A STUDY OF FULL DISPLACEMENT DESIGN OF FRAME STRUCTURES USING DISPLACEMENT SENSITIVITY ANALYSIS

by

Ashraf M. Abou-Rayan

thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Master of Science
in
Civil Engineering

APPROVED:

A. E. Somers, Jr., Chairman

S. M. Holzer

R. H. Plaut

December 1985

Blacksburg, Virginia

A STUDY OF FULL DISPLACEMENT DESIGN OF FRAME STRUCTURES USING DISPLACEMENT SENSITIVITY ANALYSIS

by

Ashraf M. Abou-Rayan

A. E. Somers, Jr., Chairman

Civil Engineering

(ABSTRACT)

The intent of this study is to develop an algorithm for structural design based on allowable displacements for structural members, independent of stresses caused by the configurations imposed. Structural design can be based on displacement constraints applied in the same basic format as stress constraints so that convergence is based on allowable displacements rather than on stresses. The objective includes the following:

1. To develop an allowable displacement criterion based on structural design considerations for steel and reinforced concrete structures.
2. To implement the design variable linking algorithm to allow each member to be designed for one displacement constraint and one design variable, even though it is composed of separate elements.
3. To develop the displacement design algorithm (independent of stresses) using Newton's method for obtaining the

displacement solution within an iterative process. (This involves the development of displacement sensitivity derivatives and a sensitivity matrix.)

4. To study the convergence characteristics of the displacement controlled design on several test structures; four structural design cases were used :
   A. single story portal frame
   B. two story portal frame
   C. three story portal frame
   D. two bay portal frame

The conclusions drawn from these examples are : (1) the convergence rate depends on the starting initial values for section properties more than the size of the structure;(2) Traynor's equations are apparently no longer valid for larger structures; (3) for small structures, the convergence rate is faster under Traynor's equations than under Ang's equations.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Dr. A. E. Somers, Jr. , for his guidance and suggestions throughout this study . Thanks are also due to Dr. S. M. Holzer and Dr. R. H. Plaut for reviewing this thesis and serving on the committee . Finally, the author would like to recognize his parents, to whom this thesis is dedicated, for their continued support and encouragement .

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1.0 INTRODUCTION

This thesis presents the development and results of an iterative method for obtaining a structural design based on allowable displacements independent of stresses that may occur. The purpose is to understand the characteristics of a displacement controlled iteration procedure. The objective includes the following :

1. Develop an allowable displacement criterion based on structural design considerations for steel and reinforced concrete structures.
2. Implement the design variable linking algorithm to allow each member to be designed for one displacement constraint and one design variable, even though it is composed of separate elements.
3. Develop the displacement design algorithm (independent of stresses) using Newton's method for obtaining the displacement solution within an iterative process (this involves the development of displacement sensitivity derivatives and a sensitivity matrix).
4. Study the convergence characteristics of the displacement controlled design on several test structures; four structural design cases were used.

In recent years there has been considerable literature devoted to various aspects of optimum structural design. Haug [6] states that the literature appears to be divided into classes depending on the design constraints imposed. Many constraints have been treated, but the essential ones are bounds on stress, displacement, buckling, natural frequency, and multiple loading conditions. Papers involving deflection constraints are less numerous .

Kirsch [8] divides the optimization methods into two different categories:

1. Analytical methods, which implement the mathematical theory of calculus (variational methods, etc.).
2. Numerical methods, which use the mathematical programming methods. These numerical methods have become widely used due to the growing use of digital computers and mainly to the rapid growth in the computer's capacity.

Sensitivity analysis plays an important role in structural optimization. Many of the current papers are devoted to sensitivity analysis. Arora [2] uses the sensitivity analysis in solving a structural optimization problem, which includes two types of constraint conditions, stress and displacement. He suggests imposing the constraints for displacements at nodal points of the structure.

Haug [6] uses the steepest descent method to solve a two-constraint structural problem (displacement and stress). The method first estimates the optimum design variables and then makes small changes in the estimate. This reduces the cost function while satisfying the constraints of the problem. Haug also assumes that displacement constraints must only be at the nodal points.

Pickett [9] uses a design variable linking technique to reduce the number of variables for a one-constraint problem. This technique allows the computer to solve large systems, and also reduces the computer time.

Romstad [11] presents a practical technique to design efficient structures with the aid of digital computers. Solving a two-constraint problem (stress and displacement), the procedure is iterative in nature and the computer program is fundamentally based on the method of matrix structural analysis. He formulates a linear programming problem where weight is the criterion of merit, and where limits on stress and displacement are derived to represent the required multiplier in each of the design parameters from a previous acceptable solution to minimize the weight of the system.

Chapter II discusses the allowable deflection criterion. The development of the procedure used in this study is presented in Chapter III, which includes the algorithm for the displacement sensitivity derivatives and the sensitivity

matrix required for computing the incremental changes in the design variables in each iteration. The incremental changes are used to compute the new design variables (based on Newton's method). Chapter IV presents the convergence criteria for the different cases. Also, a comparison is made between two different sets of dependent section property equations. Conclusions are presented in Chapter V. A review of the matrix displacement method is presented in Appendix A. Appendix B presents the program listing.

2.0 ALLOWABLE DEFLECTIONS

Methods for computing deflections of structural members have been available for many years, yet they have not yielded uniformly consistent results because many factors affecting their true magnitudes have been inadvertently ignored. Fling [4] discusses such factors:

1. Concrete is assumed to carry no tension, which is practically incorrect.
2. Permanent and transient loads should be used instead of those assumed in strength computations. For instance, the applied live load is often less than the load used for design.
3. Actual moments should be used instead of those computed in the design for strength, particularly when empirical moment factors have been used.
4. The sequence of loading during construction can drastically affect the subsequent deflections.

2.1 ALLOWABLE DEFLECTIONS

For a variety of reasons, the tendency of a structure to vibrate and experience excessive deflections can be unde-

sirable. These reasons can be classified in four categories [4]:

(1) Sensory Acceptability

This is a matter of personal opinion. Therefore, limits on deflection vary according to the culture and the main purpose of the structure.

(2) Serviceability

Since deflection limits for serviceability of the structure are related to the intended use of this structure, they can be easily defined.

(3) Effect on Nonstructural Elements

The movements of nonstructural elements (walls and ceilings) should be limited to prevent cracking or other damage. Excessive deflection can prevent doors, folding partitions, and other movable elements from operating properly.

(4) Effect on Structural Elements

The movements of nonstructural elements could affect the structural elements. Therefore, those movements must be limited to prevent the structure from behaving differently than assumed in the design. However, if this cannot be achieved, deflection should be considered as part of the design for strength.

Linear movements can be limited as follows :

1. Setting an absolute value for the allowable deflection related to a fraction of the span or a fraction of the height, depending upon the reasons for the limitation.
2. Allowable deflection in some cases could be a function of the frequency of vibration or the rate of damping.

The most important factor that should be taken into consideration is the sequence of loading the structure. Loading the structure too early may result in larger deflections than those assumed in the design.

2.2 DEFLECTION COMPUTATIONS

This study considers only two types of deflections: beam deflection and column deflection. These are the most common and important deflections of the frame structure.

Gaylord [5] suggests that the allowable deflections be computed as follows :

Beam deflection = (1/80 to 1/360) . S (2.2.1)

Column deflection = (1/300 to 1/600) . H

where S is the beam span and H is column height. Also, the building drifting can be computed using equation 2.2.2 [2]:

$$\max \Delta 1 = 0.002 \cdot H \qquad (2.2.2)$$

$$\max \Delta 2 = 0.15 \text{ in. per story}$$

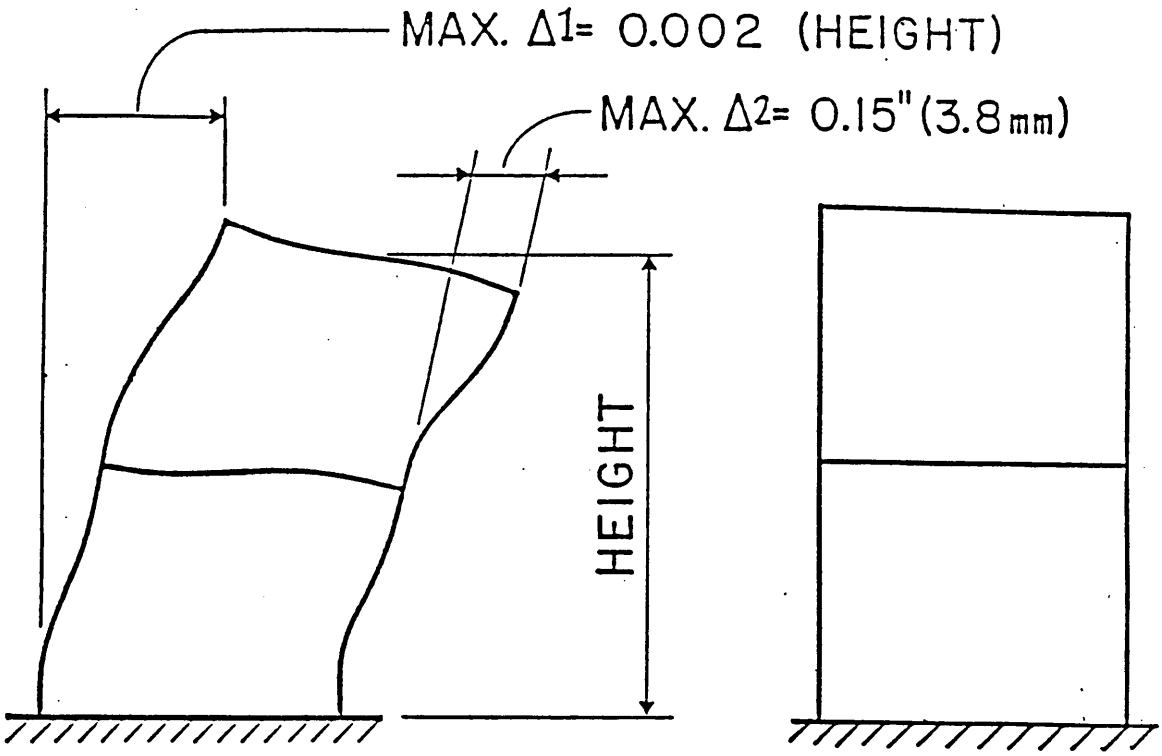
where $\Delta 1$ and $\Delta 2$ are as shown in figure 1.

For computing the column deflection, the two methods are found to be almost the same. For computing beam deflection, the maximum deflection at the middle of the span is considered.

If the beam member is divided into two equal elements, the central lateral deflection of the beam is a function of the vertical joint displacements at the end of the elements. The rule for computing the beam deflection is as follows :

1. Determine the vertical end displacement for each end of each element in the member.
2. Subtract half of the far end vertical displacement for each element from the vertical displacement of the middle joint of the member.
3. The allowable deflection is considered to be 1/80 to 1/360 times the beam span.

For column deflection the element is considered to be a complete member. The deflection in the horizontal direction is considered. The allowable column deflection is considered to be 1/300 to 1/600 times the height.



FRAMED BLDG.

Figure 1. Deflections for frame structure

2.3 OBSERVATIONS OF THE ALLOWABLE DEFLECTIONS

Because of the assumed deformed shape of the structure (due to several loading conditions), the allowable deflection for each member (beam or column) may be completely different.

1. Since the higher the building, the larger the deflection at the top, the deflection of the column members increases with the height. Therefore, the allowable deflection for the higher floors should be more the that deflection for the lower floors. This results in consistency in the deformed shape of the building, and also speeds up the convergence.
2. The allowable deflection should be chosen neither too large nor too small.

3.0 SOLUTION ALGORITHM

3.1 GENERAL PROCEDURE

A computer program has been developed for use in this study (the analysis part written by Holzer [7]; see Appendix B). Figure 2 shows the program flow chart. The analysis method used in this study is the matrix displacement method (see Appendix A). For a detailed description of the method, see Holzer [7].

3.2 MATRIX DISPLACEMENT METHOD

This method is used because of its easy implementation to the computer. Some assumptions have been made to simplify the procedure:

1. The analysis is limited to plane skeletal structures with one-dimensional axial and flexural elements having six degrees of freedom.
2. The frame structure is rigidly connected and experiences several loading conditions.
3. Loading conditions are assumed to be of a static nature.

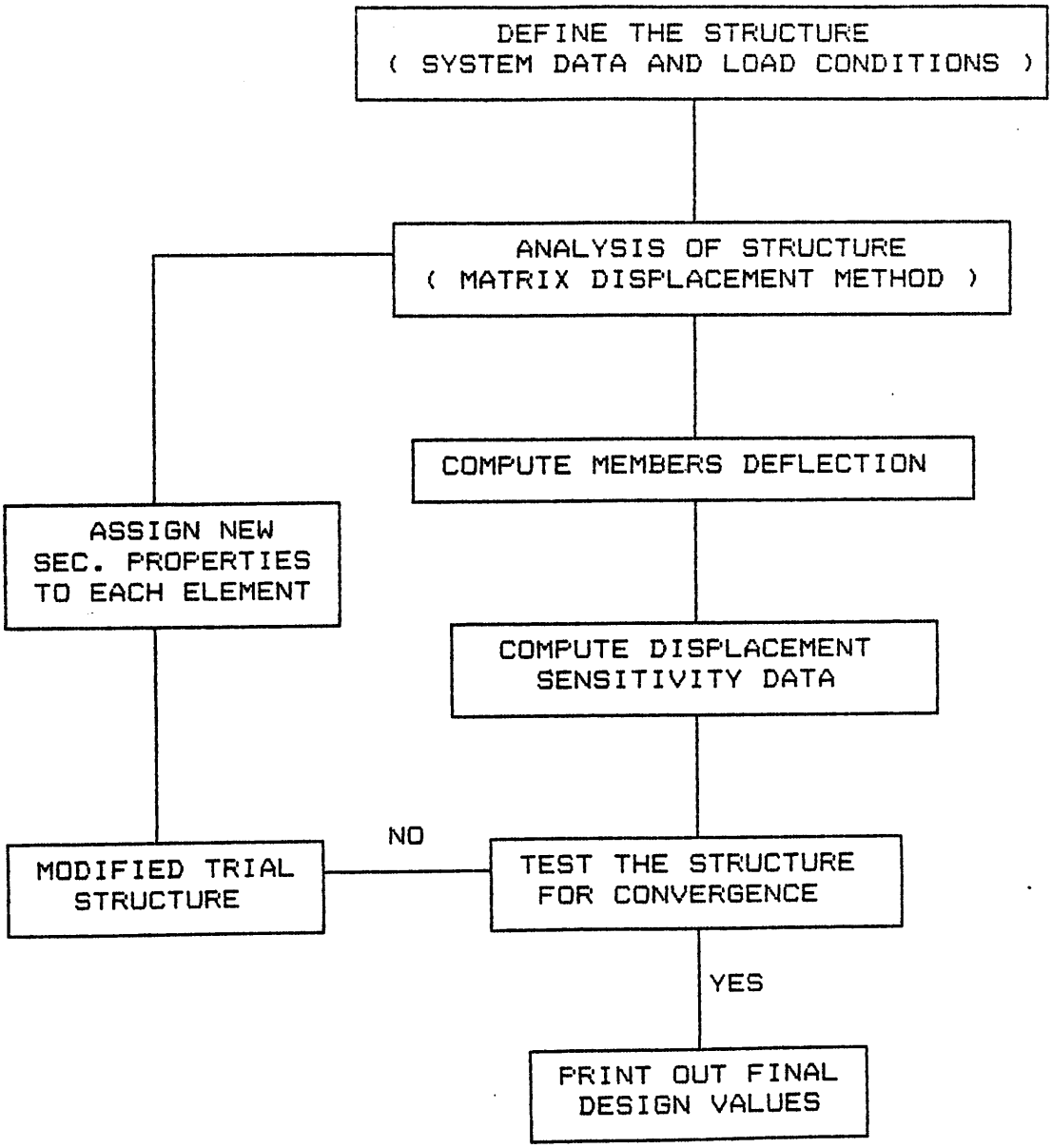


Figure 2. Flow Chart

4. The material has a linear elastic behavior.

The matrix displacement method is developed from the element model to the system model in the following sequence:

(a.) The local element model :

$$\underset{(6,1)}{f^i} = \underset{(6,6)}{k^i} \cdot \underset{(6,1)}{d^i} \qquad (3.2.1)$$

where f^i= element end forces

k^i= local element stiffness matrix

d^i= element end displacements

(b.) The global element model :

$$\underset{(6,1)}{F^i} = \underset{(6,6)}{K^i} \cdot \underset{(6,1)}{D^i} \qquad (3.2.2)$$

$$K^i = \Lambda^{iT} \cdot k^i \cdot \Lambda^i \qquad (3.2.3)$$

where F^i= global element end forces

K^i= global element stiffness matrix

D^i= global element end displacements

(c.) The generalized element model :

$$\underset{(n,1)}{F^{(i)}} = \underset{(n,n)}{K^{(i)}} \cdot \underset{(n,1)}{q} \qquad (3.2.4)$$

$$K^i \xrightarrow{M^i} K^{(i)}$$

$$F^i \xrightarrow{M^i} F^{(i)}$$

where $F^{(i)}$ = generalized element end forces vector

$K^{(i)}$ = generalized element stiffness matrix

q = generalized joint displacements

$M^{(i)}$ = member code global-system transformation

n = number of degrees of freedom

ne = number of elements in the structure

(d.) The system model :

$$K = \sum_{i=1}^{ne} K^{(i)}$$

$$\underset{(n,n)}{K} \cdot \underset{(n)}{q} = \underset{(n)}{Q} \qquad (3.2.5)$$

where K = the system stiffness matrix

Q = generalized external force vector

The details of each stiffness matrix are shown in Appendix A.

