or compiler being used, some additional statements (such as variable type declaration and formatted read/write statements) may be needed to implement the foregoing program. (It is assumed herein that the reader has a working knowledge of a programming language.)

The input data to be read by the computer program is either entered interactively by the user (responding to prompts on the screen), or is supplied in the form of a data file. The former approach is used in the computer software which can be downloaded from the publisher's website for this book. However, the latter approach is recommended for beginning programmers, because it is straightforward and requires significantly less programming. As an example, the input data file (in free-format) for the truss of Fig. 4.2(b) is given in Fig. 4.4. Note that the first line



```
6
0, 0
288, 0
576, 0
864, 0
288, 216
576, 216
3
1, 1, 1
3, 0, 1
4, 0, 1
2
29000
10000
3
8
12
16
10
1, 2, 1, 1
2, 3, 1, 1
3, 4, 2, 3
5, 6, 1, 1
2, 5, 1, 1
3, 6, 1, 1
1, 5, 1, 2
2, 6, 1, 2
3, 5, 1, 2
4, 6, 2, 3
3
2, 0, −75
5, 25, 0
6, 0, −60
```

Joint data
Support data
Material property data
Cross-sectional property data
Member data
Joint load data

Fig. 4.4 *An Example of an Input Data File*

of this data file contains the total number of joints of the truss (i.e., 6); the next six lines contain the X and Y coordinates of joints 1 through 6, respectively.

Support Data

The support data consists of (a) the number of joints that are attached to supports (NS); and (b) the joint number, and the directions of restraints, for each support joint. Since there can be at most two restrained coordinates at a joint of a plane truss (i.e., $NCJT = 2$), the restraints at a support joint of such a structure can be conveniently specified by using a two-digit code in which each digit is either a 0 or a 1. The first digit of the code represents the restraint condition at the joint in the global X direction; it is 0 if the joint is free to translate in the X direction, or it is 1 if the joint is restrained in the X direction. Similarly, the second digit of the code represents the restraint condition at the joint in the global Y direction; a 0 indicates that the joint is unrestrained in the Y direction, and a 1 indicates that it is restrained. The restraint codes for the various types of supports for plane trusses are given in Fig. 4.5. (The special case of inclined roller supports will be considered in Chapter 9.)

Considering again the example truss of Fig. 4.2(b), we can see that joint 1 is attached to a hinged support that prevents it from translating in any direction.

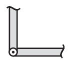
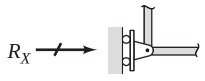
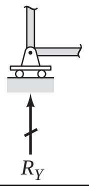
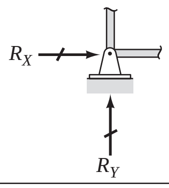
Type of Support		Restraint Code
Free joint (no support)		0, 0
Roller with horizontal reaction		1, 0
Roller with vertical reaction		0, 1
Hinge		1, 1

Fig. 4.5 *Restraint Codes for Plane Trusses*

Thus, the restraint code for joint 1 is 1,1 indicating that this joint is restrained from translating in both the X and Y directions. Similarly, the restraint codes for joints 3 and 4, which are attached to roller supports, are 0,1 because these joints are free to translate in the horizontal (X) direction, but are restrained by the rollers from translating in the vertical (Y) direction. The restraint codes of the remaining joints of the truss, which are free to translate in any direction, can be considered to be 0,0. However, it is not necessary to input codes for free joints, because the computer program considers every joint to be free, unless it is identified as a support joint.

The support data can be stored in the computer's memory in the form of an integer matrix **MSUP** of order $NS \times (NCJT + 1)$ (Fig. 4.2(d)). For plane trusses, because $NCJT = 2$, the *support data matrix* **MSUP** consists of three columns, with the number of rows equal to the number of support joints (NS). In each row of **MSUP**, the support joint number is stored in the first column, and the first and second digits of the corresponding restraint code are stored in the second and third columns, respectively. Thus, for the truss of Fig. 4.2(b), which has three support joints, the support data matrix is a 3×3 matrix, as shown in Fig. 4.2(d). Note that in the first row of **MSUP** the support joint number 1 is stored in the first column, and the first and second digits of the restraint code for this joint (i.e., 1 and 1) are stored in the second and third columns, respectively. Similarly, the second row of **MSUP** consists of the support joint number 3 in the first column, and the two digits of the corresponding restraint code (i.e., 0 and 1) in the second and third columns, respectively, and so on.

A flowchart for programming the reading and storing of the support data is given in Fig. 4.3(b), in which, as noted previously, the integer variable $NCJT$ denotes the number of structure coordinates per joint. Like this flowchart, many parts of the computer program presented in this chapter are given in a general form in terms of the variable $NCJT$, so that they can be conveniently incorporated into computer programs for analyzing other types of framed structures (e.g., beams and plane frames), which are considered in subsequent chapters. For example, as discussed in this section, by setting $NCJT = 2$, the flowchart of Fig. 4.3(b) can be used to input support data for plane trusses; whereas, as discussed subsequently in Chapter 6, the same flowchart can be used to input support data for plane frames, provided $NCJT$ is set equal to three.

An example of how the support data for a plane truss may appear in an input data file is given in Fig. 4.4.

Material Property Data

The material property data involves (a) the number of materials used in the structure (NMP), and (b) the modulus of elasticity (E) of each material. The elastic moduli are stored by the program in an *elastic modulus vector* **EM**. The number of rows of **EM** equals the number of materials (NMP), with the elastic modulus of material i stored in the ith row of the vector (Fig. 4.2(e)).

Consider, for example, the truss of Fig. 4.2(b). The truss is composed of two materials; namely, steel and aluminum. We arbitrarily select the steel ($E = 29{,}000$ ksi) to be material number 1, and the aluminum ($E = 10{,}000$ ksi) to be material number 2. Thus, the elastic modulus vector, **EM**, of the truss consists of two rows, as shown in Fig. 4.2(e); the elastic modulus of material number 1 (i.e., 29,000) is stored in the first row of **EM**, and the elastic modulus of material number 2 (i.e., 10,000) is stored in the second row.

Figure 4.3(c) shows a flowchart for programming the reading and storing of the material property data; Fig. 4.4 illustrates how this type of data may appear in an input data file.

Cross-Sectional Property Data

The cross-sectional property data consists of (a) the number of different cross-section types used for the truss members (NCP); and (b) the cross-sectional area (A) for each cross-section type. The cross-sectional areas are stored by the program in a *cross-sectional property vector* **CP**. The number of rows of **CP** equals the number of cross-section types (NCP), with the area of cross-section i stored in the ith row of the vector (Fig. 4.2(f)).

For example, three types of member cross-sections are used for the truss of Fig. 4.2(b). We arbitrarily assign the numbers 1, 2, and 3 to the cross-sections with areas of 8, 12, and 16 in.^{2}, respectively. Thus, the cross-sectional property vector, **CP**, consists of three rows; areas of cross-section types 1, 2, and 3 are stored in rows 1, 2, and 3, respectively, as shown in Fig. 4.2(f).

A flowchart for reading and storing the cross-sectional property data into computer memory is given in Fig. 4.3(d); Fig. 4.4 shows an example of an input data file containing this type of data.

Member Data

The member data consists of (a) the total number of members (NM) of the truss; and (b) for each member, the beginning joint number, the end joint number, the material number, and the cross-section type number.

The member data can be stored in computer memory in the form of an integer *member data matrix*, **MPRP**, of order $NM \times 4$ (Fig. 4.2(g)). The information corresponding to a member i is stored in the ith row of **MPRP**; its beginning and end joint numbers are stored in the first and second columns, respectively, and the material and cross-section numbers are stored in the third and fourth columns, respectively.

For example, since the truss of Fig. 4.2(b) has 10 members, its member data matrix is a 10×4 matrix, as shown in Fig. 4.2(g). From Fig. 4.2(b), we can see that the beginning and end joints for member 1 are 1 and 2, respectively; the material and cross-section numbers for this member are 1 and 1, respectively. Thus, the numbers 1, 2, 1, and 1 are stored in columns 1 through 4, respectively, of the first row of **MPRP**, as shown in Fig. 4.2(g). Similarly, we see from Fig. 4.2(b) that the beginning joint, end joint, material, and cross-section numbers for member 3 are 3, 4, 2, and 3, respectively, and they are stored, respectively, in columns 1 through 4 of row 3 of **MPRP**, and so on.

Figure 4.3(e) shows a flowchart for programming the reading and storing of the member data. An example of how member data may appear in an input data file is given in Fig. 4.4.

Load Data

The load data involves (a) the number of joints that are subjected to external loads (*NJL*); and (b) the joint number, and the magnitudes of the force components in the global *X* and *Y* directions, for each loaded joint. The numbers of the loaded joints are stored in an integer vector **JP** of order $NJL \times 1$; the corresponding load components in the *X* and *Y* directions are stored in the first and second columns, respectively, of a real matrix **PJ** of order $NJL \times NCJT$, with $NCJT = 2$ for plane trusses (see Fig. 4.2(h)). Thus, for the example truss of Fig. 4.2(a), which has three joints (2, 5, and 6) that are subjected to loads, the *load data matrices*, **JP** and **PJ**, are of orders 3×1 and 3×2, respectively, as shown in Fig. 4.2(h). The first row of **JP** contains joint number 2; the loads in the *X* and *Y* directions at this joint (i.e., 0 and −75 k) are stored in the first and second columns, respectively, of the same row of **PJ**. The information about joints 5 and 6 is then stored in a similar manner in the second and third rows, respectively, of **JP** and **PJ**, as shown in the figure.

A flowchart for programming the reading and storing of the load data is given in Fig. 4.3(f), in which *NCJT* must be set equal to 2 for plane trusses. Figure 4.4 shows the load data for the example truss in an input file.

It is important to recognize that the numerical data stored in the various matrices in Figs. 4.2(c) through (h) *completely* and *uniquely* defines the analytical model of the example truss, without any need to refer to the line diagram of the structure (Fig. 4.2(b)).

After all the input data has been read and stored in computer memory, it is considered a good practice to print this data directly from the matrices in the computer memory (or view it on the screen), so that its validity can be verified (Fig. 4.3(f)). An example of such a printout, showing the input data for the example truss of Fig. 4.2, is given in Fig. 4.6.

```
          **********************************
          *       Computer Software        *
          *              for               *
          *  MATRIX ANALYSIS OF STRUCTURES *
          *         Second Edition         *
          *               by               *
          *        Aslam Kassimali         *
          **********************************

               General Structural Data

Project Title: Figure 4-2
Structure Type : Plane Truss
Number of Joints : 6
Number of Members : 10
Number of Material Property Sets (E) : 2
Number of Cross-Sectional Property Sets : 3
```

Fig. 4.6 *A Sample Printout of Input Data*

```
                      Joint Coordinates

  Joint No.           X Coordinate          Y Coordinate
      1                0.0000E+00            0.0000E+00
      2                2.8800E+02            0.0000E+00
      3                5.7600E+02            0.0000E+00
      4                8.6400E+02            0.0000E+00
      5                2.8800E+02            2.1600E+02
      6                5.7600E+02            2.1600E+02
                           Supports

   Joint No.          X Restraint          Y Restraint
      1                   Yes                  Yes
      3                   No                   Yes
      4                   No                   Yes
                     Material Properties

  Material          Modulus of           Co-efficient of
     No.          Elasticity (E)       Thermal Expansion
      1             2.9000E+04            0.0000E+00
      2             1.0000E+04            0.0000E+00
                 Cross-Sectional Properties

                 Property No.        Area (A)
                       1            8.0000E+00
                       2            1.2000E+01
                       3            1.6000E+01
                         Member Data

 Member    Beginning   End   Material   Cross-Sectional
   No.       Joint    Joint     No.      Property No.
   1           1        2       1            1
   2           2        3       1            1
   3           3        4       2            3
   4           5        6       1            1
   5           2        5       1            1
   6           3        6       1            1
   7           1        5       1            2
   8           2        6       1            2
   9           3        5       1            2
  10           4        6       2            3
                         Joint Loads

     Joint No.            X Force        Y Force
         2               0.0000E+00     -7.5000E+01
         5               2.5000E+01      0.0000E+00
         6               0.0000E+00     -6.0000E+01
   ************* End of Input Data *************
```

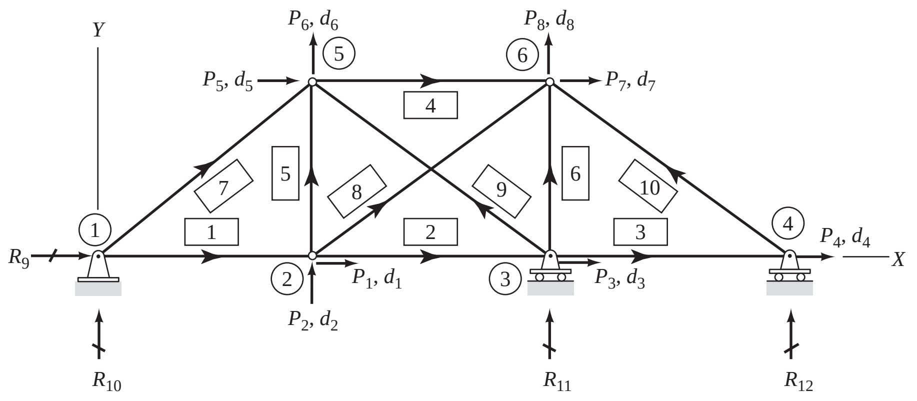
Fig. 4.6 (*continued*)

4.2 ASSIGNMENT OF STRUCTURE COORDINATE NUMBERS

Having completed the input module, we are now ready to develop the analysis module of our computer program. *The analysis module of a structural analysis program uses the input data stored in computer memory to calculate the desired response characteristics of the structure, and communicates these results to the user through an output device, such as a printer or a monitor*.

As discussed in Section 3.8, the first step of the analysis involves specification of the structure's degrees of freedom and restrained coordinates, which are collectively referred to as, simply, the *structure coordinates*. Recall that when the analysis was carried out by hand calculations (in Chapter 3), the structure coordinate numbers were written next to the arrows, in the global X and Y directions, drawn at the joints. In computerized analysis, however, these numbers must be organized in computer memory in the form of a matrix or a vector. In our program, the structure coordinate numbers are stored in an integer vector **NSC**, with the number of rows equal to the number of structure coordinates per joint ($NCJT$) times the number of joints of the structure (NJ). For plane trusses, because $NCJT = 2$, the number of rows of **NSC** equals twice the number of joints of the truss (i.e., $2NJ$). The structure coordinate numbers are arranged in **NSC** in the sequential order of joint numbers, with the number for the X coordinate at a joint followed by the number for its Y coordinate. In other words, the numbers for the X and Y structure coordinates at a joint i are stored in rows $(i - 1)2 + 1$ and $(i - 1)2 + 2$, respectively, of **NSC**. For example, the line diagram of the truss of Fig. 4.2(a) is depicted in Fig. 4.7(a) with its degrees of freedom and restrained coordinates indicated, and the corresponding 12×1 **NSC** vector is given in Fig. 4.7(b).

The procedure for assigning the structure coordinate (i.e., degrees of freedom and restrained coordinate) numbers was discussed in detail in Section 3.2.



(a) Line Diagram Showing Degrees of Freedom and Restrained Coordinates

Fig. 4.7

Structure coordinate number

$$\mathbf{NSC} = \left[\begin{array}{c} 9 \\ 10 \\ \hdashline 1 \\ 2 \\ \hdashline 3 \\ 11 \\ \hdashline 4 \\ 12 \\ \hdashline 5 \\ 6 \\ \hdashline 7 \\ 8 \end{array}\right] \begin{array}{l} \leftarrow X \\ \leftarrow Y \\ \leftarrow X \\ \leftarrow Y \\ \leftarrow X \\ \leftarrow Y \\ \leftarrow X \\ \leftarrow Y \\ \leftarrow X \\ \leftarrow Y \\ \leftarrow X \\ \leftarrow Y \end{array} \begin{array}{l} \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{Joint 1} \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{Joint 2} \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{Joint 3} \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{Joint 4} \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{Joint 5} \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \text{Joint 6} \end{array}$$

$NCJT(NJ) \times 1$

(b) Structure Coordinate Number Vector

Load magnitudes

Structure coordinate numbers

$$\mathbf{P} = \left[\begin{array}{r} 0 \\ -75 \\ 0 \\ 0 \\ 25 \\ 0 \\ 0 \\ -60 \end{array}\right] \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}$$

$NDOF \times 1$

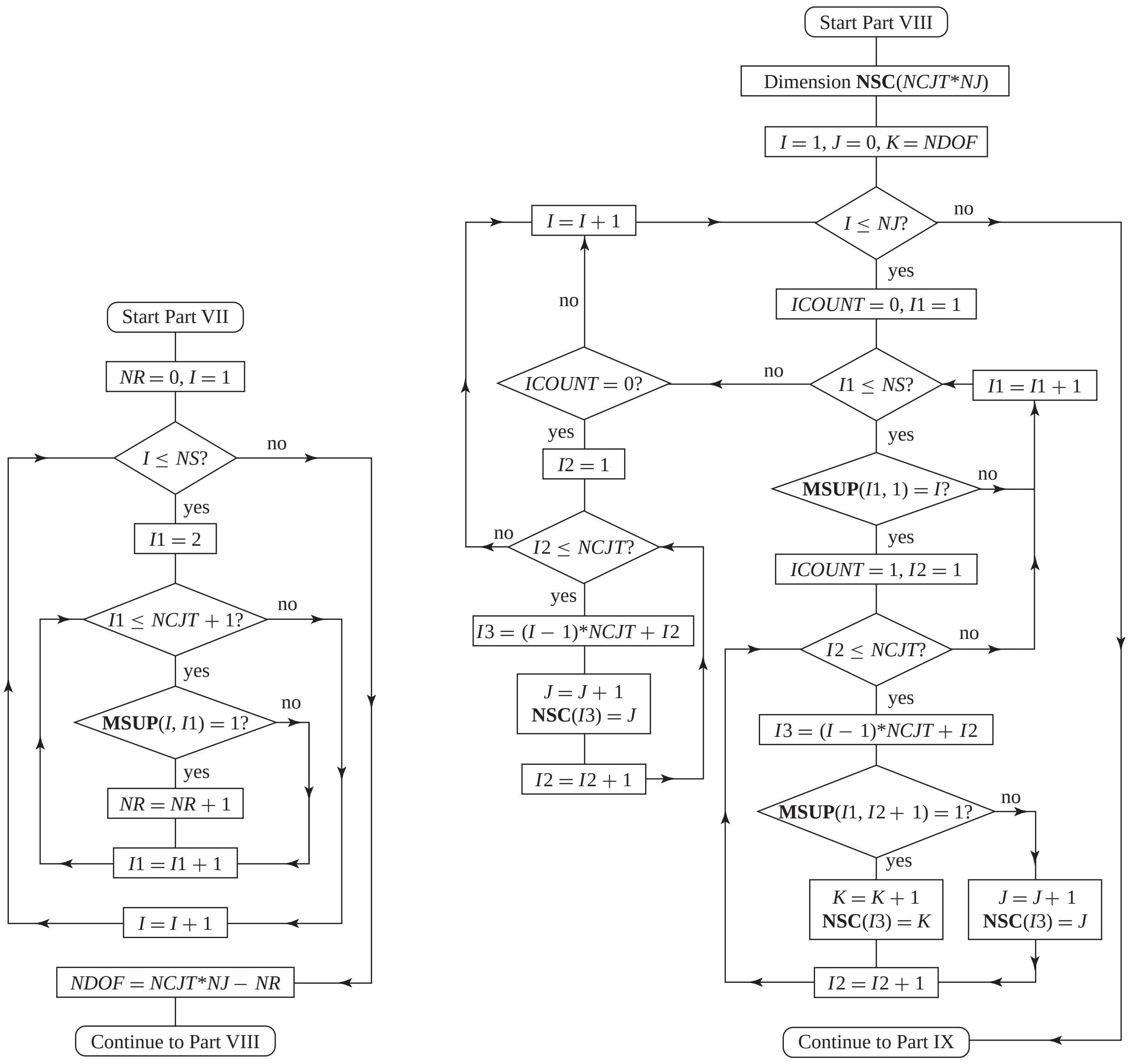
(c) Joint Load Vector

Fig. 4.7 (*continued*)

This procedure can be conveniently programmed using the flowcharts given in Fig. 4.8 on the next page. Figure 4.8(a) describes a program for determining the number of degrees of freedom and the number of restrained coordinates of the structure. (Note again that $NCJT = 2$ for plane trusses.) The program first determines the number of restrained coordinates (NR) by simply counting the number of 1s in the second and third columns of the support data matrix **MSUP**. Recall from our discussion of restraint codes in Section 4.1 that each 1 in the second or third column of **MSUP** represents a restraint (in either the X or Y direction) at a joint of the structure. With NR known, the number of degrees of freedom ($NDOF$) is evaluated from the following relationship (Eq. (3.3)).

$$NDOF = 2(NJ) - NR$$

For example, since the **MSUP** matrix for the example truss, given in Fig. 4.2(d), contains four 1s in its second and third columns, the number of



(a) Flowchart for Determining Number of Degrees of Freedom

(b) Flowchart for Generating Structure Coordinate Numbers

Fig. 4.8

restrained coordinates of the truss is four. Furthermore, since the truss has six joints (Fig. 4.2(c)), its number of degrees of freedom equals

$$NDOF = 2(6) - 4 = 8$$

Once the number of degrees of freedom ($NDOF$) has been determined, the program generates the structure coordinate number vector **NSC**, as shown by the flowchart in Fig. 4.8(b). Again, *NCJT* should be set equal to 2 for plane trusses. As this flowchart indicates, the program uses two integer counters, *J* and *K*, to keep track of the degrees-of-freedom and restrained coordinate numbers, respectively. The initial value of *J* is set equal to 0, whereas the initial value of *K* is set equal to *NDOF*.