3.3 DEVELOPMENT OF DISPLACEMENT DERIVATIVES

Kirsch [8] describes the structural system as a set of quantities where some are fixed and others are variable. The fixed parameters that define the structural system are called preassigned parameters. The independent variable parameters, not preassigned, are called design variables. The design variables, for instance, could represent:

(1) The mechanical properties of the material.

(2) The topology of the structure.

(3) The section modulus.

(4) The moment of inertia.

(5) The cross-sectional area.

Throughout this study, the design variables represent the section modulus values, referred to as the X vector.

3.3.1 GENERALIZED DISPLACEMENT DERIVATIVES

The approach used in this study is called the design state method [8]. Before discussing the development of this approach, the variable dependency is presented. The variables are defined below :

1. Section properties :

   A = Area, which is a dependent variable.

   I = Moment of inertia, which is a dependent variable.

   x =Section modulus, which is the independent variable.

2. Stiffness matrices :

   a. Local element k :

      The local element stiffness matrix is a function of the modulus of elasticity E, the element length L, area A, and moment of inertia I:

$$k^i = k^i(E, L, A, I) \qquad (3.3.1)$$

b. Global element K :

The global element matrix K is also a function of the sine (C1) and cosine (C2) directions of the element, as well as a function of all previous parameters:

$$K^i = K^i(E, L, C1, C2, A, I) \qquad (3.3.2)$$

c. System K :

$$K = \sum_{i=1}^{ne} K^{(i)} \qquad (3.3.3)$$

where

$$K^{(i)} = K^{(i)}(E, L, C1, C2, A, I) \qquad (3.3.4)$$

and :

$$K = K (E, L, C1, C2, A, I) \qquad (3.3.5)$$

In $K^{(i)}$, the parameters A and I depend only on x^i of element i, but in system K they depend on all x^i (i = 1, 2,..n). Also, the only variable parameters in K are A and I.

3. Load Vector :

$$Q = \bar{Q} - \overset{\Delta}{Q} \qquad (3.3.6)$$

where :

Q= equivalent joint force vector

$\bar{Q}$= applied joint force vector

$\overset{\Delta}{Q}$= constrained joint force vector

$$\overset{\Delta}{Q} = \overset{\Delta}{Q}\ (\ L,\ x1,\ x2,\ p\ ) \qquad (3.3.7)$$

$$Q = Q\ (\ \bar{Q},\ L,\ x1,\ x2,\ p\ ) \qquad (3.3.8)$$

where :

p = element action

x1 = Load distance from a-end

x2 = Load distance from b-end

The parameter A is not considered in the function Q because dead load is not considered.

4. Response :

$$\{q\} = [K]^{-1}.\ \{Q\} \qquad (3.3.9)$$

From equation 3.3.5 and equation 3.3.8, the generalized displacement vector q is a function, in general, of the following :

$$q = q\ (\ E,\ L,\ C1,\ C2,\ A,\ I,\ Q) \qquad (3.3.10)$$

where the only parameters that depend on the design variables are the areas A and the moments of inertia I.

The design state approach is developed as follows : let the function G represents the system model

$$\underset{(n,1)}{G} = \underset{(n,n)}{K} \cdot \underset{(n,1)}{q} - \underset{(n,1)}{Q} = 0 \qquad (3.3.11)$$

According to equation 3.3.5 and equation 3.3.10, the system stiffness matrix K and displacement vector q are each functions of the areas A and the moments of inertia I. Differentiating G with respect to the design variable x (W.R.T. x) for member j results in equation 3.3.12 :

$$\frac{\partial G}{\partial x_j} = \frac{\partial (K.q)}{\partial x_j} - \frac{\partial Q}{\partial x_j} \qquad (3.3.12)$$

and $$\frac{\partial Q}{\partial x_j} = 0 \qquad (3.3.13)$$

because Q is not a function of the design variable. Equation 3.3.12 can be rewritten as :

$$\frac{\partial G}{\partial x_j} = K \cdot \frac{\partial q}{\partial x_j} + \frac{\partial K}{\partial x_j} \cdot q \qquad (3.3.14)$$

$$K \cdot \frac{\partial q}{\partial x_j} = - \frac{\partial K}{\partial x_j} \cdot q \qquad (3.3.15)$$

Realizing that the system stiffness matrix K is the sum of the generalized element stiffness matrices,

$$K = K^{(1)} + K^{(2)} + \ldots + K^{(ne)} \qquad (3.3.16)$$

it is clear that

$$\frac{\partial K}{\partial x_j} = \frac{\partial K^{(1)}}{\partial x_j} + \frac{\partial K^{(2)}}{\partial x_j} + \ldots \frac{\partial K^{(ne)}}{\partial x_j} = \sum_{i=1}^{ne} \frac{\partial K^{(i)}}{\partial x_j} \qquad (3.3.17)$$

$K^{(i)}$ is a function of area (A^i) and moment of inertia (I^i) for that element only. Therefore,

$$\frac{\partial K}{\partial x_j} = \frac{\partial K^j}{\partial A_j} \cdot \frac{\partial A^j}{\partial x_j} + \frac{\partial K^j}{\partial I_j} \cdot \frac{\partial I^j}{\partial x_j} \qquad (3.3.18)$$

$\partial A^i/\partial x_j = 0$, $\partial I^i/\partial x_j = 0$ for i=j

Thus, the matrix $\partial K^{(i)}/\partial x_j$ has value only when i=j

$$\frac{\partial K^i}{\partial x_j} = \frac{\partial K^i}{\partial x_i} \qquad (3.3.19)$$

To compute $\partial K^{(j)}/\partial A_j$ and $\partial K^{(j)}/\partial I_j$, the coefficients of the element stiffness matrix, g1-g7, are differentiated with respect to A^j and I^j respectively. The coefficients and the differentiations are shown in Appendix A [10].

To compute the section property derivatives of equation 3.3.18, two sets of section property functions are differentiated with respect to the design variable. The first set is Traynor's equations [12]; the second is Brown and Ang's equations [3] (see Chapter V for comparison of the two sets). Tables 1 and 2 present the equations and their derivatives respectively.

Solving equation 3.3.15 gives the following expression:

$$\frac{\partial q}{\partial x_j} = - K^{-1} . \frac{\partial K}{\partial x_j} . q \qquad (3.3.20)$$

The left side term could be represented as follows :

$$\frac{\partial q}{\partial x}_j = \begin{bmatrix} \partial q_1/\partial x_1 \cdots\cdots\cdots \partial q_1/\partial x_{ne} \\ \vdots \qquad\qquad\qquad \vdots \\ \partial q_n/\partial x_1 \cdots\cdots\cdots \partial q_n/\partial x_{ne} \end{bmatrix} \qquad (3.3.21)$$

3.3.2 DESIGN VARIABLE LINKING

The design displacements are the displacement quantities associated with each structural member to be used in the application of the displacement controlled design. Two types

of members are defined : beams (horizontal members) and columns (vertical members). Each beam is composed of two elements and the design displacement is the local central lateral displacement. Each column is composed of one element and the design displacement is the difference in the lateral displacements at the ends (relative lateral displacement).

The matrix displacement analysis is concerned with elements and points, where joints are the connection joints of the elements. If, as in the case for beams, the structural member is composed of more than one element, then these elements must be linked together so that the computer understands that they compose one member, which has one controlling design displacement. In addition, the entire member has only one design variable; i.e., each element of the set is identical. The algorithm required to control these relationships between elements and structural members is called variable linking [8]. Each of the design displacements can be related to the generalized structural displacements q.

Equation 3.3.22 represents the relationship between the element design variable and the member design variable.

$$\underset{(ne)}{\{x\}} = \underset{(ne,m)}{[\,T\,]} \cdot \underset{(m)}{\{X\}} \qquad (3.3.22)$$

where

x= element design variable vector.

X= member design variable vector

T= design variable transformation matrix.

m= number of members (groups)

Equation 3.3.22 for case I (page 36) can be expressed as :

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

which represents the following relationships :

$x_1 = X_1$

$x_2 = X_2$

$x_3 = X_2$

$x_4 = X_3$

Equation 3.3.20 gives the change of the generalized displacements q with respect to the change in the design variable for element j (x_j).

Sice the element design variables are functions of the member design variables, and the generalized displacements are functions of element design variables, see equation 3.3.10; There are two sets of functions as in equation 3.3.23 and equation 3.3.24 :

$$q_1 = f_1 (x_1 \ldots\ldots\ldots x_{ne})$$
$$\vdots \qquad\qquad (3.3.23)$$
$$q_n = f_n (x_1 \ldots\ldots\ldots x_{ne})$$

$$q_1 = F_1 (X_1 \ldots\ldots\ldots X_m)$$
$$\vdots \qquad\qquad (3.3.24)$$
$$q_n = F_n (X_1 \ldots\ldots\ldots X_m)$$

where

x= element design variable.

X= member design variable.

The partial derivatives of these composite functions can be obtained by the chain rule.

$$\frac{\partial q_k}{\partial X_j} = \sum_{i=1}^{ne} \frac{\partial q_k}{\partial x_i} \cdot \frac{\partial x_i}{\partial X_j} \qquad (3.3.25)$$

Equation 3.3.25 can be expressed concisely in matrix form. The partial derivatives $\partial q/\partial X_j$ are the entries in the n × m matrix

$$(\partial q/\partial X_j) = \begin{bmatrix} \partial q_1/\partial X_1 & \cdots & \partial q_1/\partial X_m \\ \vdots & & \vdots \\ \partial q_n/\partial X_1 & \cdots & \partial q_n/\partial X_m \end{bmatrix} \qquad (3.3.26)$$

The formulas involve two other matrices; one is equation 3.3.21 and the other is equation 3.3.27 :

$$(\partial x_i/\partial X_j) = \begin{bmatrix} \partial x_1/\partial X_1 & \cdots & \partial x_1/\partial X_m \\ \vdots & & \vdots \\ \partial x_n/\partial X_1 & \cdots & \partial x_n/\partial X_m \end{bmatrix} \qquad (3.3.27)$$

Realizing that the derivatives of x with respect to X to compose the matrix in equation 3.3.27 are the same as the T matrix. Therefore, the T matrix is used instead of equation 3.3.27 throughout the program. Equation 3.3.26 represents the change in the generalized displacement for member i with respect to the change in the design variable for member j.

3.3.3 DESIGN DISPLACEMENT TRANSFORMATION

This study considers only one design displacement per member (see Chapter II). The transformation matrix A is needed to relate the joint displacements to the member design displacements as shown below :

$$\{e\} = [\ A\ ] \cdot \{q\} \qquad (3.3.28)$$
$$(m) \quad (m,n) \quad (n)$$

where :

e = member design displacement vector

A = member displacement transformation matrix.

q = joint displacement vector.

Differentiating e with respect to the member design variable X results in equation 3.3.29 :

$$\frac{\partial e^i}{\partial X_j} = \frac{\partial (A.q)}{\partial X_j} = A \cdot \frac{\partial q}{\partial X_j} + \frac{\partial A}{\partial X_j} \cdot q \qquad (3.3.29)$$

The last term is equal to zero, since A is a constant; therefore, equation 3.3.29 becomes :

$$\underset{(m,1)}{\frac{\partial e^i}{\partial X_j}} = \underset{(m,m)}{A} \cdot \underset{(n,1)}{\frac{\partial q}{\partial X_j}} \qquad (3.3.30)$$

Equation 3.3.30 is the change in the design displacement of member i due to a change in the design variable of member j :

$$(\partial e/\partial X) = \begin{bmatrix} \partial e^1/\partial X_1 & \cdots & \partial e^1/\partial X_m \\ \vdots & & \vdots \\ \partial e^m/\partial X_1 & \cdots & \partial e^m/\partial X_m \end{bmatrix} \qquad (3.3.31)$$

To compute the incremental change in the design variable, (ΔX), the first approximation for a single design displacement can be defined as :

$$\Delta e^i = \frac{\partial e^i}{\partial X_1} \cdot \Delta X_1 + \frac{\partial e^i}{\partial X_2} \cdot \Delta X_2 + \cdots + \frac{\partial e^i}{\partial X_m} \cdot \Delta X_m \qquad (3.3.32)$$

Equation 3.3.32 can be expressed in matrix form as follows :

$$\underset{(m,m)}{\frac{\partial e}{\partial X}} \cdot \underset{(m)}{\Delta X} = \underset{(m)}{\Delta e} \qquad (3.3.33)$$

$$\begin{bmatrix} \partial e^1/\partial X_1 & \partial e^1/\partial X_m \\ \partial e^m/\partial X_1 & \partial e^m/\partial X_m \end{bmatrix} \begin{bmatrix} \Delta X_1 \\ \Delta X_m \end{bmatrix} = \begin{bmatrix} \Delta e^1 \\ \Delta e^m \end{bmatrix} \qquad (3.3.34)$$

where Δe = vector of all design displacement changes

ΔX = vector of all design variable changes

and $\partial e/\partial X$ is the displacement sensitivity matrix. In this set of equations the ΔX represents the unknowns.

Using Newton's method, equation 3.3.35 represents the new set of design variables.

$$\underset{(m)}{X^{k+1}} = \underset{(m)}{X^k} + \underset{(m)}{\Delta X} \qquad (3.3.35)$$

where k is the iteration number.