The structure coordinates are numbered, one joint at a time, starting at joint 1 and proceeding sequentially to the joint number *NJ*. First, the number of the joint under consideration, *I*, is compared with the numbers in the first column of the support matrix **MSUP** to determine whether or not *I* is a support joint. If a match is found between *I* and one of the numbers in the first column of **MSUP**, then the counter *ICOUNT* is set equal to 1; otherwise, the value of *ICOUNT* remains 0 as initially assigned.

If joint *I* is not a support joint (i.e., $ICOUNT = 0$), then the number for its degree of freedom in the *X* direction is obtained by increasing the degrees-of-freedom counter *J* by 1 (i.e., $J = J + 1$), and this value of *J* is stored in row number $(I - 1)2 + 1$ of the **NSC** vector. Next, the value of *J* is again increased by 1 (i.e., $J = J + 1$) to obtain the number for the degree of freedom of joint *I* in the *Y* direction, and the new value of *J* is stored in row $(I - 1)2 + 2$ of the **NSC** vector.

If joint *I* is found to be a support joint (i.e., $ICOUNT = 1$), then the second column of the corresponding row of **MSUP** is checked to determine whether joint *I* is restrained in the *X* direction. If the joint is restrained in the *X* direction, then the number for its *X*-restrained coordinate is obtained by increasing the restrained coordinate counter *K* by 1 (i.e., $K = K + 1$), and this value of *K* is stored in row $(I - 1)2 + 1$ of the **NSC** vector. However, if the joint is not restrained in the *X* direction, then the degrees-of-freedom counter *J* is increased by 1, and its value (instead of that of *K*) is stored in row $(I - 1)2 + 1$ of the **NSC** vector. Next, the restraint condition in the *Y* direction at joint *I* is determined by checking the third column of the corresponding row of **MSUP**. If the joint is found to be restrained, then the counter *K* is increased by 1; otherwise, the counter *J* is increased by the same amount. The new value of either *K* or *J* is then stored in row $(I - 1)2 + 2$ of the **NSC** vector.

The computer program repeats the foregoing procedure for each joint of the structure to complete the structure coordinate number vector, **NSC**. As shown in Fig. 4.8(b), this part of the program (for generating structure coordinate numbers) can be conveniently coded using *Do Loop* or *For–Next* types of programming statements.

4.3 GENERATION OF THE STRUCTURE STIFFNESS MATRIX

The structure coordinate number vector **NSC**, defined in the preceding section, can be used to conveniently determine the member code numbers needed to establish the structure stiffness matrix **S**, without any reference to the visual image of the structure (e.g., the line diagram). The code numbers at the beginning of a member of a general framed structure are stored in the following rows of the **NSC** vector:

$$\left.\begin{array}{ll}\textbf{NSC}\text{ row for the first code number} & = (JB-1)NCJT+1\\ \textbf{NSC}\text{ row for the second code number} & = (JB-1)NCJT+2\\ \quad\vdots & \quad\vdots\\ \textbf{NSC}\text{ row for the }NCJT\text{th code number} & = (JB-1)NCJT+NCJT\end{array}\right\}\qquad\textbf{(4.1a)}$$

in which *JB* is the beginning joint of the member. Similarly, the code numbers at the end of the member, connected to joint *JE*, can be obtained from the following rows of the **NSC**:

$$\left.\begin{array}{ll}\textbf{NSC}\text{ row for the first code number} & = (JE-1)NCJT+1\\ \textbf{NSC}\text{ row for the second code number} & = (JE-1)NCJT+2\\ \quad\vdots & \quad\vdots\\ \textbf{NSC}\text{ row for the }NCJT\text{th code number} & = (JE-1)NCJT+NCJT\end{array}\right\}\qquad\textbf{(4.1b)}$$

Suppose, for example, that we wish to determine the code numbers for member 9 of the truss of Fig. 4.2(b). First, from the member data matrix **MPRP** of this truss (Fig. 4.2(g)), we obtain the beginning and end joints for this member as 3 and 5, respectively. (This information is obtained from row 9, columns 1 and 2, respectively, of **MPRP**.) Next, we determine the row numbers of the **NSC** vector in which the structure coordinate numbers for joints 3 and 5 are stored. Thus, at the beginning of the member (Eq. (4.1a) with $NCJT = 2$),

$$\begin{array}{ll}\textbf{NSC}\text{ row for the first code number} & = (3-1)2+1=5\\ \textbf{NSC}\text{ row for the second code number} & = (3-1)2+2=6\end{array}$$

Similarly, at the end of the member (Eq. (4.1b)),

$$\begin{array}{ll}\textbf{NSC}\text{ row for the first code number} & = (5-1)2+1=9\\ \textbf{NSC}\text{ row for the second code number} & = (5-1)2+2=10\end{array}$$

The foregoing calculations indicate that the code numbers for member 9 are stored in rows 5, 6, 9, and 10 of the **NSC** vector. Thus, from the appropriate rows of the **NSC** vector of the truss given in Fig. 4.7(b), we obtain the member's code numbers to be 3, 11, 5, 6. A visual check of the truss's line diagram in Fig. 4.7(a) indicates that these code numbers are indeed correct.

The procedure for forming the structure stiffness matrix **S** by assembling the elements of the member global stiffness matrices **K** was discussed in detail in Sections 3.7 and 3.8. A flowchart for programming this procedure is presented in Fig. 4.9, in which *NCJT* should be set equal to 2 for plane trusses. As indicated by the flowchart, this part of our computer program begins by initializing all the elements of the **S** matrix to 0. The assembly of the structure stiffness matrix is then carried out using a *Do Loop*, in which the following operations are performed for each member of the structure.

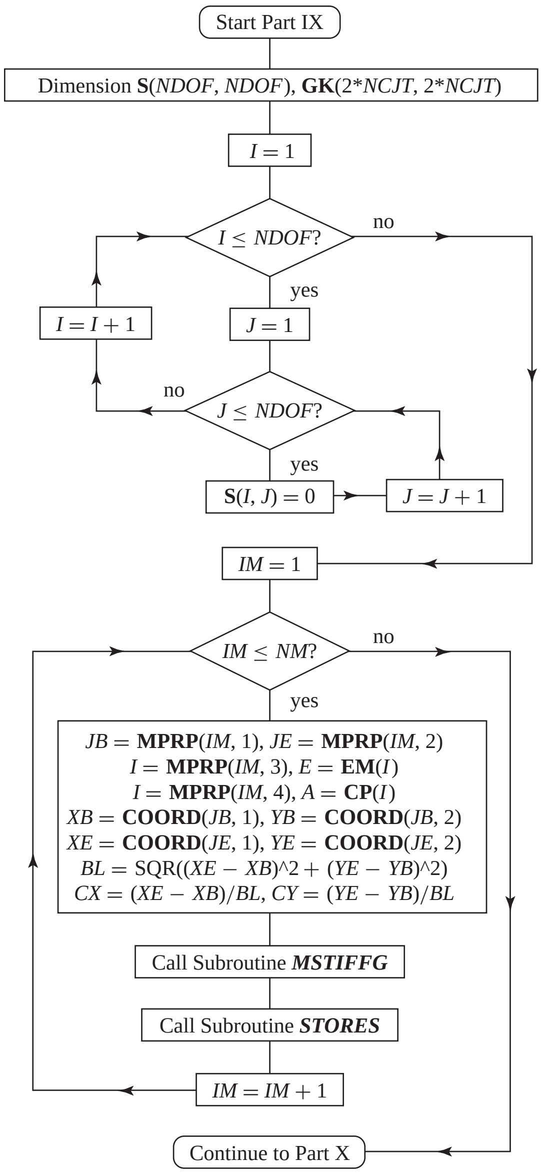
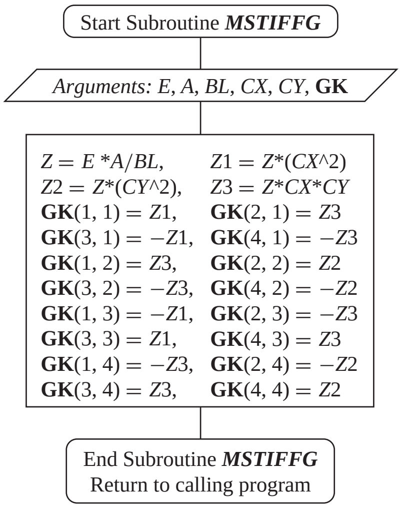

Start Part IX
Dimension **S**(*NDOF*, *NDOF*), **GK**(2**NCJT*, 2**NCJT*)
I = 1
I ≤ *NDOF*?
no
yes
I = *I* + 1
J = 1
no
J ≤ *NDOF*?
yes
S(*I*, *J*) = 0
J = *J* + 1
IM = 1
IM ≤ *NM*?
no
yes
JB = **MPRP**(*IM*, 1), *JE* = **MPRP**(*IM*, 2)
I = **MPRP**(*IM*, 3), *E* = **EM**(*I*)
I = **MPRP**(*IM*, 4), *A* = **CP**(*I*)
XB = **COORD**(*JB*, 1), *YB* = **COORD**(*JB*, 2)
XE = **COORD**(*JE*, 1), *YE* = **COORD**(*JE*, 2)
BL = SQR((*XE* − *XB*)^2 + (*YE* − *YB*)^2)
CX = (*XE* − *XB*)/*BL*, *CY* = (*YE* − *YB*)/*BL*
Call Subroutine ***MSTIFFG***
Call Subroutine ***STORES***
IM = *IM* + 1
Continue to Part X

Fig. 4.9 *Flowchart for Generating Structure Stiffness Matrix for Plane Trusses*

1. *Evaluation of member properties.* For the member under consideration, *IM*, the program reads the beginning joint number, *JB*, and the end joint number, *JE*, from the first and second columns, respectively, of the member data matrix **MPRP**. Next, the material property number is read from the third column of **MPRP**, and the corresponding value of the modulus of elasticity, *E*, is obtained from the elastic modulus vector **EM**. The program then reads the number of the member cross-section type from the fourth column of **MPRP**,


Start Subroutine *MSTIFFG*
Arguments: E, A, BL, CX, CY, **GK**
End Subroutine *MSTIFFG*
Return to calling program

```
Z = E *A/BL,        Z1 = Z*(CX^2)
Z2 = Z*(CY^2),      Z3 = Z*CX*CY
GK(1, 1) = Z1,      GK(2, 1) = Z3
GK(3, 1) = −Z1,     GK(4, 1) = −Z3
GK(1, 2) = Z3,      GK(2, 2) = Z2
GK(3, 2) = −Z3,     GK(4, 2) = −Z2
GK(1, 3) = −Z1,     GK(2, 3) = −Z3
GK(3, 3) = Z1,      GK(4, 3) = Z3
GK(1, 4) = −Z3,     GK(2, 4) = −Z2
GK(3, 4) = Z3,      GK(4, 4) = Z2
```

Fig. 4.10 *Flowchart of Subroutine **MSTIFFG** for Determining Member Global Stiffness Matrix for Plane Trusses*

and obtains the corresponding value of the cross-sectional area, A, from the cross-sectional property vector **CP**. Finally, the X and Y coordinates of the beginning joint JB and the end joint JE are obtained from the joint coordinate matrix **COORD**, and the member's length, BL, and its direction cosines, CX ($= \cos\theta$) and CY ($= \sin\theta$), are calculated using Eqs. (3.62).

2. *Determination of member global stiffness matrix* **GK** (= **K**) *by subroutine **MSTIFFG**.* After the necessary properties of the member under consideration, IM, have been evaluated, the program calls on the subroutine ***MSTIFFG*** to form the member stiffness matrix in the global coordinate system. (A flowchart of this subroutine is shown in Fig. 4.10.) Note that in the computer program, the member global stiffness matrix is named **GK** (instead of **K**) to indicate that it is a real (not an integer) matrix. As the flowchart in Fig. 4.10 indicates, the subroutine simply calculates the values of the various stiffness coefficients, and stores them into appropriate elements of the **GK** matrix, in accordance with Eq. (3.73).

3. *Storage of the elements of member global stiffness matrix* **GK** *into structure stiffness matrix* **S** *by subroutine **STORES**.* Once the matrix **GK** has been determined for the member under consideration, IM, the program (Fig. 4.9) calls the subroutine ***STORES*** to store the pertinent elements of **GK** in their proper positions in the structure stiffness matrix **S**. A flowchart of this subroutine, which essentially consists of two nested *Do Loops,* is given in Fig. 4.11. As this flowchart indicates, the outer *Do Loop* performs the following operations sequentially for each row of the **GK** matrix, starting with row 1 and ending with row 2($NCJT$): (a) the member code number $N1$ corresponding to the row under consideration, I, is obtained from the **NSC** vector using the procedure discussed previously in this

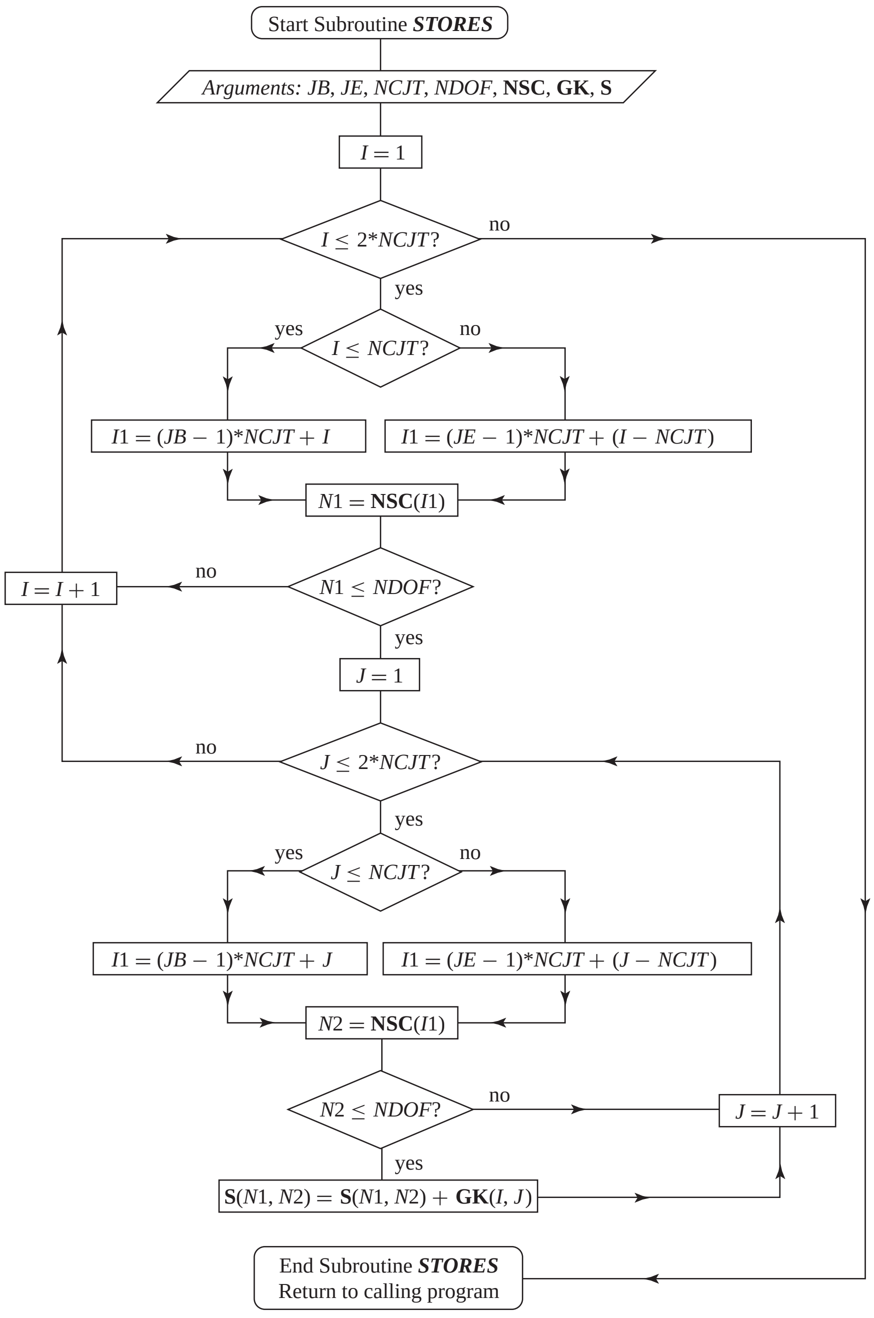

Start Subroutine ***STORES***
Arguments: JB, JE, NCJT, NDOF, **NSC**, **GK**, **S**
$I = 1$
$I \le 2*NCJT$?
no
yes
$I \le NCJT$?
yes
no
$I1 = (JB - 1)*NCJT + I$
$I1 = (JE - 1)*NCJT + (I - NCJT)$
$N1 = \mathbf{NSC}(I1)$
$N1 \le NDOF$?
no
yes
$I = I + 1$
$J = 1$
$J \le 2*NCJT$?
no
yes
$J \le NCJT$?
yes
no
$I1 = (JB - 1)*NCJT + J$
$I1 = (JE - 1)*NCJT + (J - NCJT)$
$N2 = \mathbf{NSC}(I1)$
$N2 \le NDOF$?
no
yes
$J = J + 1$
$\mathbf{S}(N1, N2) = \mathbf{S}(N1, N2) + \mathbf{GK}(I, J)$
End Subroutine ***STORES***
Return to calling program

Fig. 4.11 *Flowchart of Subroutine* ***STORES*** *for Storing Member Global Stiffness Matrix in Structure Stiffness Matrix*

section; (b) if $N1$ is less than or equal to *NDOF*, then the inner *Do Loop* is activated; otherwise, the inner loop is skipped; and (c) the row number I is increased by 1, and steps (a) through (c) are repeated. The inner loop, activated from the outer loop, performs the following operations sequentially for each column of the **GK** matrix, starting with column 1 and ending with column 2(*NCJT*): (a) the member code number $N2$ corresponding to the column under consideration, J, is obtained from the **NSC** vector; (b) if $N2$ is less than or equal to *NDOF*, then the value of the element in the Ith row and Jth column of **GK** is added to the value of the element in the $N1$th row and $N2$th column of **S**; otherwise, no action is taken; and (c) the column number J is increased by 1, and steps (a) through (c) are repeated. The inner loop ends when its steps (a) and (b) have been applied to all the columns of **GK**; the program control is then returned to step (c) of the outer loop. The subroutine ***STORES*** ends when steps (a) and (b) of the outer loop have been applied to all the rows of the **GK** matrix, thereby storing all the pertinent elements of the global stiffness matrix of the member under consideration, *IM*, in their proper positions in the structure stiffness matrix **S**.

Refocusing our attention on Fig. 4.9, we can see that formation of the structure stiffness matrix is complete when the three operations, described in the foregoing paragraphs, have been performed for each member of the structure.

4.4 FORMATION OF THE JOINT LOAD VECTOR

In this section, we consider the programming of the next analysis step, which involves formation of the joint load vector **P**. A flowchart for programming this process is shown in Fig. 4.12. Again, when analyzing plane trusses, the value of *NCJT* should be set equal to 2 in the program. It is seen from the figure that this part of our computer program begins by initializing each element of **P** to 0. The program then generates the load vector **P** by performing the following operations for each row of the load data vector **JP**, starting with row 1 and proceeding sequentially to row *NJL*:

1. For the row under consideration, I, the number of the loaded joint $I1$ is read from the **JP** vector.

2. The number of the X structure coordinate, N, at joint $I1$ is obtained from row $I2 = (I1 - 1)2 + 1$ of the **NSC** vector. If $N \leq NDOF$, then the value of the element in the Ith row and the first column of the load data matrix **PJ** (i.e., the X load component) is added to the Nth row of the load vector **P**; otherwise, no action is taken.