3.3.4 CONVERGENCE TEST

The program tests the system for convergence using the deflection ratio as follows :

$$DR = e^i / e^i_a$$

where DR = displacement ratio.

$$e^i - e^i_a = 0$$

$$\frac{e^i - e^i_a}{e^i_a} = DR - 1 = 0 \qquad (3.3.36)$$

Setting some tolerance ε, equation 3.3.36 becomes

$$|DR - C| \leq \varepsilon \qquad (3.3.37)$$

where $C = 1 - \varepsilon$

If the convergence test fails, a new trial structure is computed using the new design variable and one of the two sets of section property functions. To assign the new computed member properties to each corresponding element to start a new iteration, a relation between elements in each group (member) is developed (see Nassis-Schneiderman diagram for Subroutine Assign).

3.4 MULTIPLE LOADING CONDITIONS

For a structure under multiple loading conditions, there is a unique response associated with each loading condition. Therefore, to predict the structural response to an isolated loading condition, one must examine the sensitivity matrix for that loading condition only.

However, in this algorithm the design variable for a member is determined by the maximum deflection for this member under multiple loading conditions. Likewise, the controlling deflection sensitivities for each member are determined by the controlling loading condition that caused the maximum deflection for that member. Therefore, the sensitivity matrix consists of a composite of the controlling sensitivities for each member. Thus, the composite sensitivity matrix for a structure under multiple loading conditions is equivalent to the sensitivity matrix for a structure under a single loading condition. The rule for constructing the composite matrix is as follows :

1. Determine the critical loading condition that causes the maximum deflection for each member.
2. From the sensitivity matrix for the coressponding critical loading condition for each member i, retain $\partial e^i/\partial X_j$ to construct the composite sensitivity matrix, where $\partial e^i/\partial X_j$ represents row i.

Table 1. Traynor's Equations and their Derivatives

$[\ 0 < S < 1100\ in^3]$

$A=3.62415+(0.119637).S-(9.70882\times10^{-5}).S^2+(5.34160\times 10^{-8}).S^3$

$\partial A/\partial S=0.119637-(19.41765\times10^{-5}).S+16.02478\times10^{-8}.S^2$

$[\ 0 < S < 50\ in^3]$

$I=(7.98)\ .\ S$

$\partial I/\partial S=7.98$

$[\ 50\ in^3 < S < 1100\ in^3\ ]$

$I=-191.3096+(10.90819).S+(0.01840775).s^2-(1.076241\times 10^{-5}).S^3$

$\partial I/\partial S=10.90819+(0.0368155).S-3.228723\times10^{-5}\times S^2$

where S = section modulus

Table 2. Brown and Ang's Equations and their Derivatives

$[\ 0 < S < 503\ in^3\ ]$

$A=0.464\ [((290+S)^2 - 84100)/60.6]^{0.5}$

$\partial A/\partial S=0.0298025\ (2S+580)/(S^2+580S)^{0.5}$

$[\ 503\ in^3 < S < 1113\ in^3\ ]$

$A=(18.5111\ S + 1988.9336)/256$

$\partial A/\partial S=0.072309$

$[\ 0 < S < 503\ in^3\ ]$

$I=((290+S)^2-84100)/60.6$

$\partial I/\partial S=(S+290)/30.3$

$[\ 503\ in^3 < S < 1113\ in^3\ ]$

$I=18.5111\ S - 311.0664$

$\partial I/\partial S= 18.5111$

where S = section modulus

4.0 EXAMPLES AND RESULTS

4.1 GENERAL CONSIDERATIONS

The purpose of this chapter is to demonstrate the procedure and compare the convergence characteristics for several examples under multiple loading conditions, and to compare the results of the two sets of section property functions mentioned earlier (see tables 1 and 2).

Four cases with different section properties and two different sets of trial structure equations have been conducted in this study. The convergence tolerance for all cases is one percent(EPS=0.01). Other necessary information, such as initial section properties, number of iterations, etc., is provided with each case. Table 3 shows the number of iterations for each case.

4.2 CASE STUDY I : SINGLE STORY PORTAL FRAME

The single story portal frame was analyzed under three loading conditions. Figure 3 shows the frame layout, as well as joints, elements, and member locations. Figure 4 shows the three loading conditions, which are the same for all tested cases.

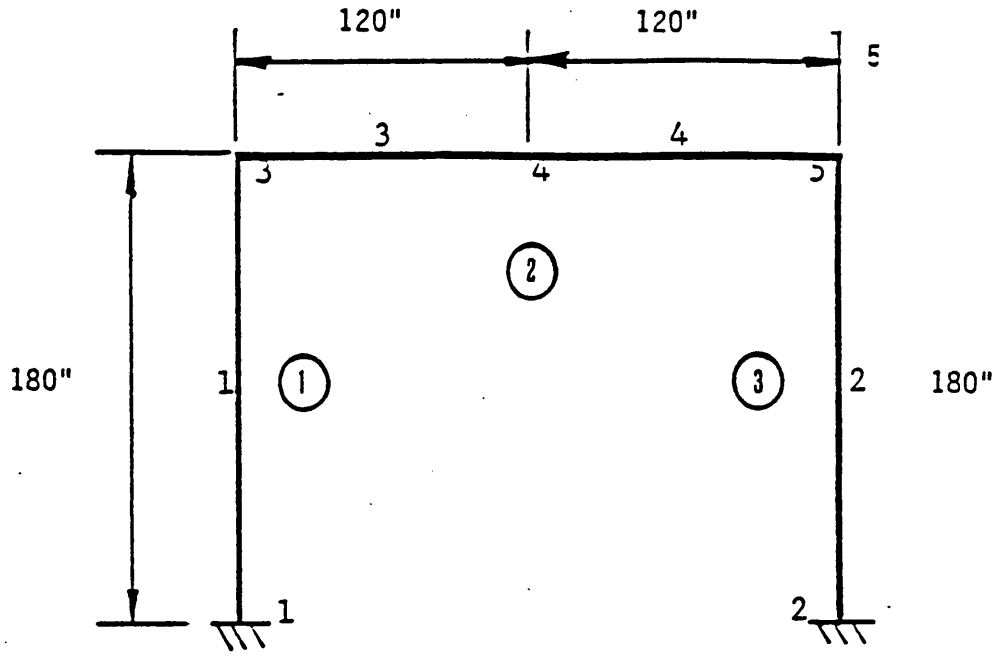


Figure 3. Member and Joint Locations for Case Study I

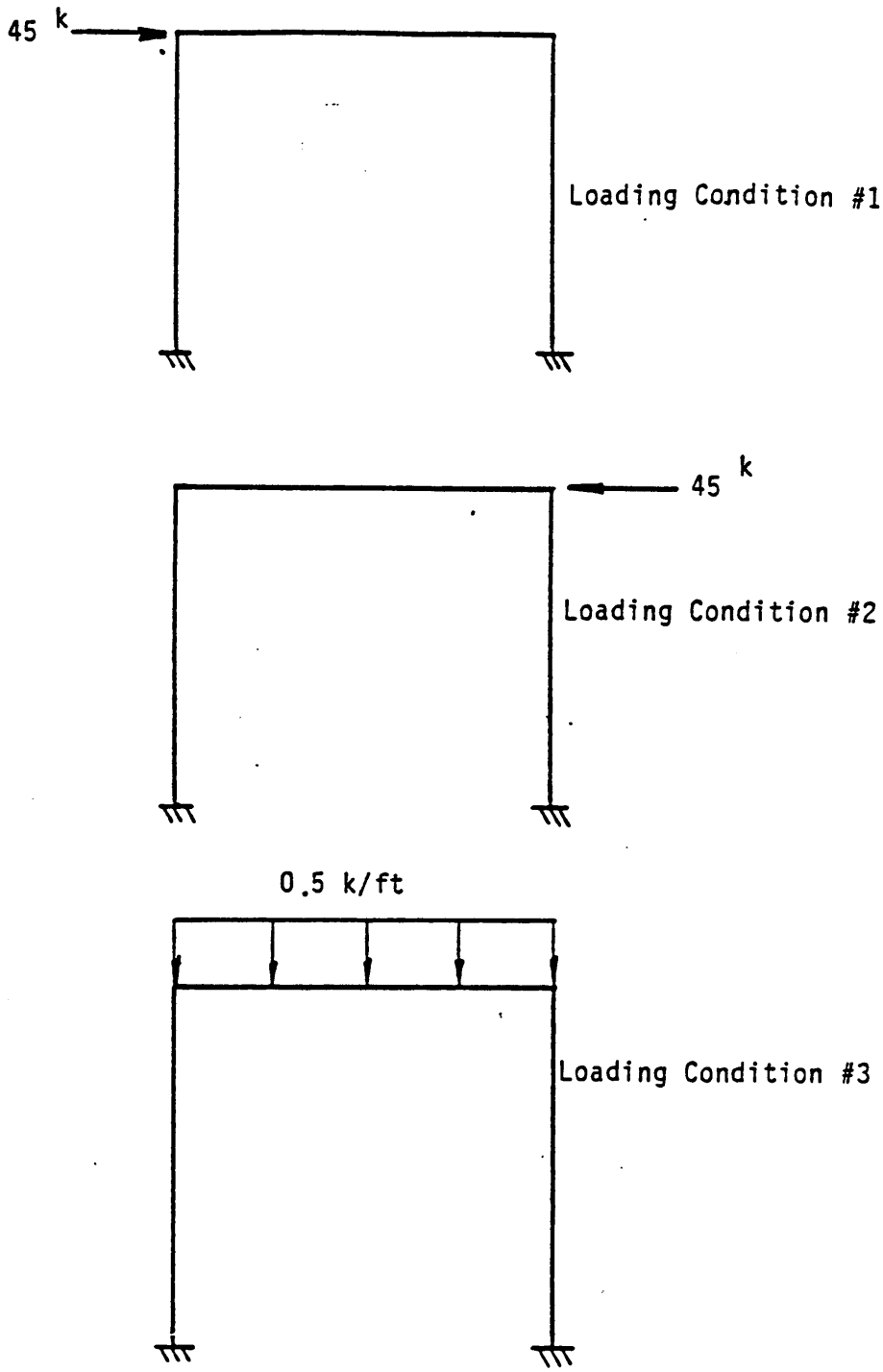


Figure 4. Loading Conditions for Case Study I

The frame was tested under three different initial section property values. The first test (I) was conducted for very small initial values; the second (II) for initial values which were close to the final design values; the third (III) for large initial values. All tests were conducted under two different sets of section property functions. Table 4 and table 5 show the initial starting values and the final design values respectively.

The first loading condition is a wind load applied to the top left joint in a positive direction with a value of 45 k. The second one is also a wind load but applied to the top right joint in a negative direction with value of -45 k. The third loading condition is a distributed load of -0.5 k/in applied to the beams. As shown in table 5, the structure under three loading conditions collectively yields a symmetric design condition.

As shown in table 3, the structure tends to converge faster under Traynor's equations for the three different initial design values. Also, there was not much change in the number of iterations under Traynor's equations when starting with small or large values. However, with initial values close to the final values, it is clear that the number of iterations was cut down significantly. On the other hand, for Ang's equations there was a drop from 15 iterations to 10 iterations when starting with large values instead of small values.

The results for the maximum deflections under the three loading conditions and the results for the final (allowable) deflections are shown in table 6. When the structure was analyzed for the first iteration, the deflection for the column members was 1.11 inch and for the beam member was 2.515 inches. After three iterations, the deflections were almost equal to the allowable ones. Also, the convergence indicator was near the convergence point after the third iteration.

Two displacement sensitivity matrices (for the first, the third, and final iterations respectively) are shown in table 6. It has been noticed for this case that the sensitivity matrix after the third cycle of iteration represents a fairly accurate estimate of the final matrix because it changes slightly with each successive iteration after the third one.

The structure was also tested under the displacement ratio modification rule (analogous to the stress ratio modification rule) which computes the new design variable according to the deflection ratio (DR) instead of using Newton's method. Table 3 shows no change in number of iterations under this method.

4.3 CASE STUDY II : TWO STORY PORTAL FRAME

The test for this case was conducted under the same loading conditions and the same initial design values as case one.

The structure and the loading conditions are shown in Figures 5 and 6 respectively. Also, table 7 shows the final design values for the case. For initial design values close to the final values, the test shows little difference in number of iterations when using any set of section property functions. But for small initial values the system tends to converge faster under Ang's equations, and for large initial values the system tends to converge faster under Traynor's equations.

The sensitivity matrices shown in table 8 are for the first and the final iterations respectively, using both sets of equations. The maximum deflections under all loading conditions and the final deflections are shown in table 9.

The case was also tested for convergence using the displacement ratio modification rule. But for this case the system did not converge after 45 iterations.

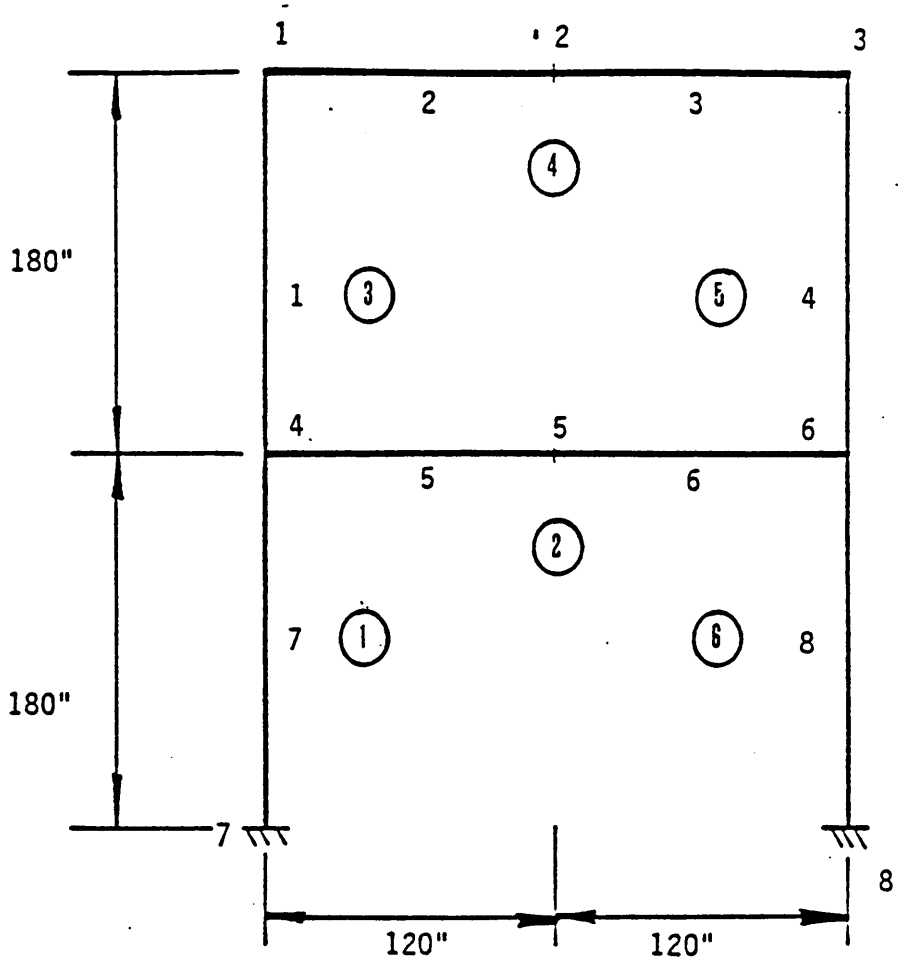


Figure 5. Member and Joint Locations for Case Study II

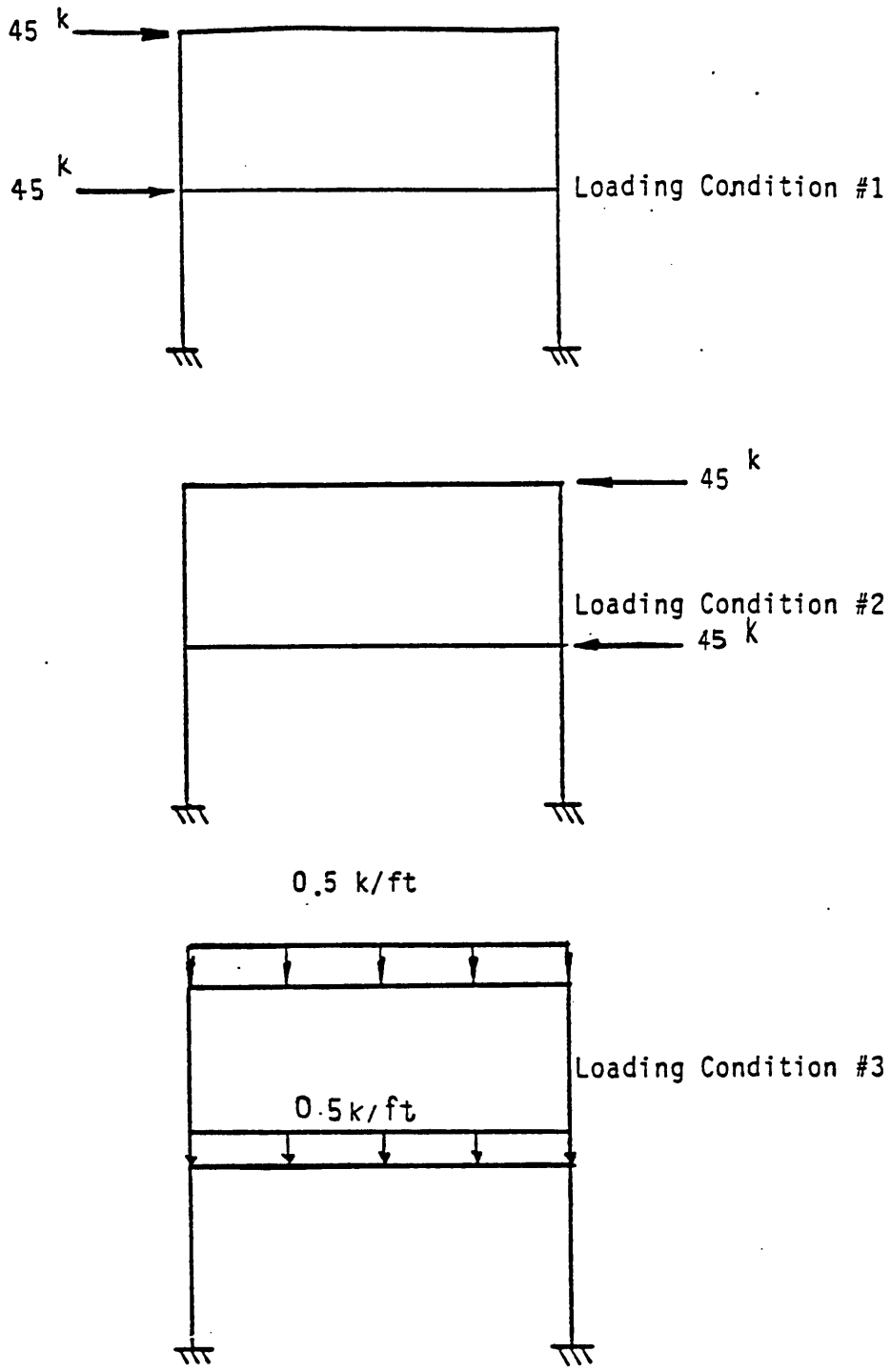


Figure 6. Loading Conditions for Case Study II

4.4 CASE STUDY III : THREE STORY PORTAL FRAME

The same analysis procedure and loading conditions were used for this case. The structure and the loading conditions are shown in figures 7 and 8 respectively.

In this case the system converged under Ang's equations, but not under Traynor's equations or under the displacement ratio modification rule. Traynor's equations are apparently not valid when the design variables are larger than the maximum section modulus in the range specified (see tables 1 and 2) and give negative values for the area and moment of inertia. This results in a condition of singularity in the stiffness matrix because some of the stiffness coefficients are negative.

This problem was encountered only in case II and case III. To overcome it, an upper bound for the design variables was set to be 1100 in^3. If the new computed design variable is greater than this bound, the design variable is reduced to a value of 600 to start a new iteration. This modification did not work for case III because the structure was large. The reason for nonconvergence is that the limit on the design variables does not allow the incremental design variable, ΔX, to change freely, as dictated by the behavior constraints and the sensitivity matrix for each iteration. So, each iteration cuts down the change in the design variable to a constant

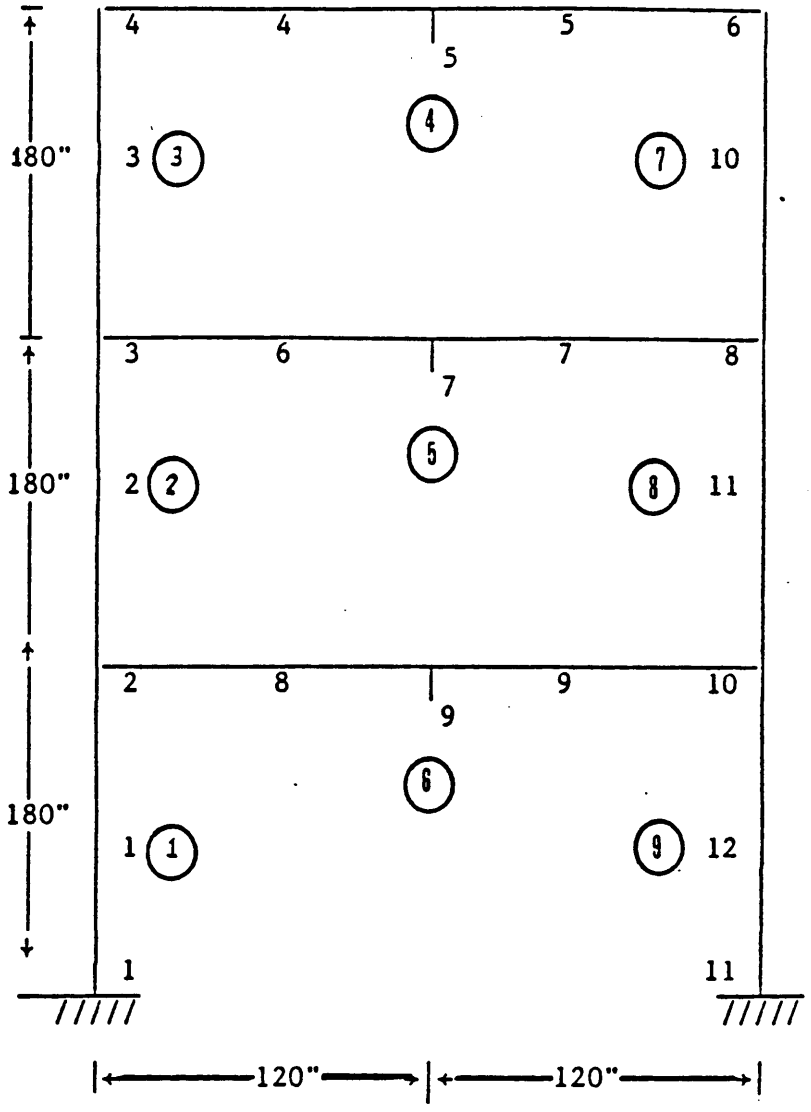


Figure 7. Member and Joint Locations for Case Study III

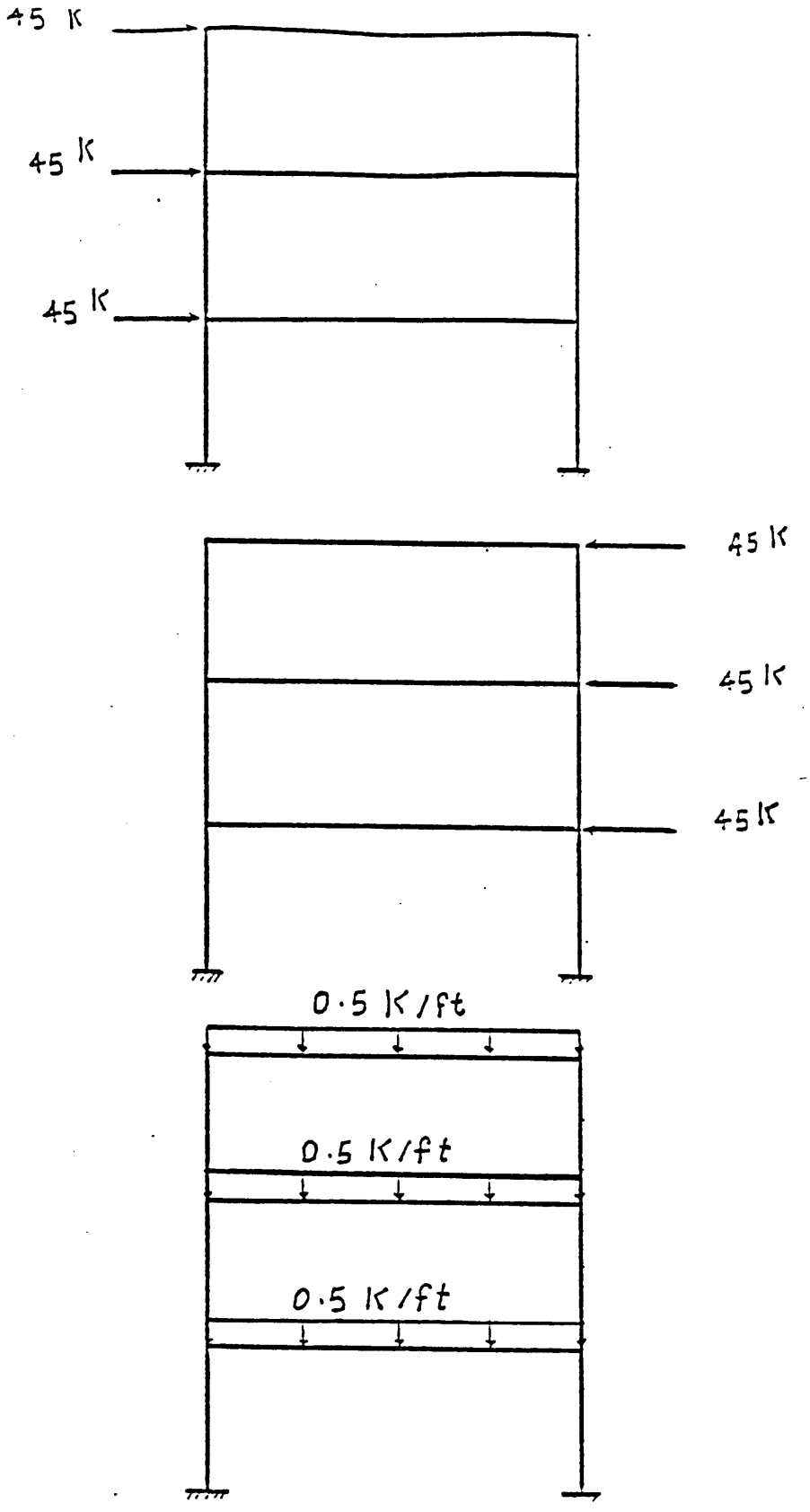


Figure 8. Loading Conditions for Case Study III

value, which results in an increase in the number of iterations and gives the nonconvergent condition.

Also, for this case the sensitivity matrix after the third cycle of iteration represents a fairly accurate estimate of the final matrix.

4.5 CASE STUDY IV : TWO BAY PORTAL FRAME

The same loading conditions and the same initial design values were used. The results show that the structure converged faster under Traynor's equations. Figures 9 and 10 show the structure and the loading conditions respectively. Table 10 shows the final design values for the structure. Table 11 shows the initial and the final member deflection.