3. The **NSC** row number $I2$ is increased by 1 (i.e., $I2 = I2 + 1$), and the structure coordinate number, N, of the Y coordinate is read from the **NSC**. If $N \leq NDOF$, then the value of the element in the Ith row and the second column of **PJ** (i.e., the Y load component) is added to the Nth row of **P**; otherwise, no action is taken.

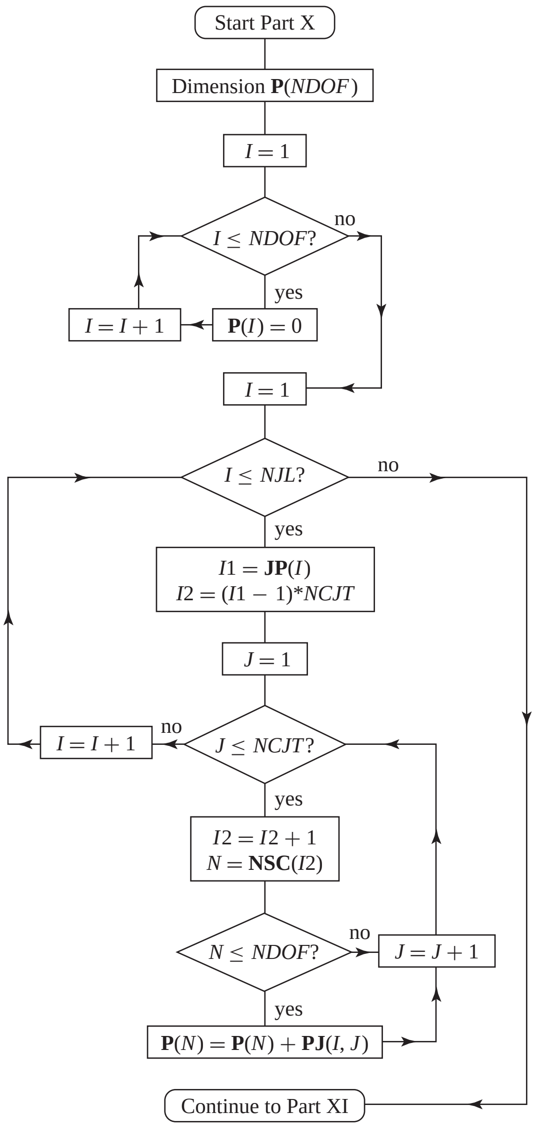


Fig. 4.12 *Flowchart for Forming Joint Load Vector*

The foregoing operations are repeated for each loaded joint of the structure to complete the joint load vector **P**.

To illustrate this procedure, let us form the joint load vector **P** for the example truss of Fig. 4.2(a) without referring to its visual image or line diagram (i.e., using only the input data matrices and the **NSC** vector). Recall that in Section 4.2, using the **MSUP** matrix, we determined that the number of degrees of freedom of this structure equals 8. Thus, the joint load vector **P** for the truss must be of order 8×1.

We begin generating **P** by focusing our attention on row 1 (i.e., $I = 1$) of the load data vector **JP** (Fig. 4.2(h)), from which we determine the number of the first loaded joint, $I1$, to be 2. We then determine the row of the **NSC** in which the number of the X structure coordinate at joint 2 is stored, using the following relationship:

$$I2 = (2 - 1)2 + 1 = 3$$

From row 3 of the **NSC** vector given in Fig. 4.7(b), we read the number of the structure coordinate under consideration as 1 (i.e., $N = 1$). This indicates that the force component in the first row and first column of the load data matrix **PJ** (i.e., the X component of the load acting at joint 2) must be stored in the first row of **P**; that is, **P**(1) = 0. Next, we increase $I2$ by 1 (i.e., $I2 = 4$) and, from row 4 of the **NSC**, we find the number of the Y structure coordinate at the joint to be 2 (i.e., $N = 2$). This indicates that the load component in the first row and second column of **PJ** is to be stored in the second row of **P**; that is, **P**(2) = −75.

Having stored the loads acting at joint 2 in the load vector **P**, we now focus our attention on the second row of **JP** (i.e., $I = 2$), and read the number of the next loaded joint, $I1$, as 5. We then determine the **NSC** row where the number of the X structure coordinate at joint 5 is stored as

$$I2 = (5 - 1)2 + 1 = 9$$

From row 9 of the **NSC** (Fig. 4.7(b)), we find the number of the structure coordinate under consideration to be 5 (i.e., $N = 5$). Thus, the force component in row 2 and column 1 of **PJ** must be stored in row 5 of **P**; or **P**(5) = 25. Next, we increase $I2$ by 1 to 10, and from row 10 of the **NSC** read the structure coordinate number, N, as 6. Thus, the load component in the second row and second column of **PJ** is stored in the sixth row of **P**; or **P**(6) = 0.

Finally, by repeating the foregoing procedure for row 3 of **JP**, we store the X and Y force components at joint 6 in rows 7 and 8, respectively, of **P**. The completed joint load vector **P** thus obtained is shown in Fig. 4.7(c).

4.5 SOLUTION FOR JOINT DISPLACEMENTS

Having programmed the generation of the structure stiffness matrix **S** and the joint load vector **P**, we now proceed to the next part of our computer program, which calculates the joint displacements, **d**, by solving the structure stiffness relationship, **Sd** = **P** (Eq. (3.89)). A flowchart for programming this analysis step is depicted in Fig. 4.13. The program solves the system of simultaneous equations, representing the stiffness relationship, **Sd** = **P**, using the Gauss–Jordan elimination method discussed in Section 2.4.

It should be recognized that the program for the calculation of joint displacements, as presented in Fig. 4.13, involves essentially the same operations as the program for the solution of simultaneous equations given in Fig. 2.2. However, in the previous program (Fig. 2.2), the elementary operations were

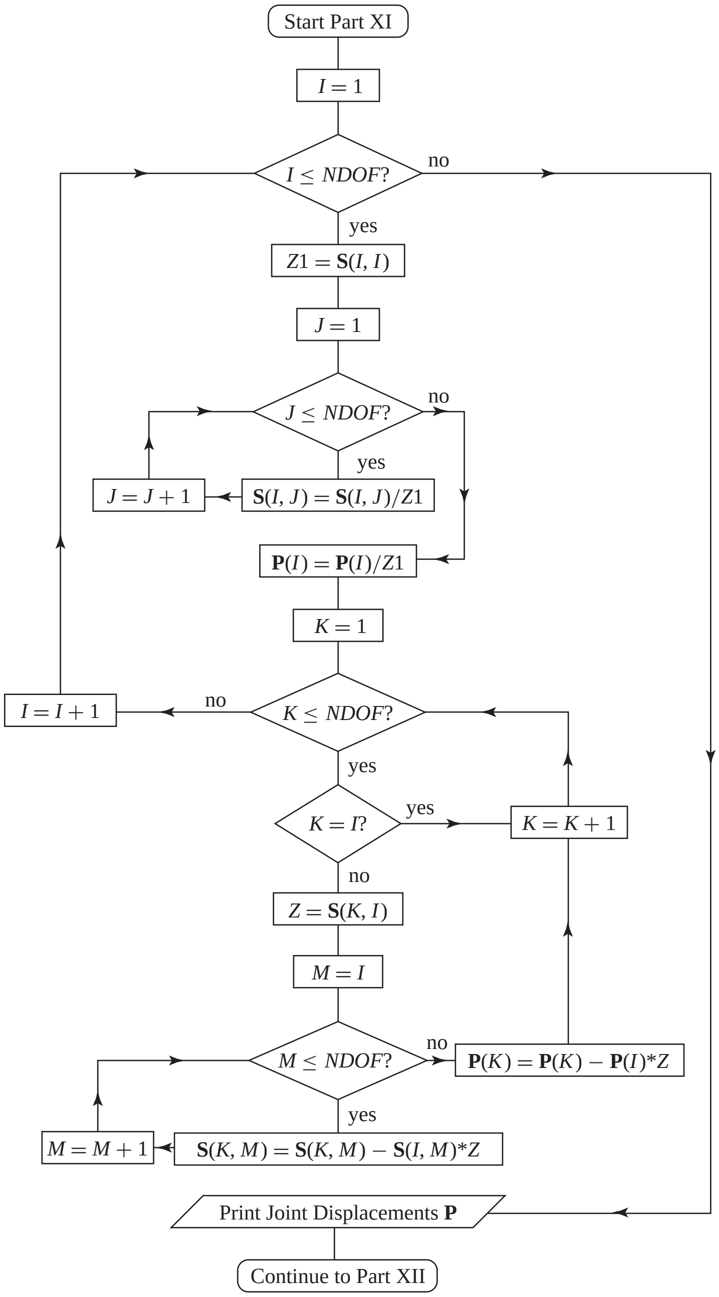


Fig. 4.13 *Flowchart for Calculation of Joint Displacements by Gauss–Jordan Method*

applied to an augmented matrix; in the present program (Fig. 4.13), to save space in computer memory, no augmented matrix is formed, and the elementary operations are applied directly to the structure stiffness matrix **S** and the joint load vector **P**. Thus, at the end of the Gauss–Jordan elimination process, the **S** matrix is reduced to a unit matrix, and the **P** vector contains values of the joint displacements. In the rest of our computer program, therefore, **P** (instead of **d**) is considered to be the joint displacement vector. The joint displacements thus obtained can be communicated to the user through a printout or on the screen.

4.6 CALCULATION OF MEMBER FORCES AND SUPPORT REACTIONS

In this section, we consider programming of the final analysis step, which involves calculation of the member forces and support reactions. A flowchart for programming this analysis step is presented in Fig. 4.14, with $NCJT = 2$ for plane trusses. As shown there, this part of our computer program begins by initializing each element of the reaction vector, **R**, to 0. The member forces and support reactions are then determined by performing the following operations for each member of the structure, via a *Do Loop*.

1. *Evaluation of member properties.* For the member under consideration, *IM*, the program reads the beginning joint number *JB*, the end joint number *JE*, the modulus of elasticity *E*, the cross-sectional area *A*, and the *X* and *Y* coordinates of the beginning and end joints. It then calculates the member length, *BL*, and direction cosines, *CX* ($= \cos\theta$) and *CY* ($= \sin\theta$), using Eqs. (3.62).

2. *Evaluation of member global end displacements* **V** **(= v)** *by subroutine* ***MDISPG*.** After the properties of the member under consideration, *IM*, have been calculated, the computer program calls subroutine ***MDISPG***, to obtain the member end displacements in the global coordinate system. A flowchart of this subroutine is given in Fig. 4.15. As this flowchart indicates, after initializing **V** to 0, the subroutine reads, in order, for each of the member end displacements, V_I, the number of the corresponding structure coordinate, *N*, at joint *JB* or *JE*, from the **NSC** vector. If the structure coordinate number *N*, corresponding to an end displacement V_I, is found to be less than or equal to *NDOF*, then the value of the element in the *N*th row of the joint-displacement vector **P** (= **d**) is stored in the *I*th row of the member displacement vector **V**.

3. *Determination of member transformation matrix* **T** *by subroutine* ***MTRANS***. After the global end-displacement vector **V** for the member under consideration, *IM*, has been evaluated, the main program (Fig. 4.14) calls on the subroutine ***MTRANS*** to form the member transformation matrix **T**. A flowchart of this subroutine is shown in Fig. 4.16. As this figure indicates, the subroutine first initializes **T** to 0, and then simply stores the values of the direction

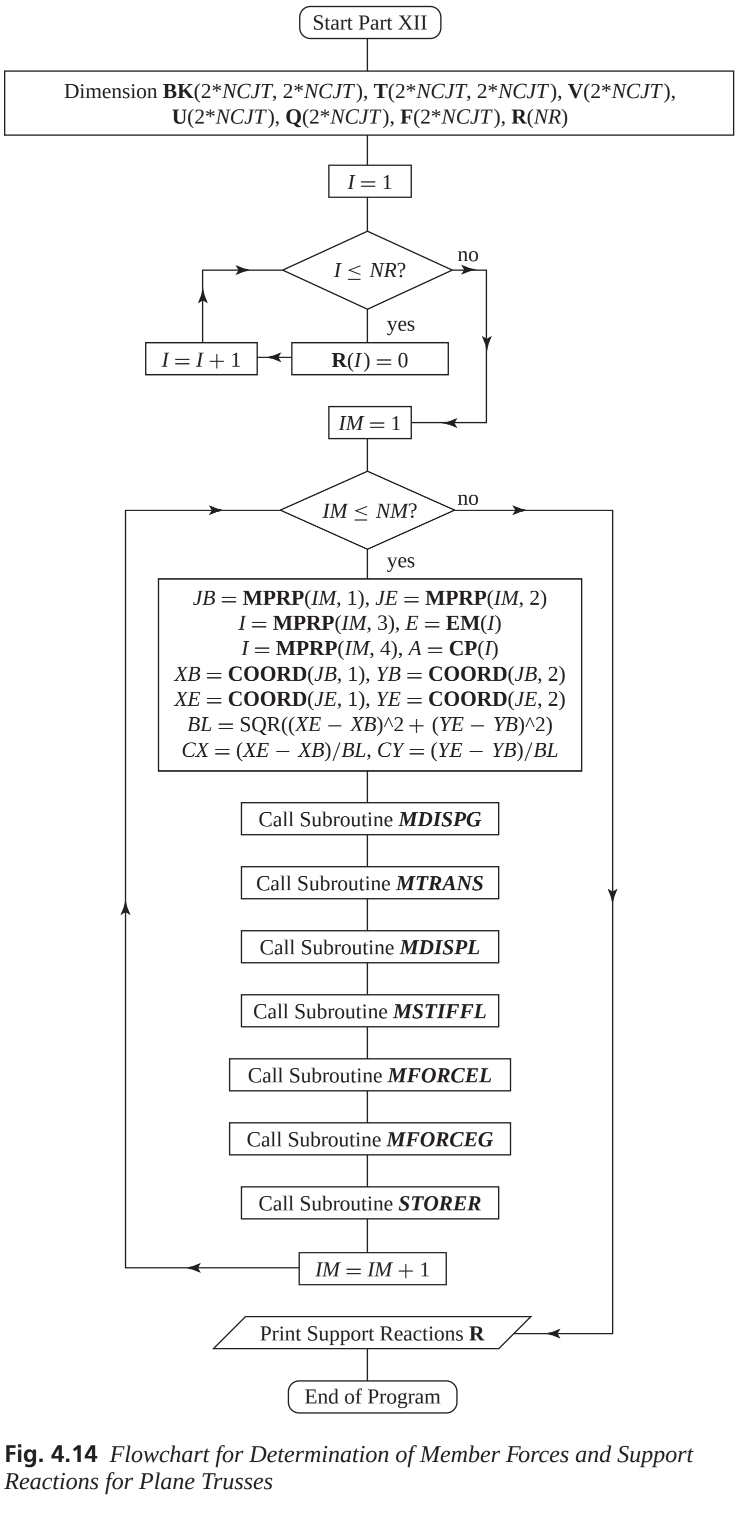

Start Part XII
Dimension **BK**(2*$NCJT$, 2*$NCJT$), **T**(2*$NCJT$, 2*$NCJT$), **V**(2*$NCJT$), **U**(2*$NCJT$), **Q**(2*$NCJT$), **F**(2*$NCJT$), **R**(NR)
$I = 1$
$I \leq NR$?
no
yes
R(I) = 0
$I = I + 1$
$IM = 1$
$IM \leq NM$?
no
yes
JB = **MPRP**(IM, 1), JE = **MPRP**(IM, 2)
I = **MPRP**(IM, 3), E = **EM**(I)
I = **MPRP**(IM, 4), A = **CP**(I)
XB = **COORD**(JB, 1), YB = **COORD**(JB, 2)
XE = **COORD**(JE, 1), YE = **COORD**(JE, 2)
BL = SQR(($XE - XB$)^2 + ($YE - YB$)^2)
$CX = (XE - XB)/BL$, $CY = (YE - YB)/BL$
Call Subroutine ***MDISPG***
Call Subroutine ***MTRANS***
Call Subroutine ***MDISPL***
Call Subroutine ***MSTIFFL***
Call Subroutine ***MFORCEL***
Call Subroutine ***MFORCEG***
Call Subroutine ***STORER***
$IM = IM + 1$
Print Support Reactions **R**
End of Program

Fig. 4.14 *Flowchart for Determination of Member Forces and Support Reactions for Plane Trusses*

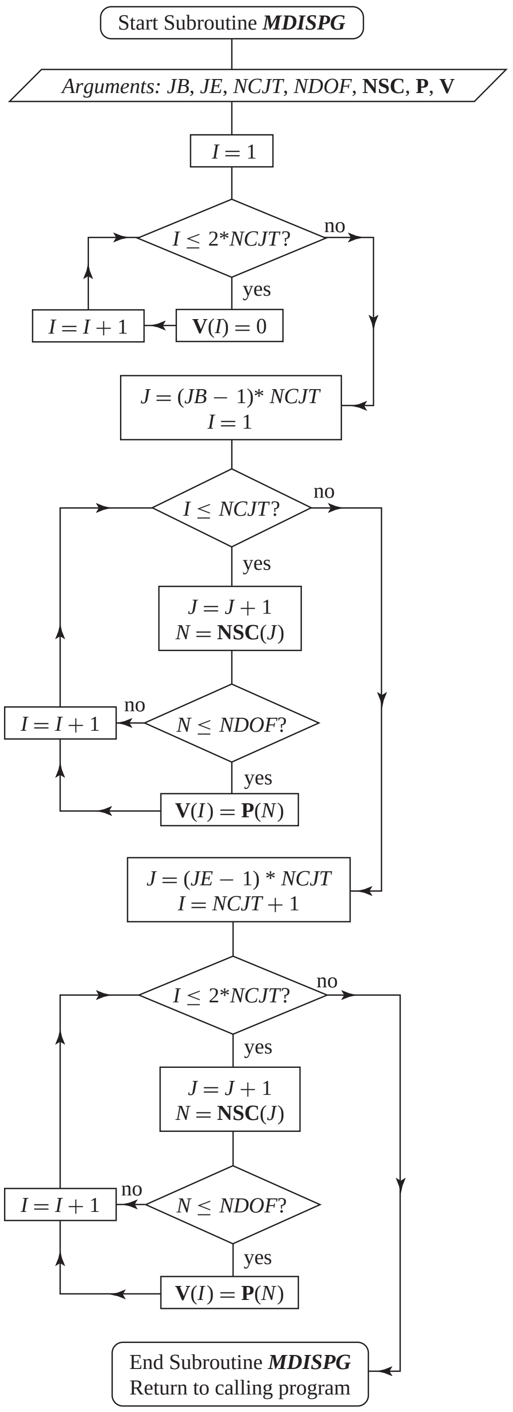


Fig. 4.15 *Flowchart of Subroutine* ***MDISPG*** *for Determining Member Global Displacement Vector*

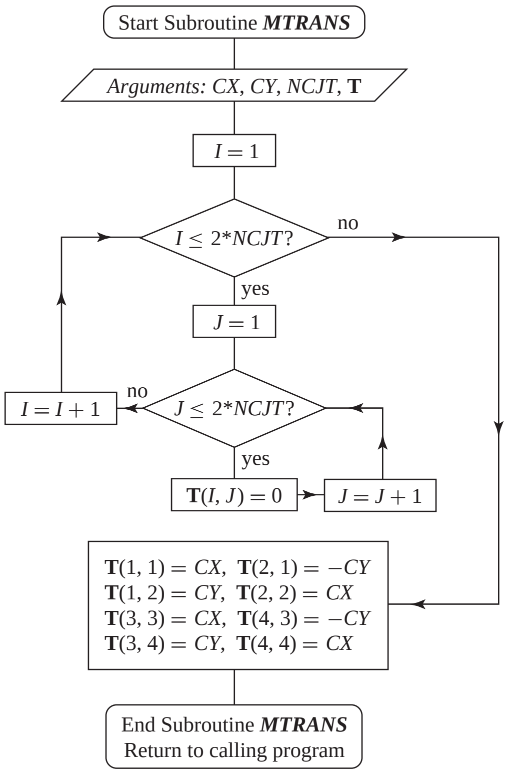

Start Subroutine MTRANS
Arguments: CX, CY, NCJT, T
I = 1
I ≤ 2*NCJT?
no
yes
J = 1
J ≤ 2*NCJT?
no
I = I + 1
yes
T(I, J) = 0
J = J + 1
T(1, 1) = CX, T(2, 1) = −CY
T(1, 2) = CY, T(2, 2) = CX
T(3, 3) = CX, T(4, 3) = −CY
T(3, 4) = CY, T(4, 4) = CX
End Subroutine MTRANS
Return to calling program

Fig. 4.16 *Flowchart of Subroutine* ***MTRANS*** *for Determining Member Transformation Matrix for Plane Trusses*

cosines *CX* and *CY*, with appropriate plus or minus signs, into various elements of **T** in accordance with Eq. (3.61).