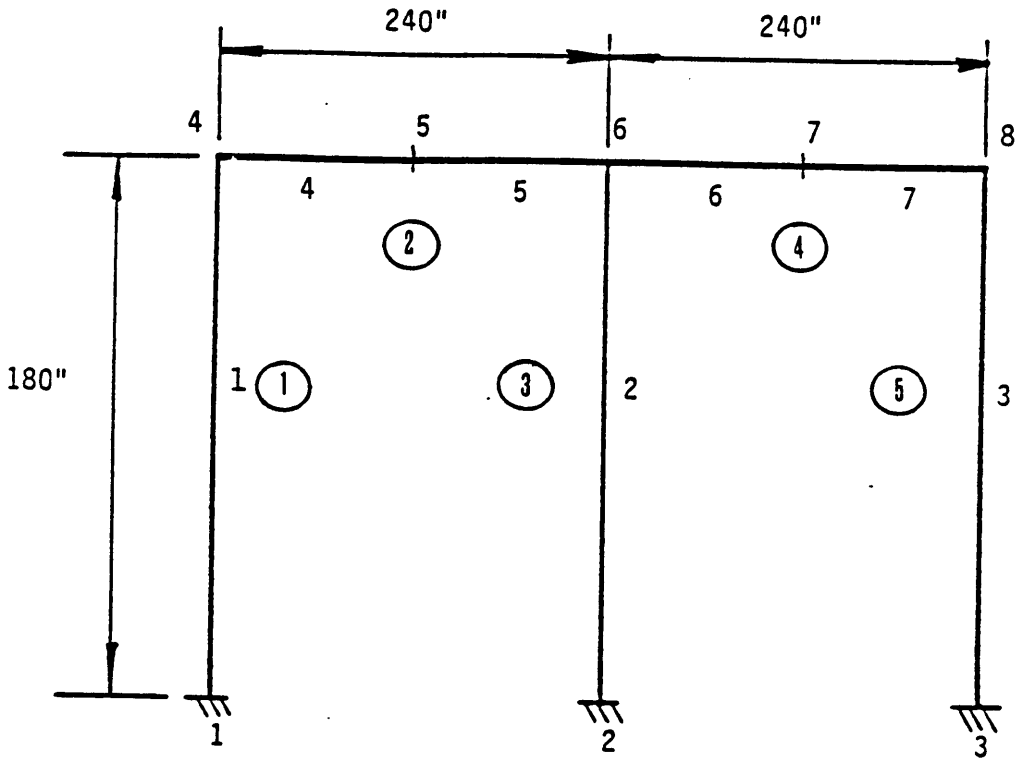


Figure 9. Member and Joint Locations for Case Study IV

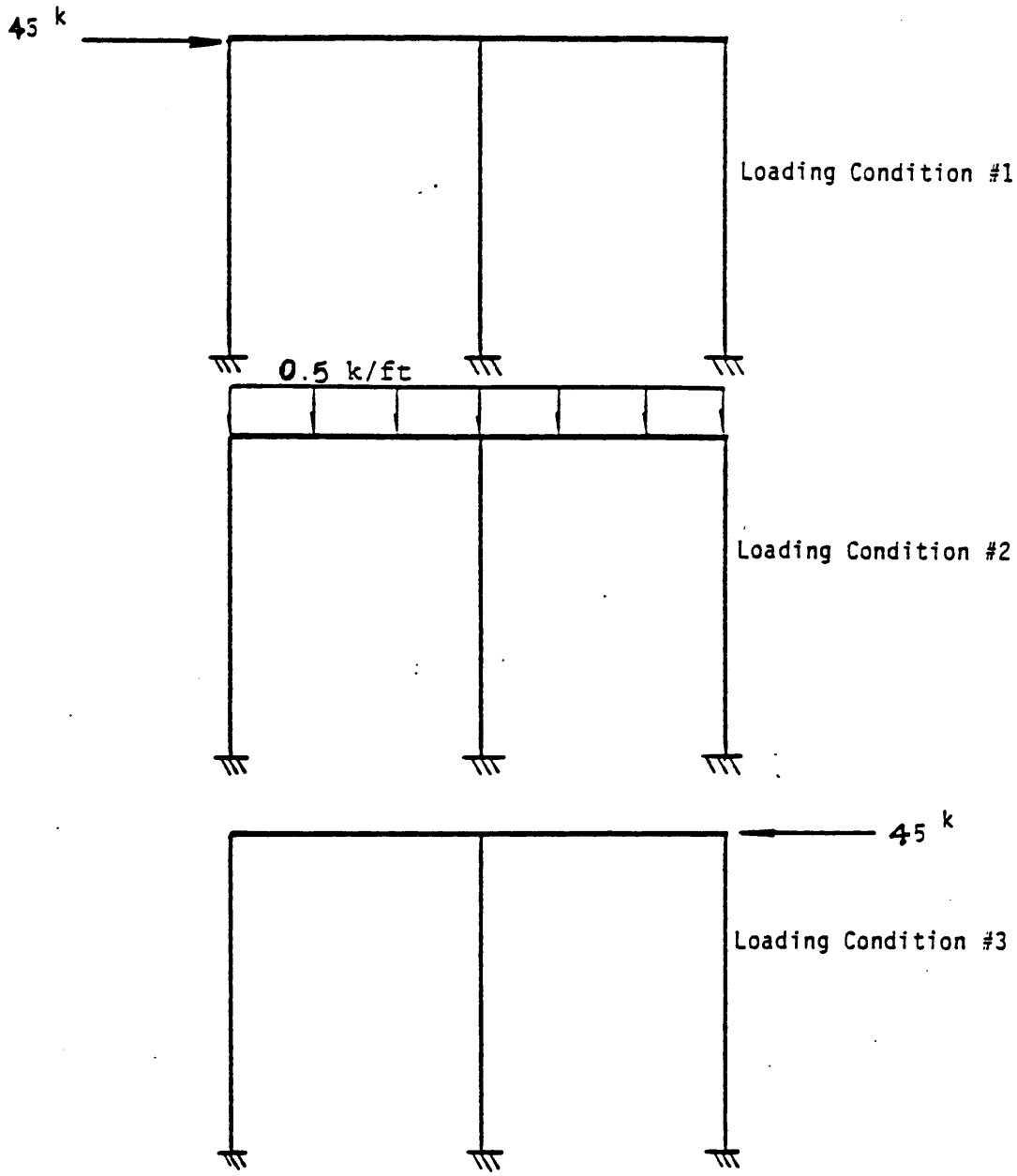


Figure 10. Loading Conditions for Case Study IV

Table 3. Iteration Numbers For All Cases Under Both Sets

Sec. prop.	One story		Two story		Three st.	
	ANG	TRAY	ANG	TRAY	ANG	TRAY
I	15	13	13	17	14	
II	6	6	10	9	17	
III	10	12	17	10	20	
Two Bay I	17	6				
II	13	6				
Modif. rule	6	6				

Table 4. Section Properties (Initial Values For All Cases)

Sec. prop.	A	I	S
I	0.5	8.0	1.0
II	10	800	70
III	100	10000	1000

Table 5. Final Design Values For Case I

Member	A		I		S	
	ANG	TRAY	ANG	TRAY	ANG	TRAY
1	23.05	23.20	2469.46	2470.25	193.47	189.89
2	19.78	19.54	1818.08	1818.13	150.77	149.70
3	23.06	23.21	2469.95	2470.63	193.51	189.92

Table 6. Deflection Values & Sensitivity Matrices - Case I

MEMBER.	Initial Defl	Final Defl.
1	1.1060	0.6000
2	-2.5158	-2.0000
3	-1.1106	-0.6000

$\partial e/\partial X_j$ =

-0.005651	-0.005354	-0.005408	
0.006554	0.024180	0.006554	First iteration
0.005408	0.005354	0.005651	

$\partial e/\partial X_j$ =

-0.001552	-0.002971	-0.001441	
0.001641	0.024657	0.001640	Third iteration
-0.001442	0.002971	0.001552	

$\partial e/\partial X_j$ =

-0.001540	-0.002953	-0.001431	
0.001626	0.024632	0.001625	Final iteration
-0.001431	0.002953	0.001540	

Table 7. Final Design Values For Case II

Member	A		I		S	
	ANG	TRAY	ANG	TRAY	ANG	TRAY
1	36.63	38.72	6233.52	6234.85	389.60	390.52
2	15.41	14.88	1103.49	1101.82	98.55	102.03
3	14.12	13.62	926.57	927.79	84.50	89.72
4	10.48	10.32	510.20	510.22	49.14	58.70
5	15.41	14.88	1103.67	1101.81	98.56	102.03
6	36.64	38.72	6234.22	6235.27	389.63	390.54

Table 8. Sensitivity Matrices For Case II

$\partial e/\partial X_j =$

-0.007279	-0.000481	-0.001601	-0.014812	-0.000574	-0.006990
-0.002249	-0.009084	-0.013734	-0.017276	-0.008980	-0.002455
-0.000135	0.002811	0.009444	-0.000078	0.002811	-0.000135
0.000390	0.000427	-0.000309	0.007546	0.000427	0.000390
0.002455	0.008980	0.013734	0.017276	0.009084	0.002249
0.006990	0.000574	0.001601	0.014812	0.000481	0.007279

Under Traynor's Set (First Iteration)

$\partial e/\partial X_j =$

-0.000637	-0.000022	-0.000394	-0.002374	-0.000082	-0.000561
0.000466	0.002194	0.003783	0.002839	0.002036	0.000533
-0.000009	0.000721	0.002903	0.000004	0.000721	-0.000009
0.000025	0.000035	-0.000047	0.007173	0.000035	0.000025
-0.000533	-0.002036	-0.003783	-0.002839	-0.002194	-0.000466
0.000561	0.000082	0.000394	0.002374	0.000022	0.000637

Under Traynor's Set (Final Iteration)

$\partial e/\partial X_j =$

-0.006566	-0.000427	-0.001418	-0.013134	-0.000509	-0.006306
-0.002027	-0.008053	-0.012180	-0.015289	-0.007961	-0.002212
-0.000122	0.002490	0.008369	-0.000067	0.002490	-0.000122
0.000352	0.000379	-0.000272	0.006683	0.000379	0.000352
0.002212	0.007961	0.012180	0.015289	0.008053	0.002027
0.006306	0.000509	0.001418	0.013134	0.000427	0.006566

Under Ang's Set (First Iteration)

$\partial e/\partial X_j =$

-0.000701	-0.000020	-0.000350	-0.002063	-0.000072	-0.000619
0.000514	0.001954	0.003362	0.002464	0.001815	0.000587
-0.000010	0.000644	0.002576	0.000006	0.000644	-0.000010
0.000028	0.000031	-0.000041	0.006195	0.000031	0.000028
-0.000587	-0.001815	-0.003362	-0.002464	-0.001954	-0.000514
0.000619	0.000073	0.000350	0.002063	0.000020	-0.000701

Under Ang's Set (Final Iteration)

Table 9. Final Deflection Values For Case II

MEMBER.	Initial Defl	Final Defl.
1	1.7011	0.5000
2	2.2854	-1.0000
3	-0.5746	-0.3000
4	-0.3740	-0.3000
5	-2.2854	1.0000
6	-1.7011	-0.5000

Table 10. Final Design Values For Case IV

Member	A		I		S	
	ANG	TRAY	ANG	TRAY	ANG	TRAY
1	10.0	10.97	464.09	588.025	45.0	64.66
2	12.76	12.06	756.23	724.05	70.45	74.87
3	11.3	9.82	593.09	450.93	56.47	54.09
4	12.76	12.06	756.15	723.98	70.45	74.86
5	10.0	10.97	464.28	588.22	45.02	64.67

Table 11. Deflection Values-Case IV(Small Initial Sec. Prop.)

MEMBER.	Initial Defl.	Final Defl.
1	45.1471	0.6499
2	-25.8166	-0.3000
3	-44.6449	0.6299
4	-25.8166	-0.3000
5	-45.1474	-0.6499

5.0 CONCLUSION

The four cases represented in this investigation have been used to study the following :

5.1 CONVERGENCE CHARACTERISTICS

As shown in the examples, the convergence rate depends on the structure size for large initial values and gives almost the same numbers of iteration for small initial values. Some experience with allowable deflection and the structural behavior is required to get a fast convergence rate.

For most cases, the displacement sensitivity matrix gave afairly accurate estimate for the final matrix after the third iteration. The sensitivity matrix is not only used as a convergence indicator but also gives valuable information about the structural behavior.

The allowable deflection has the most important effect on the convergence rate and a background of experience is required; i.e., the allowable column deflection should be prescribed so that the higher floors allow more deflection than the lower floors.

5.2 COMPARISON BETWEEN TRAYNOR'S EQUATIONS AND ANG'S EQUATIONS

As shown in Table 3, the system tends to converge faster under Traynor's equations. However, this is only true for small structures (up to two story) because these sets of equations are apparently no longer valid for larger structures or even under heavy loads, because the size of the members may have to exceed those in the range defined for which the equations were developed. Analyzing this problem through this study, the main reason for nonconvergence condition under Traynor's equations for large structures is that the upper bound limit causes the convergence indicator to oscillate about one point when it reaches the limit. In other words, this limit does not permit the design variable increment, ΔX, to change freely as dictated by the behavior constraints. Therefore, when the size of the structure is large, Ang's equations are preferable because this set of equations does not have any limit and allows the design variable increment, ΔX, to change freely with each iteration.

This algorithm differs from others primarily in the way in which the design improvement ΔX is computed. However, using Newton's Method has one disadvantage; the method has some difficulties in adjusting itself in the first few iterations, which slows the convergence rate but only in the beginning of the iterative procedure.

6.0 EXTENSION

The next step in the complete structural design algorithm is to combine the stress constraints and displacement constraints into one so that the final design does not violate either. In the final design state, the stresses should be as close as possible to their allowable stress values while satisfying the displacement constraints.

The other factor in the complete design is to check the final structure for the optimum conditions (minimum weight). Modifications could be made to approach the minimum weight criterion if necessary [9]. This algorithm would combine stress constraints, displacement constraints, and minimum weight.

It would be useful to study the displacement sensitivity and stress sensitivity coefficients for a frame structure to obtain some insight, and to compare their physical and computational characteristics.

REFERENCES

1. Arora, J. S., Haug, E. J., "Methods of Design Sensitivity Analysis in Structural Optimization", AIAA Journal, Vol. 17, September, 1979, pp. 970-974.

2. Arora , J. S., Haug, E. J., Rim, K., "Optimal Design of Plane Frames ",Journal of the Structural Division, ASCE, Vol. 101, No. ST10, October, 1975, pp. 2063-1077.

3. Brown, D. M., Ang, A. H-S., "Structural Optimization by Nonlinear Programming",Journal of the Structural Division, ASCE, Vol. 92, No. ST6, December, 1966, P. 319.

4. Fling, R. S., "Allowable Deflections", American Concrete Institute Journal, Vol. 67, No.4, June, 1968, PP. 433-442.

5. Gaylord, Jr., E. H., Gaylord, C. N., Structural Engineering Handbook, McGraw-Hill, Inc., New York, New York, 1979.

6. Haug, E. J. , Pan, K. C., Streeter, T. D., "A Computational Method For Optimal Structural Design .I. Piecewise Uniform Structures", International Journal For Numerical Methods in Engineering, Vol. 5, 1972, pp. 171-184.

7. Holzer, S. M., Computer Analysis of Structures: Matrix Structural Analysis, Structured Programming, Elsevier Science, INC., New York, New York, 1985.

8. Kirsch, U., Optimum Structural Design, McGraw-Hill, Inc., New York, New York, 1981.

9. Pickeit, R. M., Rubinstein, M. F., Nelson, R. B., "Automated Structural Synthesis Using a Reduced Number of Design Coordinates", AIAA Journal,Vol. 11, No. 4, April, 1973, pp. 489-494.

10. Raze, J. D., "The Development of a Minimum Weight Design Algorithm Using Fully Stressed Design Iteration", Thesis presented to the Department of Civil Engineering at Virginia Polytechnic Institute and State University in Blacksburg, Virginia, in 1984, in partial fulfillment for the requirement for the degree of Master of Science.

11. Romstad, K. M., Wang, C. K., "Optimum Design of Framed Structures", Journal of the Structural Division, ASCE, Vol. 94,No. ST12, December, 1968, PP. 2817-2845.

12. Traynor, K. H., "A Study of Features of the Fully Stressed Design Algorithm", Thesis presented to the Department of Civil Engineering at Virginia Polytechnic Institute and State University in Blacksburg, Virginia, in 1982, in partial fulfillment for the requirement for the degree ofMaster ofScience.

13. Class Notes, CE 4001, Matrix Structural Analysis, Fall Quarter, 1984.

14. Class Notes, CE 4002, Matrix Structural Analysis, Winter Quarter, 1984.

15. Class Notes, CE 5640, Computer-Aided Structural Design, Spring Quarter, 1985.

APPENDIX A.

A.1 Matrix Displacement Method

In the matrix displacement method [7] the unknowns are the joint displacements. Each joint is considered to have three degrees of freedom. The relation between the element-end forces and element-end displacements is represented by stiffness matrix.

A typical element is modeled as a beam-column. Equation A.1 represents the form of the model and equation A.2 is the matrix form of equation A.1. Figure A.1 shows the beam-column element model.

$$\underset{(6,1)}{f^i} = \underset{(6,6)(6,1)}{k^i} \cdot d^i \qquad (A.1)$$

where

f^i = element-end forces

d^i = element-end displacements

k^i = local element stiffness matrix

$i = i^{th}$ element

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \alpha \begin{bmatrix} \beta & 0 & 0 & -\beta & 0 & 0 \\ 0 & 12 & 6L & 0 & -12 & 6L \\ 0 & 6L & 4L^2 & 0 & -6L & 2L^2 \\ -\beta & 0 & 0 & \beta & 0 & 0 \\ 0 & -12 & -6L & 0 & 12 & -6L \\ 0 & 6L & 2L^2 & 0 & -6L & 4L^2 \end{bmatrix} \cdot \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} \qquad (A.2)$$

where

$$\alpha = E\ I/L^3$$

$$\beta = A\ L^3/I$$

A.2 Global Frame Element

The local element model (equation A.1) is transformed into a global element through the coordinate transformation matrix A.3 and A.4.

$$\underset{(6,1)}{F^i} = \underset{(6,6)}{K^i} \cdot \underset{(6,1)}{D^i} \qquad (A.3)$$

$$\{d\} = \{\Lambda\} \cdot \{D\} \qquad (A.4)$$

$$\{F\} = [\Lambda^T] \cdot \{f\} \qquad (A.5)$$

The expanding equation for equation A.3 is equation A.6.

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} = \alpha \begin{bmatrix} g_1 & g_2 & g_4 & -g_1 & -g_2 & g_4 \\ g_2 & g_3 & g_5 & -g_2 & -g_3 & g_5 \\ g_4 & g_5 & g_6 & -g_4 & -g_5 & g_7 \\ -g_1 & -g_2 & -g_4 & g_1 & g_2 & -g_4 \\ -g_2 & -g_3 & -g_5 & g_2 & g_3 & -g_5 \\ g_4 & g_5 & g_7 & -g_4 & -g_5 & g_6 \end{bmatrix} \cdot \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \\ D_6 \end{bmatrix} \qquad (A.6)$$

A.3 System Model

The system stiffness matrix is formed by a contribution from each global element stiffness matrix. The procedure is as follows :

1. Each element obtains its global element stiffness matrix, k^i as in equation A.4 imposing conditions of compatibility, which can be expressed by the member code matrix M, which describes the type and ordering number of degrees of freedom.
2. k^i for each member is transformed into an n×n generalized global stiffness matrix, where n is the number of degrees of freedom for the system. Thus, the generalized global stiffness matrix coefficient is only for the entries in k^i which have a corresponding degree of freedom. The sum of all generalized global stiffness matrices is the final system stiffness which is represented by equation A.7.

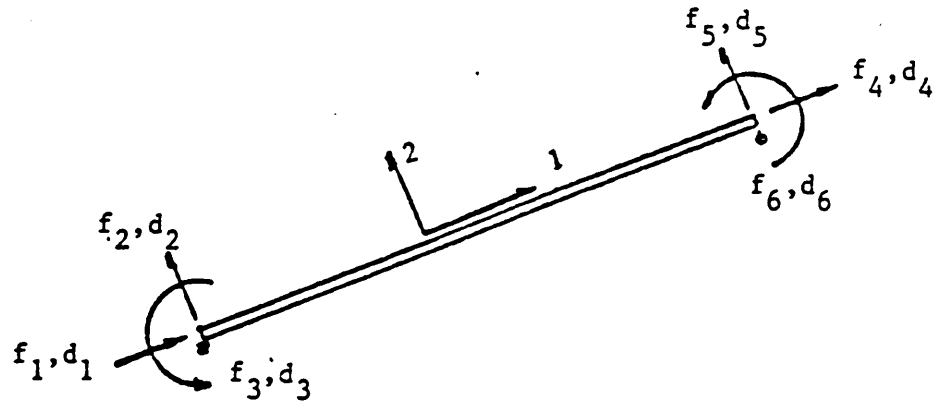
$$\underset{(n,n)}{K} = \Sigma_i^{ne} \underset{(n,n)}{k^{(i)}} \qquad (A.7)$$

where

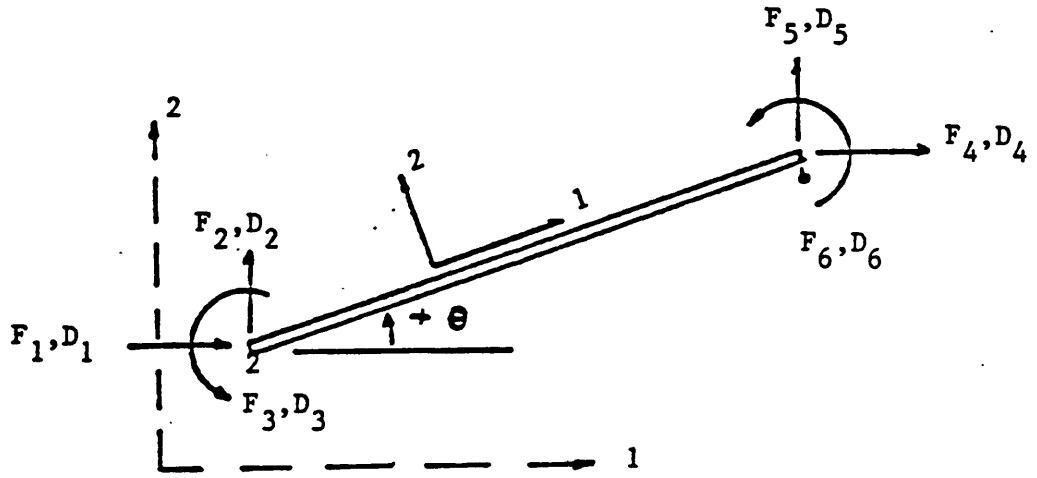
n = number of degrees of freedom

Using the same format as in equation A.1, the system model is

$$\underset{(n,n)}{K} \cdot \underset{(n,1)}{q} = \underset{(n,1)}{Q} \qquad (A.8)$$



a. Local Beam-Column Element Model



b. Global Element Model

Figure 11. Beam-Column Model

where

K = system stiffness matrix

q = generalized displacement vector

Q = equivalent joint load vector

The system model represents a set of simultaneous equations; solving for the unknown yields the generalized displacement vector q.

Table 12. Stiffness Coefficients Derivatives W.R.T. S

$ga_1 = Ec_1^2/L$

$ga_2 = Ec_1c_2/L$

$ga_3 = Ec_2^2/L$

$ga_4 = 0$

$ga_5 = 0$

$ga_6 = 0$

$ga_7 = 0$

$gi_1 = 12\ Ec_2^2/L^3$

$gi_2 = -12\ Ec_1c_2/L^3$

$gi_3 = 12\ Ec_1^2/L^3$

$gi_4 = -6Ec_2/L^2$

$gi_5 = 6Ec_1/L^2$

$gi_6 = 4E/L$

$gi_7 = 2E/L$

where: $g_1 = \alpha(\beta c_1^2 + 12c_2^2)$

$g_2 = \alpha c_1c_2(\beta - 12)$

$g_3 = \alpha(\beta c_2^2 + 12c_1^2)$

$g_4 = -\alpha 6Lc_2$

$g_5 = \alpha 6Lc_1$

$g_6 = \alpha 4L^2$

$g_7 = \alpha 2L^2$

and

$\alpha = EI/L^3$

$\beta = AL^2/I$

APPENDIX B.

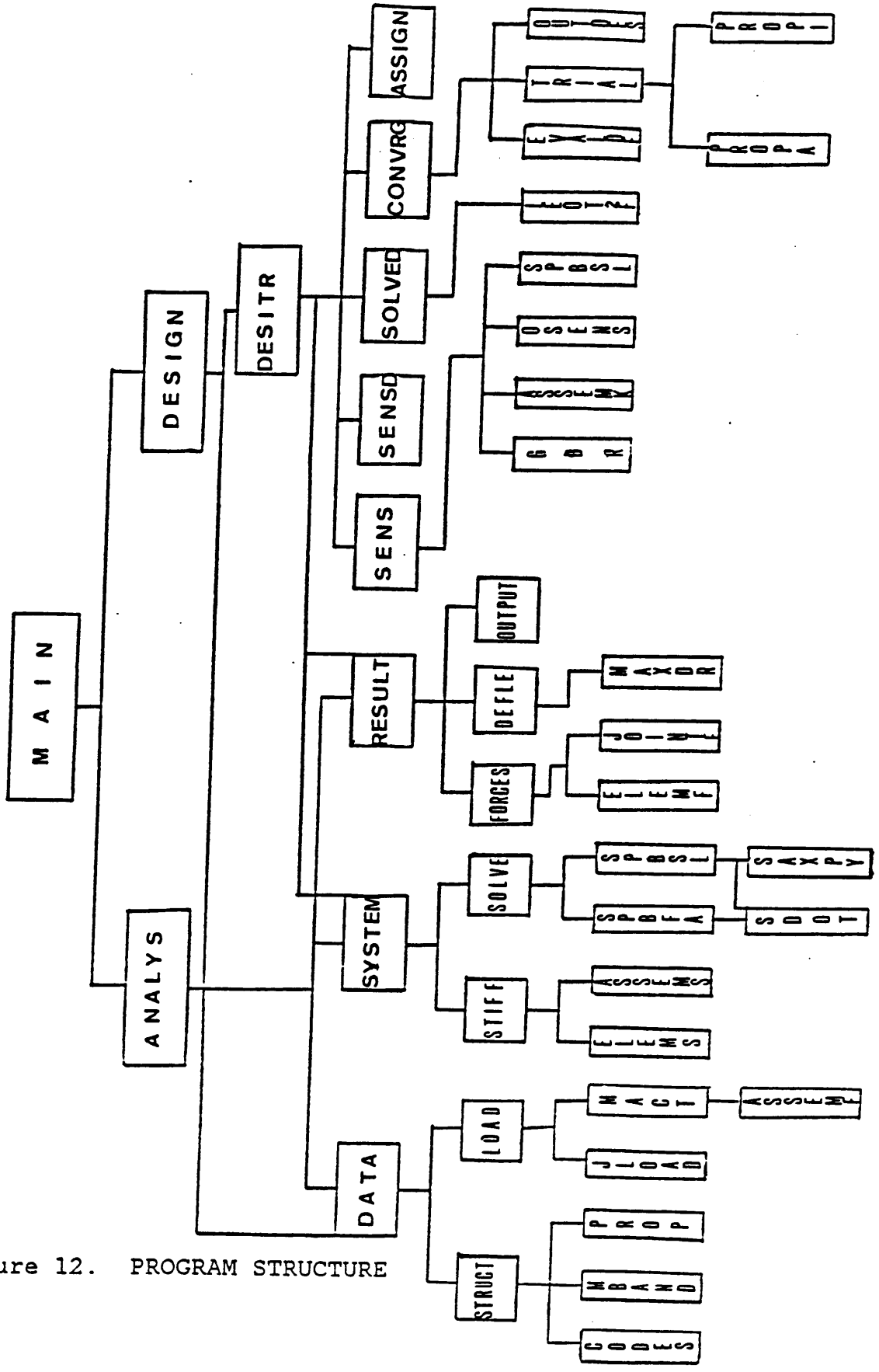
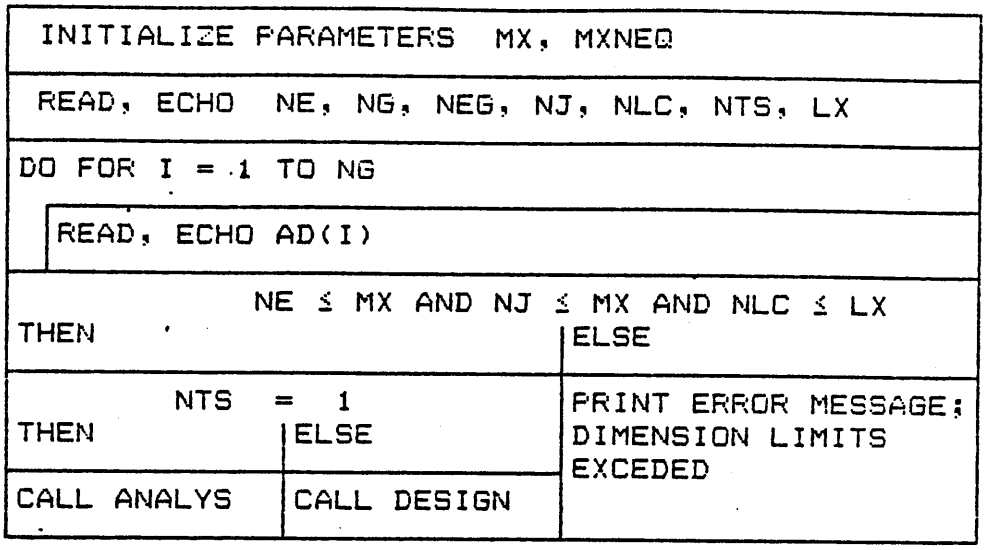
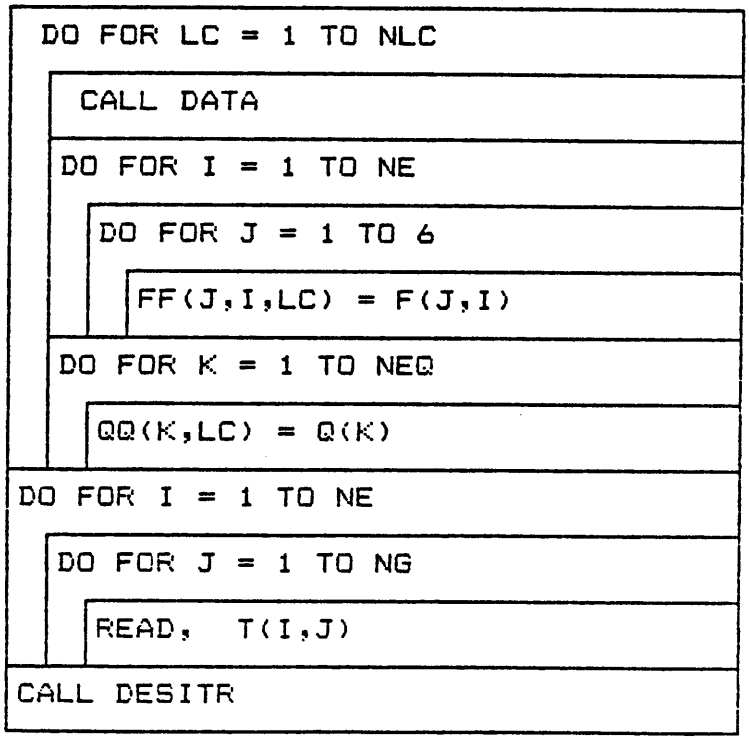


Figure 12. PROGRAM STRUCTURE



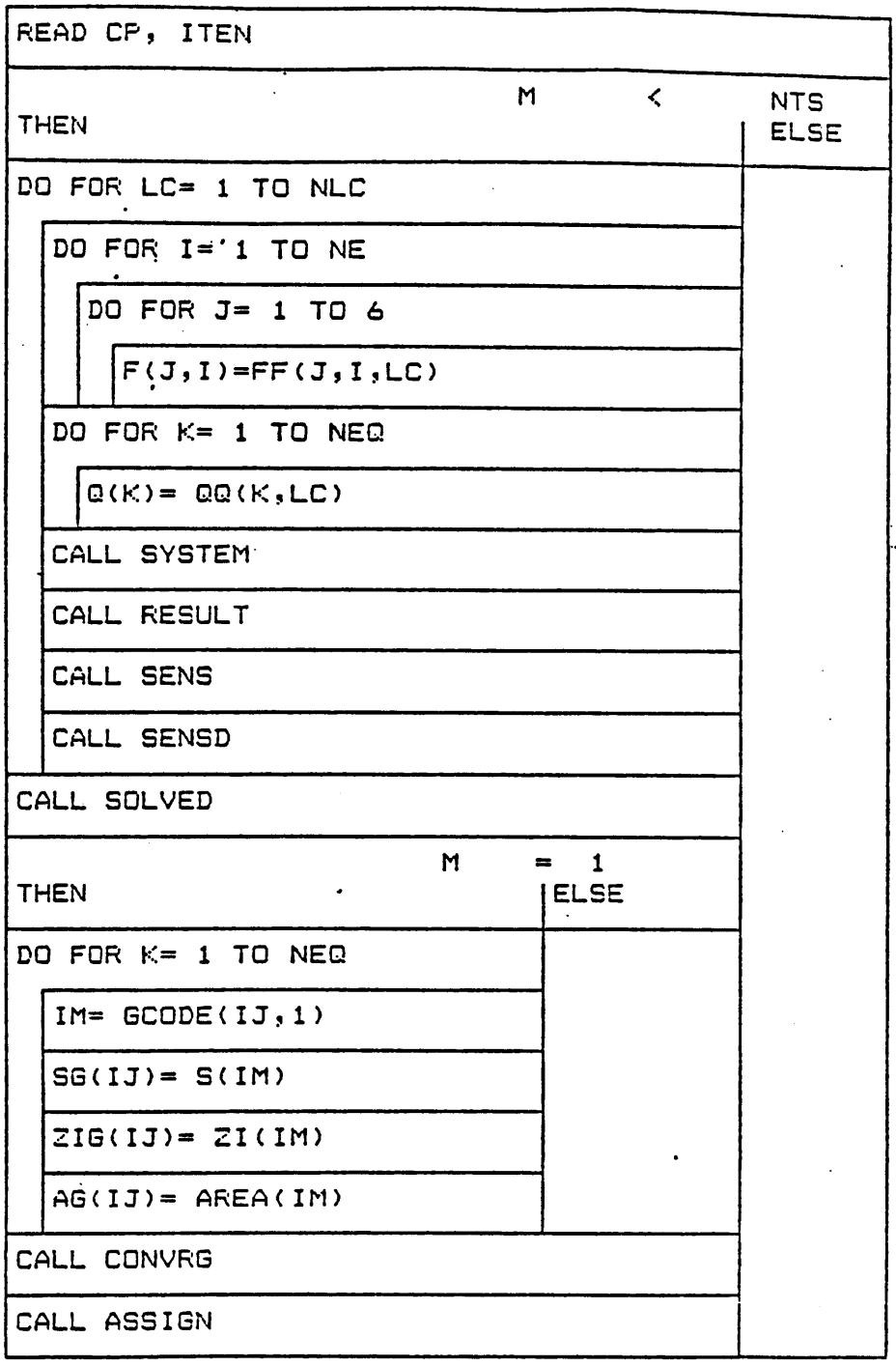
MAIN PROGRAM



SUBROUTINE DESIGN

INPUT ARGUMENTS: NE, NG, NEG, NJ, NLC, NTS, AD
OUTPUT ARGUMENTS: FF, QQ, T, F, Q, AREA, ZI, S, EMOD, CTE, ELENG, C1, C2, MCODE, JCODE, MINC, NA, NEQ, MBD, GCODE