4. *Calculation of member local end displacements* **U (= u)** *by subroutine* ***MDISPL***. Next, as shown in Fig. 4.14, the program calls subroutine ***MDISPL*** to obtain the local end displacements of the member under consideration, *IM*. From the flowchart given in Fig. 4.17, we can see that after initializing **U** to 0, this subroutine calculates the member local end-displacement vector by applying the relationship **U = TV** (Eq. (3.63)). The procedure for multiplying matrices was discussed in Section 2.3, and subroutine ***MDISPL*** (Fig. 4.17) uses essentially the same operations as the program for matrix multiplication given in Fig. 2.1.

5. *Determination of member local stiffness matrix* **BK (= k)** *by subroutine* ***MSTIFFL***. After the local end displacements of the member under

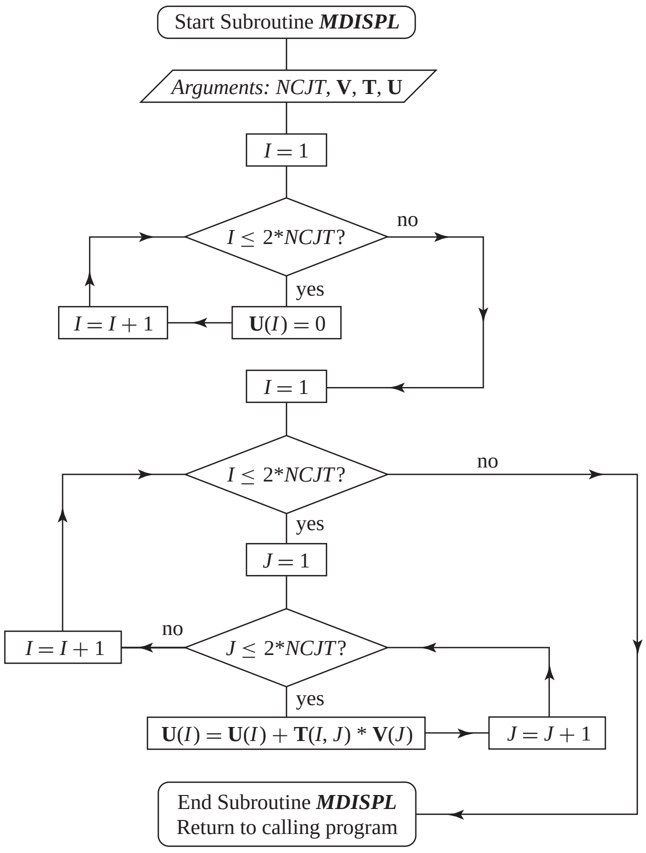


Fig. 4.17 *Flowchart of Subroutine* ***MDISPL*** *for Determining Member Local Displacement Vector*

consideration, *IM*, have been evaluated, the program calls subroutine ***MSTIFFL*** to form the member stiffness matrix in the local coordinate system. A flowchart of this subroutine is shown in Fig. 4.18, in which the member local stiffness matrix is identified by the name **BK** (instead of **k**) to indicate that it is a real matrix. As this figure indicates, the subroutine, after initializing **BK** to 0, simply calculates the values of the various stiffness coefficients and stores them in appropriate elements of **BK**, in accordance with Eq. (3.27).

6. *Evaluation of member local end forces* **Q** *by subroutine* ***MFORCEL***. As shown in Fig. 4.14, the program then calls subroutine ***MFORCEL*** to obtain the local end forces of the member under consideration, *IM*. From the flowchart depicted in Fig. 4.19, we can see that, after initializing **Q** to 0, this

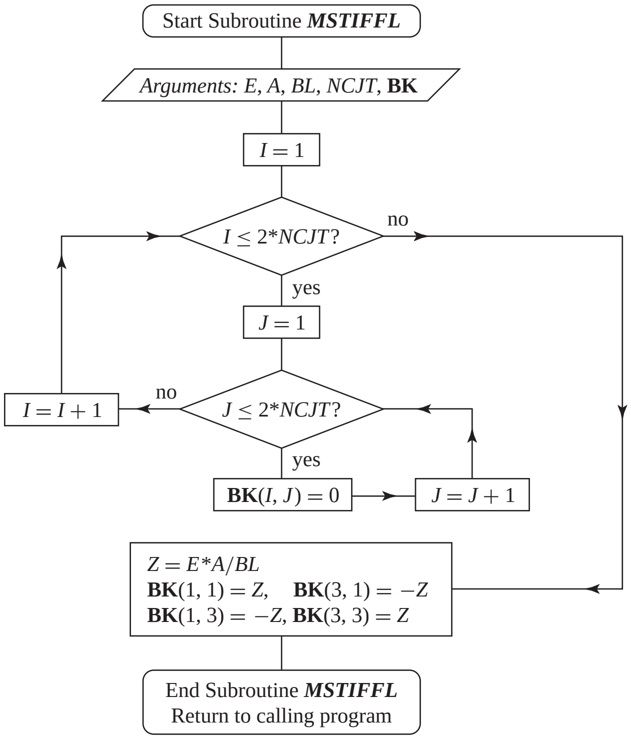


Fig. 4.18 *Flowchart of Subroutine* ***MSTIFFL*** *for Determining Member Local Stiffness Matrix for Plane Trusses*

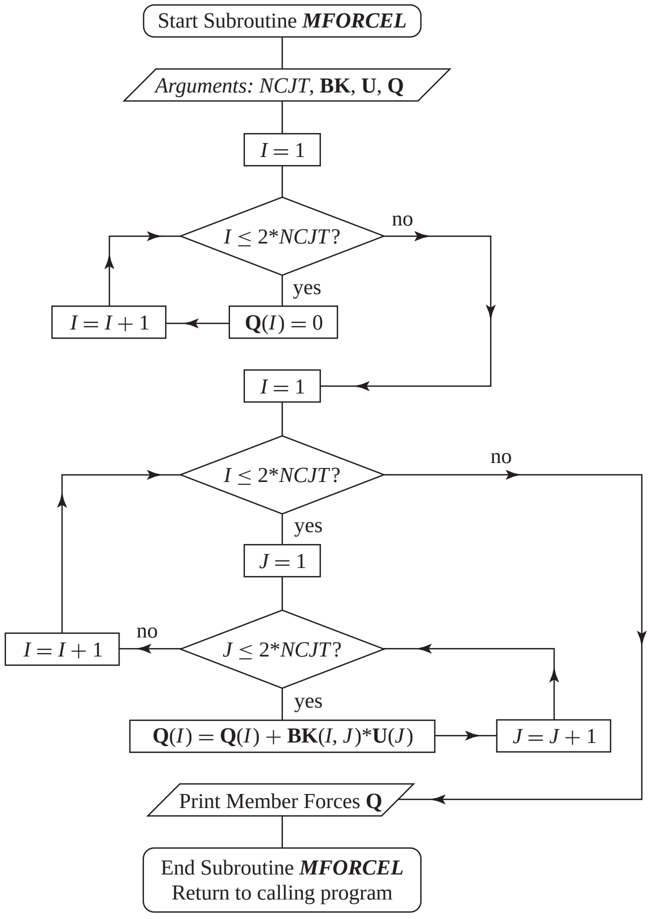


Fig. 4.19 *Flowchart of Subroutine* ***MFORCEL*** *for Determining Member Local Force Vector*

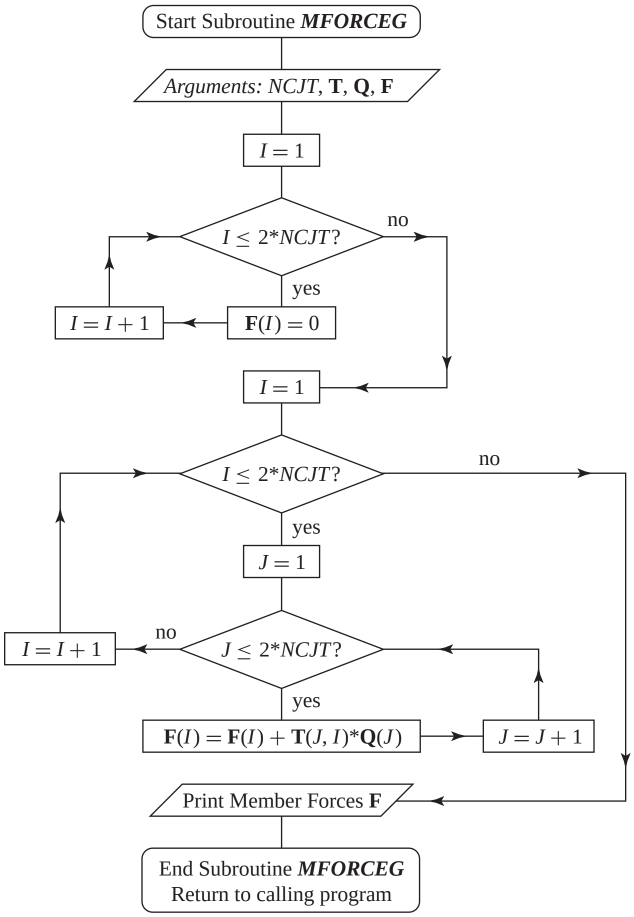

Start Subroutine *MFORCEG*
Arguments: NCJT, **T**, **Q**, **F**
$I = 1$
$I \le 2*NCJT?$
no
yes
$I = I + 1$
$\mathbf{F}(I) = 0$
$I = 1$
$I \le 2*NCJT?$
no
yes
$J = 1$
$I = I + 1$
no
$J \le 2*NCJT?$
yes
$\mathbf{F}(I) = \mathbf{F}(I) + \mathbf{T}(J, I)*\mathbf{Q}(J)$
$J = J + 1$
Print Member Forces **F**
End Subroutine *MFORCEG*
Return to calling program

Fig. 4.20 *Flowchart of Subroutine* ***MFORCEG*** *for Determining Member Global Force Vector*

subroutine calculates the member local end forces using the relationship $\mathbf{Q} = \mathbf{BK\,U}$ (Eq. (3.7)). The **Q** vector thus obtained is then printed or displayed on the screen.