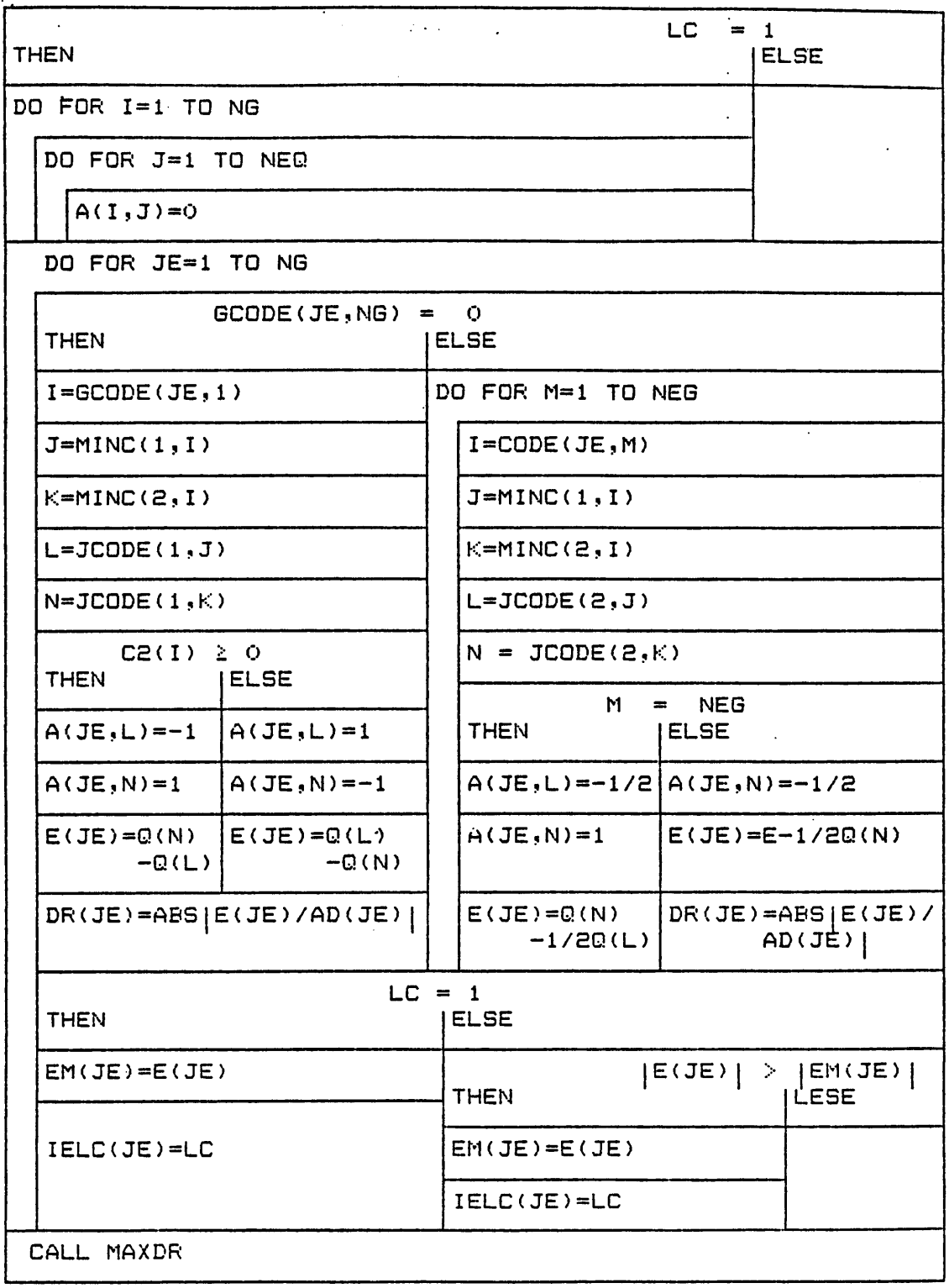
Figure 13. MAIN PROGRAM & SUBROUTINE DESIGN



INPUT ARGUMENTS: NE, NG, NEG, NJ, NLC, NTS, AD, FF, QQ, MBD, T, F, P, Q, AREA, AI, S, EMOD, CTE, ELENG, C1, C2, MCODE, JCODE, MINC, NA, NEQ, GCODE

OUTPUT ARGUMENTS: SS, Q, MXNEQ, G, EM, A, E, DR, RM, IELC, DKGA, DKGI, DAS, DIS, DKS, QS, QD, QDG, DE, DEM, DELM, DLTS, AG, ZIG, SG, ILC

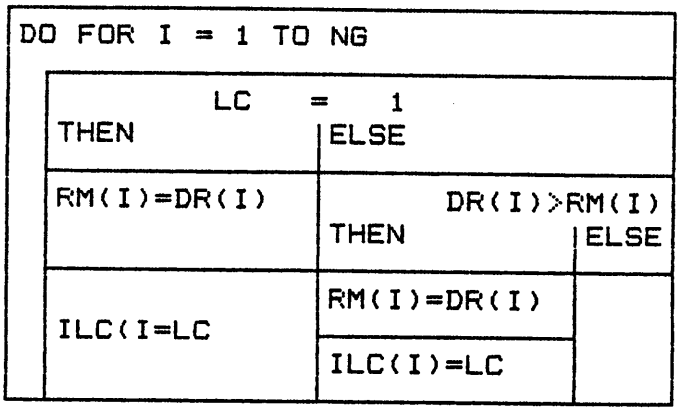
Figure 14. SUBROUTINE DESITR



SUBROUTINE DEFLE

Figure 15. SUBROUTINE DEFLE

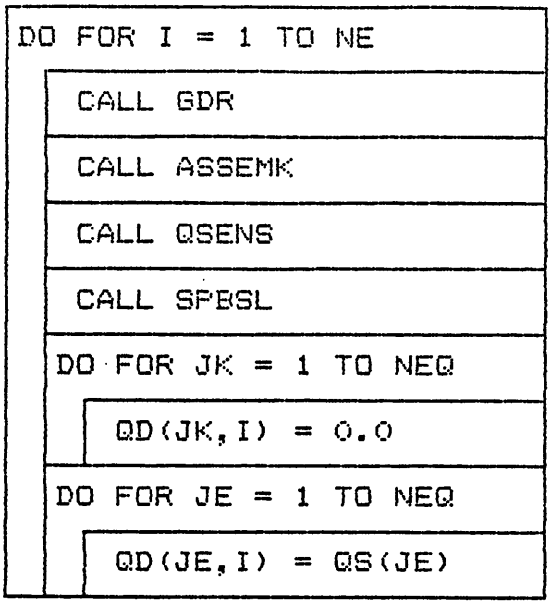
INPUT ARGUMENTS: Q, JCODE, GCODE, MINC, NG, NEG, LC, C2, NEQ, AD
OUTPUT ARGUMENTS: E, EM, DR, RM, ILC, IELC, A



SUBROUTINE MAXDR

INPUT ARGUMENTS: LC,DR, NG
OUTPUT ARGUMENTS: ILC, RM

Figure 16. SUBROUTINE MAXDR



SUBROUTINE SENS

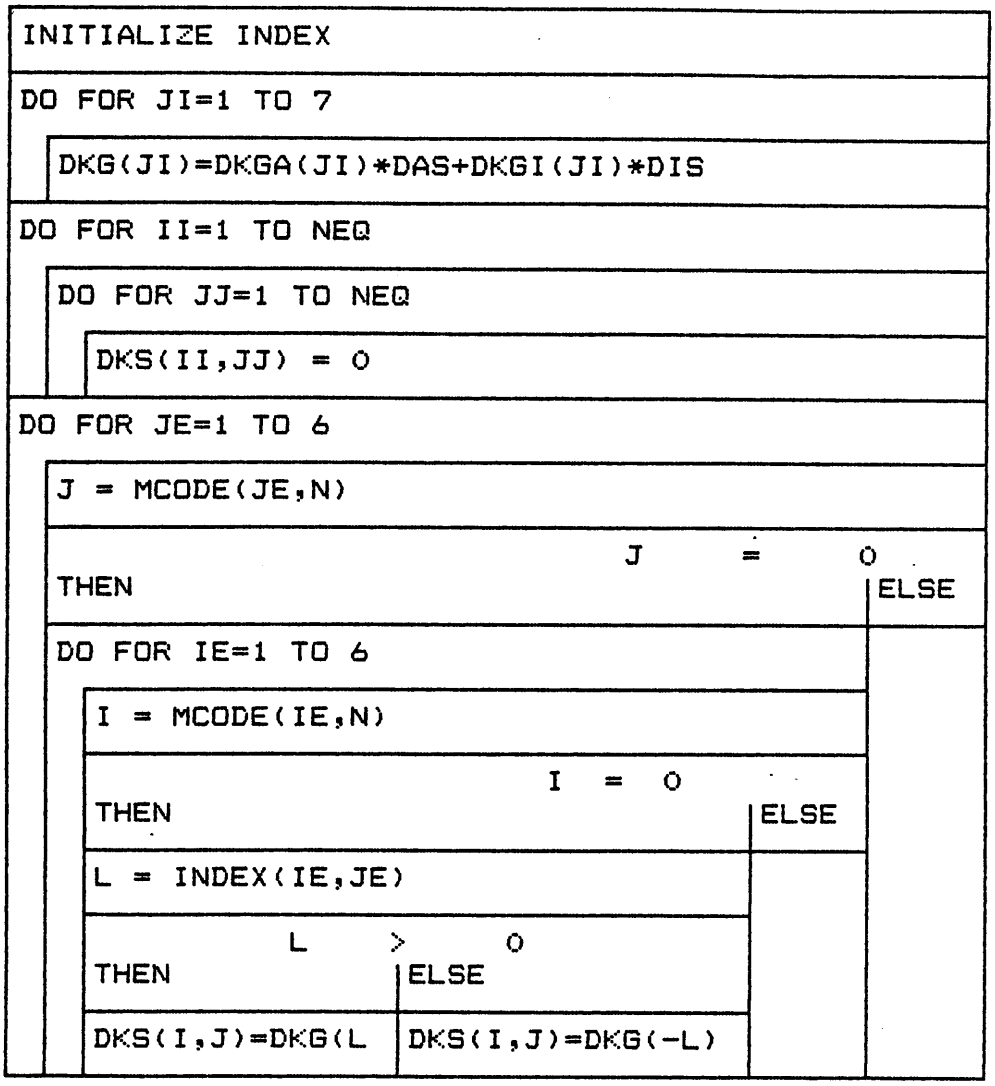
INPUT ARGUMENTS: C1, C2, EMOD, ELENG, NE, S, Q, NEQ, MXNEQ, SS, MCODE, JCODE, MINC, LC, MBD, ITEN, NG
OUTPUT ARGUMENTS: DKGA, DKGI, DAS, DIS, DKS, QS, QD