7. *Calculation of member global end forces* **F** *by subroutine* ***MFORCEG***. After the local end forces of the member under consideration, *IM*, have been evaluated, the computer program calls subroutine ***MFORCEG*** to calculate the member end forces in the global coordinate system. A flowchart of this subroutine is given in Fig. 4.20. From the figure, we can see that after initializing **F** to 0, the subroutine calculates the global end forces by applying the relationship $\mathbf{F} = \mathbf{T}^T\mathbf{Q}$ (Eq. (3.66)); these forces are then communicated to the user through a printer or on the screen.

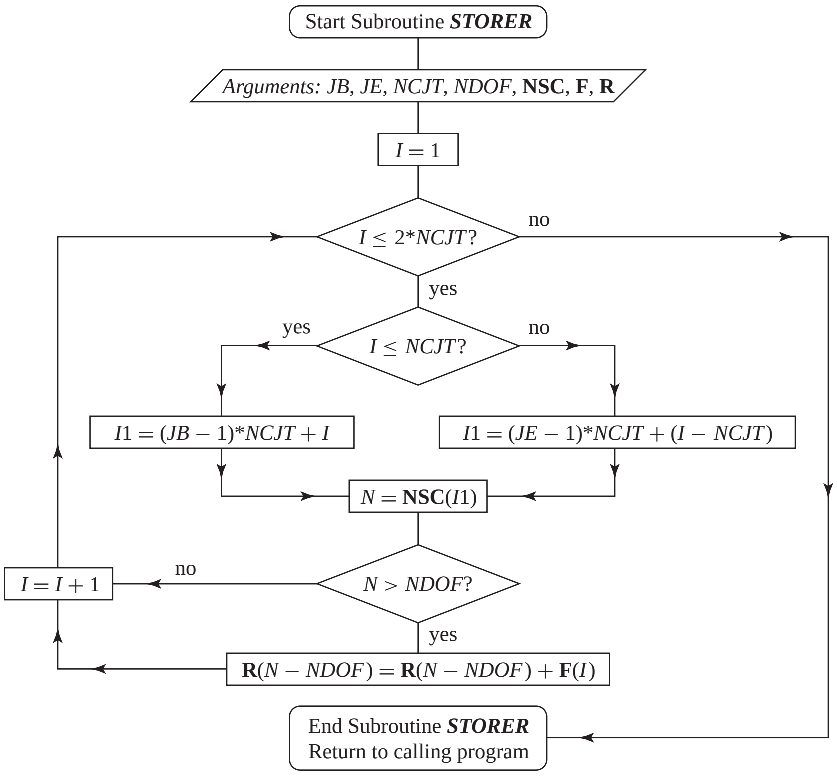

Start Subroutine ***STORER***
Arguments: JB, JE, NCJT, NDOF, **NSC**, **F**, **R**
$I = 1$
$I \leq 2*NCJT$?
no
yes
$I \leq NCJT$?
yes
no
$I1 = (JB - 1)*NCJT + I$
$I1 = (JE - 1)*NCJT + (I - NCJT)$
$N = \mathbf{NSC}(I1)$
$I = I + 1$
no
$N > NDOF$?
yes
$\mathbf{R}(N - NDOF) = \mathbf{R}(N - NDOF) + \mathbf{F}(I)$
End Subroutine ***STORER***
Return to calling program

Fig. 4.21 *Flowchart of Subroutine **STORER** for Storing Member Global Forces in Support Reaction Vector*

8. *Storage of the elements of member global force vector* **F** *in reaction vector* **R** *by subroutine **STORER***. Once the global force vector **F** has been determined for the member under consideration, *IM*, the program (Fig. 4.14) calls subroutine ***STORER*** to store the pertinent elements of **F** in their proper positions in the support reaction vector **R**. A flowchart of this subroutine, which essentially consists of a *Do Loop*, is given in Fig. 4.21. As shown in this flowchart, the subroutine reads, in order, for each of the member's forces, F_I, the number of the corresponding structure coordinate, N, from the **NSC** vector. If $N > NDOF$, then the value of F_I is added to the $(N - NDOF)$th row of the reaction vector **R**.

Returning our attention to Fig. 4.14, we can see that the formation of the reaction vector **R** is completed when the foregoing eight operations have been performed for each member of the structure. The support reactions thus obtained can then be communicated to the user via a printer or on the screen. A sample printout is given in Fig. 4.22, showing the results of the analysis for the example truss of Fig. 4.2.

```
***************************************************
*               Results of Analysis               *
***************************************************

                  ===================
                  Joint Displacements
                  ===================

 Joint No.       X Translation        Y Translation
 ---------       -------------        -------------
     1             0.0000E+00           0.0000E+00
     2             7.4568E-02          -2.0253E-01
     3             1.1362E-01           0.0000E+00
     4             1.0487E-01           0.0000E+00
     5             5.7823E-02          -1.5268E-01
     6             2.8344E-02          -7.9235E-02

                  ====================
                  Member Axial Forces
                  ====================

           Member        Axial Force (Qa)
           ------        ----------------
              1            6.0069E+01 (T)
              2            3.1459E+01 (T)
              3            4.8629E+00 (C)
              4            2.3747E+01 (C)
              5            5.3543E+01 (T)
              6            8.5105E+01 (C)
              7            4.3836E+01 (C)
              8            3.5762E+01 (T)
              9            4.5402E+01 (C)
             10            6.0787E+00 (T)

                  =================
                  Support Reactions
                  =================

  Joint No.            X Force             Y Force
  ---------            -------             -------
      1             -2.5000E+01          2.6301E+01
      3              0.0000E+00          1.1235E+02
      4              0.0000E+00         -3.6472E+00

**************** End of Analysis ****************
```

Fig. 4.22 *A Sample Printout of Analysis Results*

SUMMARY

In this chapter, we have developed a general computer program for the analysis of plane trusses subjected to joint loads. The general program consists of a main program, which is subdivided into twelve parts, and nine subroutines. Brief descriptions of the various parts of the main program, and the subroutines, are provided in Table 4.1 for quick reference.

Table 4.1 *Computer Program for Analysis of Plane Trusses*

Main program part	Description
I	Reads and stores joint data (Fig. 4.3(a))
II	Reads and stores support data (Fig. 4.3(b))
III	Reads and stores material properties (Fig. 4.3(c))
IV	Reads and stores cross-sectional properties (Fig. 4.3(d))
V	Reads and stores member data (Fig. 4.3(e))
VI	Reads and stores joint loads (Fig. 4.3(f))
VII	Determines the number of degrees of freedom *NDOF* of the structure (Fig. 4.8(a))
VIII	Forms the structure coordinate number vector **NSC** (Fig. 4.8(b))
IX	Generates the structure stiffness matrix **S** (Fig. 4.9); subroutines called: ***MSTIFFG*** and ***STORES***
X	Forms the joint load vector **P** (Fig. 4.12)
XI	Calculates the structure's joint displacements by solving the stiffness relationship, $\mathbf{Sd} = \mathbf{P}$, using the Gauss–Jordan elimination method. The vector **P** now contains joint displacements (Fig. 4.13).
XII	Determines the member end force vectors **Q** and **F**, and the support reaction vector **R** (Fig. 4.14); subroutines called: ***MDISPG, MTRANS, MDISPL, MSTIFFL, MFORCEL, MFORCEG,*** and ***STORER***

Subroutine	Description
MDISPG	Determines the member global displacement vector **V** from the joint displacement vector **P** (Fig. 4.15)
MDISPL	Calculates the member local displacement vector $\mathbf{U} = \mathbf{TV}$ (Fig. 4.17)
MFORCEG	Determines the member global force vector $\mathbf{F} = \mathbf{T}^T\mathbf{Q}$ (Fig. 4.20)
MFORCEL	Evaluates the member local force vector $\mathbf{Q} = \mathbf{BK}\,\mathbf{U}$ (Fig. 4.19)
MSTIFFG	Forms the member global stiffness matrix **GK** (Fig. 4.10)
MSTIFFL	Forms the member local stiffness matrix **BK** (Fig. 4.18)
MTRANS	Forms the member transformation matrix **T** (Fig. 4.16)
STORER	Stores the pertinent elements of the member global force vector **F** in the reaction vector **R** (Fig. 4.21)
STORES	Stores the pertinent elements of the member global stiffness matrix **GK** in the structure stiffness matrix **S** (Fig. 4.11)

PROBLEMS

The objective of the following problems is to develop, incrementally, a computer program for the analysis of plane trusses; while testing each program increment for correctness, as it is being developed. The reader is strongly encouraged to manually solve as many of the problems (3.16 through 3.25) as possible, so that these hand-calculation results can be used to check the correctness of the various parts of the computer program.

Section 4.1

4.1 Develop an input module of a computer program for the analysis of plane trusses, which can perform the following operations:

a. read from a data file, or computer screen, all the necessary input data;
b. store the input data in computer memory in the form of scalars, vectors, and/or matrices, as appropriate; and
c. print the input data from computer memory.

Check the program for correctness by inputting data for the trusses of Problems 3.16 through 3.25, and by carefully examining the printouts of the input data to ensure that all data have been correctly read and stored.

Section 4.2

4.2 Extend the program developed in Problem 4.1, so that it can perform the following additional operations:

a. determining the number of degrees of freedom (*NDOF*) of the structure;
b. forming the structure coordinate number vector **NSC**; and
c. printing out the *NDOF* and **NSC**.

To check the program for correctness, use it to determine the *NDOF* and **NSC** for the trusses of Problems 3.16 through 3.25, and compare the computer-generated results to those obtained by hand calculations.

Section 4.3

4.3 Extend the program of Problem 4.2 to generate, and print, the structure stiffness matrix **S**. Use the program to generate the structure stiffness matrices for the trusses of Problems 3.16 through 3.25, and compare the computer-generated **S** matrices to those obtained by hand calculations.

Section 4.4

4.4 Extend the program developed in Problem 4.3 to form, and print, the joint load vector **P**. Apply the program to the trusses of Problems 3.16 through 3.25, and compare the computer-generated **P** vectors to those obtained by hand calculations.

Section 4.5

4.5 Extend the program of Problem 4.4 so that it can: (a) calculate the structure's joint displacements by solving the stiffness relationship, **Sd** = **P**, using the Gauss–Jordan elimination method; and (b) print the joint displacements. Using the program, determine the joint displacements for the trusses of Problems 3.16 through 3.25, and compare the computer-generated results to those obtained by hand calculations.

Section 4.6

4.6 Extend the program developed in Problem 4.5 so that it can determine and print: (a) the local end forces, **Q**, for each member of the truss; and (b) the support reaction vector **R**. Use the program to analyze the trusses of Problems 3.16 through 3.25, and compare the computer-generated results to those obtained by hand calculations.