DIFFERENTIAT K W.R.T. A
DIFFERENTIAT K W.R.T. I
DIFFERENTIAT TRAYNOR'S EQU. W.R.T. S
DIFFERENTIAT ANG'S EQU. W.R.T. S

SUBROUTINE GDR

INPUT ARGUMENTS: C1, C2, EMOD, ELENGE, I, S, ITEN
OUTPUT ARGUMENTS: DKGA, DKGI, DAS, DIS

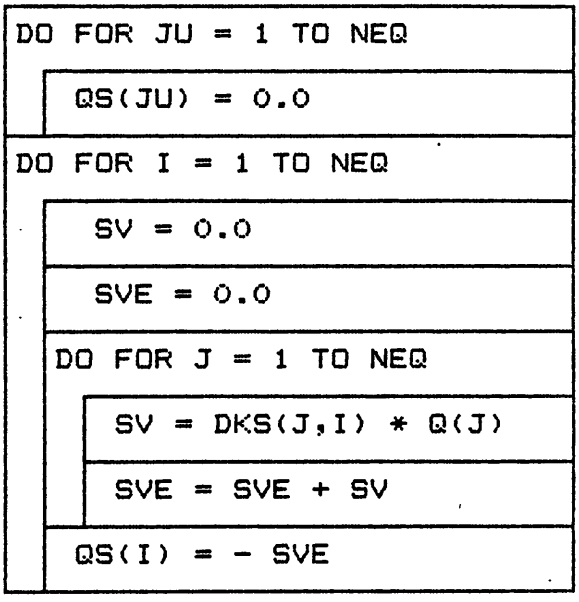
Figure 17. SUBROUTINES SENS & GDR



SUBROUTINE ASSEMK

INPUT ARGUMENTS: N, NEQ, MCODE, DKGA, DKGI, DAS, MXNEQ, DIS
OUTPUT ARGUMENT: DKS

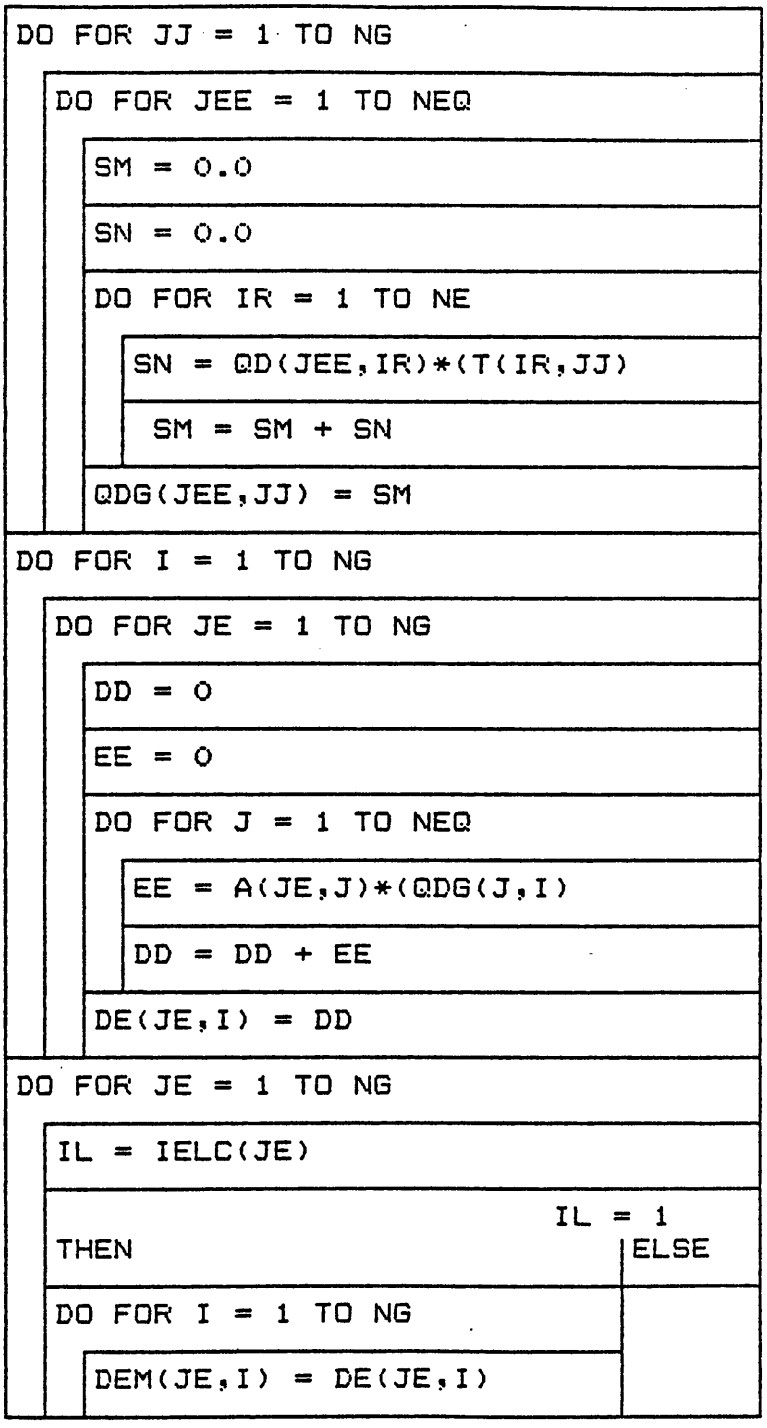
Figure 18. SUBROUTINE ASSEMK



SUBROUTINE QSENS

INPUT ARGUMENTS: Q, DKS, NEQ, LC, MXNEQ, NEQ, I
OUTPUT ARGUMENTS: QS

Figure 19. SUBROUTINE QSENS



INPUT ARGUMENTS: LC, NE, NG, NEG, NEQ, T, A, MXNEQ,
EM, GCODE, IELC, M
OUTPUT ARGUMENTS: DEM, QDG, DE

Figure 20. SUBROUTINE SENSD

DO FOR JE = 1 TO NG	
EM(JE) < 0	
THEN	ELSE
DLEM(JE)=-AD(JE)-EM(JE	DLEM(JE)=AD(JE)-EM(JE)
IDGT = 4	
MM = 1	
CALL LEQT2F	
DO FOR J = 1 TO NG	
DLTS(J) = DLEM(J)	

SUBROUTINE SOLVED

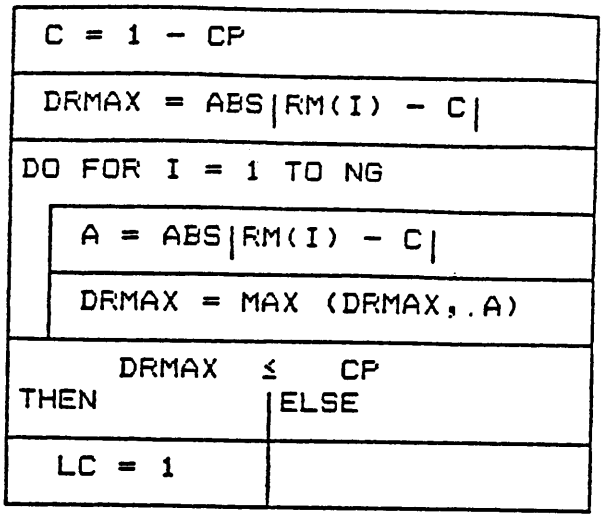
INPUT ARGUMENTS: DEM, EM, NG, AD
OUTPUT ARGUMENTS: DLEM, DLTS

IC = 0	
CALL EVALDE	
IC = 0	
THEN	ELSE
M = M + 1.	M = NTS + 1
CALL TRIAL	CALL OUTDES

SUBROUTINE CONVRG

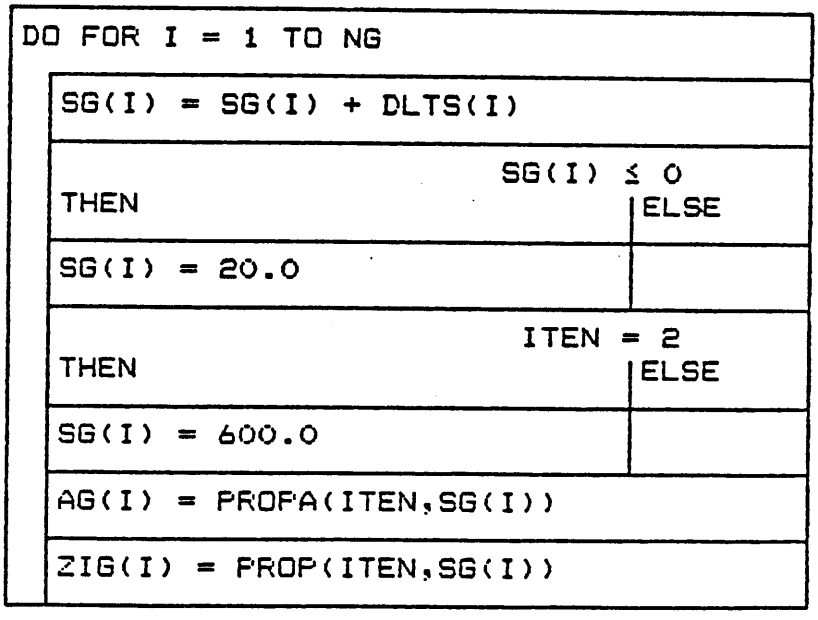
INPUT ARGUMENTS: RM, CP, NG, SG, M, ILC, DLTS, ITEN
OUTPUT ARGUMENTS: NTS, AG, ZIG, SG, M, IC

Figure 21. SUBROUTINES SOLVED & CONVRG



SUBROUTINE EVALDE

INPUT ARGUMENTS: RM, CP, IC, NG SUBROUTINE EVALDA
OUTPUT ARGUMENT: IC



SUBROUTINE TRIAL

INPUT ARGUMENTS: SG, NG, DLTS, ITEN
OUTPUT ARGUMENTS: SG, AG, ZIG

Figure 22. SUBROUTINES EVALDE & TRIAL

N = 1	
THEN	ELSE
COMPUTE NEW AREAS (TRAYNOR'S EQUATIONS)	COMPUTE NEW AREAS (ANG'S EQUATIONS)
PROPA = TAREA	

SUBROUTINE PROPA

INPUT ARGUMENTS: SG, ITEN
OUTPUT ARGUMENT: AG

N = 1	
THEN	ELSE
COMPUTE NEW MOMENTS OF INERTIA (TRAYNOR'S EQUATIONS)	COMPUTE NEW MOMENTS OF INERTIA (ANG'S EQUATIONS)
PROPI = TZI	

SUBROUTINE PROPI

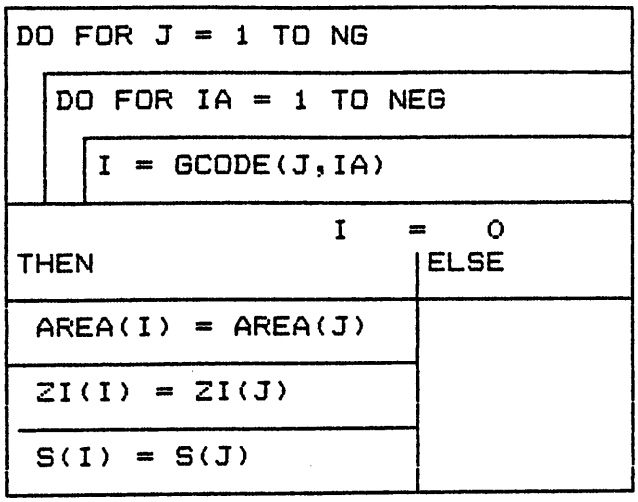
INPUT ARGUMENTS: SG, ITEN
OUTPUT ARGUMENT: ZIG

DO FOR I = 1 TO NG
PRINT, FINAL DESIGN VALUES (AG,ZIG,SG,RM,ILC,M,CP)

SUBROUTINE OUTDES

INPUT ARGUMENTS: AG, ZIG, SG, RM, ILC
OUTPUT ARGUMENT: NONE

Figure 23. SUBROUTINES PROPA, PROPI, & OUTDES



INPUT ARGUMENTS: SG, AG, ZIG, NG, NEG, GCODE
OUTPUT ARGUMENTS: S, AREA, ZI

Figure 24. SUBROUTINE ASSIGN

PARAMETER LISE

MXNE	MAXIMUM NUMBER OF ELEMENTS
MXNEQ	MAXIMUM NUMBER OF EQUATIONS (DEGREES OF FREEDOM)
LX	MAXIMUM NUMBER OF LOAD CONDITIONS

INPUT LIST

AD(I)	ALLOWABLE DEFLECTION FOR MEMBER I
ACT	MEMBER ACTION MAGNITUDE
AREA	AREA OF ELEMENT i
CP	CONVERGENCE TOLERANCE
CTE	COEFFICIENT OF THERMAL EXPANSION
DIST1	DISTANCE TO THE LOAD FROM THE A-END
DIST2	DISTANCE TO THE LOAD FROM THE B-END
EMOD	MODULUS OF ELASTICITY OF ELEMENT i
FORCE	MAGNITUDE OF JOINT LOAD
JDIR	DIRECTION OF JOINT LOAD
JNUM	JOINT NUMBER OF JOINT LOAD APPLICATION
ITEN	DEPENDENT SECTION PROPERTIES INDICATOR
GCODE(NG,NEG)	MEMBER CODE MATRIX
MINC	MEMBER INCIDENCES
NE	NUMBER OF ELEMENTS
NG	NUMBER OF GROUPS
NEG	NUMBER OF ELEMENTS IN THE GROUP
NJ	NUMBER OF JOINTS
NLC	NUMBER OF LOAD CONDITIONS
NTS	NUMBER OF TRIAL STRUCTURES
MAT	MEMBER ACTION TYPE
MN	MEMBER NUMBER
S	SECTION MODULUS OF ELEMENT i
T	DESIGN VARIABLE TRANSFORMATION MATRIX
ZI	MOMENT OF INERTIA OF ELEMENT i
X(1,J),X(2,J)	GLOBAL 1, 2-COORDINATES OF JOINT J

MATRIX DISPLACEMENT METHOD VARIABLE LIST

C1(I),C2(I)	DIRECTION COSINES OF ELEMENT i
D(6)	GLOBAL ELEMENT DISPLACEMENT VECTOR
ELENGE(I)	LENGTH OF ELEMENT i
INDEX(6,6)	INDEX MATRIX
F(6,MX)	LOCAL ELEMENT FORCE MATRIX
FF(6,NE,LC)	LOCAL ELEMENT FORCE VECTORS FOR ALL LOAD CONDITIONS
G(7)	GLOBAL ELEMENT STIFFNESS COEFFICIENTS
JCODE(3,MX)	JOINT CODE MATRIX
LC	LOAD CONDITION
MBD	HALF BANDWIDTH
MCODE(6,MX)	MEMBER CODE MATRIX

```
NA(I)             NUMBER OF ACTIONS ON ELEMENT i
NEQ               NUMBER OF EQUATIONS (DEGREES OF FREEDOM)
P(3,MX)           JOINT FORCE MATRIX
Q(MXNEQ)          JOINT LOAD, DISPLACEMENT, VECTOR
QQ(NEQ,LC)        JOINT DISPLACEMENTS FOR ALL LOAD CONDITIONS
SS(MXNEQ,MXNEQ)   SYSTEM STIFFNESS BAND MATRIX
**********************************************************
            FULLY DISPLACED DESIGN VARIABLE LIST
**********************************************************
A(NG,NX)          MEMBER DISPLACEMENT TRANSFORMATION MATRIX
E(I)              DEFLECTION OF MEMBER I
EM(I)             CONTROLLING (MAX.) DEFLECTION OF MEMBER I
DR(I)             DEFLECTION RATIO FOR MEMBER I
RM(I)             CORRESPONDING DEFLECTION RATIO TO EM(I)
IELC(I)           INDICATOR WHICH IDENTIFIES THE LOADING
                  CONDITION AT WHICH EM(I) OCCURS
ILC(I)            INDICATOR WHICH IDENTIFIES THE LOADING
                  CONDITION AT WHICH RM(I) OCCURS
**********************************************************
          DISPLACEMENT SENSITIVITY VARIABLE LIST
**********************************************************
x(i)              DESIGN VARIABLE OF ELEMENT i
DKGA(1-7)         DERIVATIVES OF GLOBAL STIFFNESS COEFFICIENTS
                  W.R.T. AREA
DKGI(1-7)         DERIVATIVES OF GLOBAL STIFFNESS COEFFICIENTS
                  W.R.T. MOMENT OF INERTIA
DAS               DERIVATIVES OF DEPENDENT SECTION PROPERTIES
                  EQUATIONS FOR AREA W.R.T. SECTION MODULUS, S
DIS               DERIVATIVES OF DEPENDENT SECTION PROPERTIES
                  EQUATIONS FOR MOMENT OF INAREA W.R.T.
                  SECTION MODULUS, S
DKG(1-7)          COMBINED DERIVATIVE STIFFNESS COEFFICIENTS
DKS(NEQ,NEQ)      DERIVATIVE OF STIFFNESS MATRIX W.R.T.
                  SECTION MODULUS, S
QS(NEQ)           DERIVATIVE LOAD VECTOR FOR ELEMENT i W.R.T.
                  DESIGN VARIABLE, S
QD(NEQ)           DERIVATIVE SOLUTION (UNKNOWN) VECTOR FOR
                  ELEMENT i W.R.T. DESIGN VARIABLE, S
X(I)              DESIGN VARIABLE OF MEMBER I
QDG(NEQ)          DERIVATIVE SOLUTION (UNKNOWN) VECTOR FOR
                  MEMBER I W.R.T. DESIGN VARIABLE, SG
DE(NG,NG)         SENSITIVITY MATRIX (DERIVATIV OF E W.R.T
                  SECTION MODULUS, SG)
DEM(NG,NG)        SENSITIVITY MATRIX CORRESPONDING TO THE
                  CONTROLLING LOAD CONDITIONS
AG(I)             AREA OF MEBER I
SG(I)             SECTION MODULUS OF MEMBER I
ZIG(I)            MOMENT OF INERTIA OF MEMBER I
DLEM(NG)          VECTOR OF DESIGN DISPLACEMENT CHANGES
DLTS(NG)          VECTOR OF DESIGN VARIABLE CHANGE
**********************************************************
```

```
C*************************************************************
C*                          PROGRAM MENU                      *
C*************************************************************
C     LIST-DIRECTED INPUT
C     INPUT UNITS: KIP, INCH, RADIAN, FAHRENHEIT
C
C     1. ENTER DATE IN THE FORM '12/19/85'(IN MAIN)
C           DATE
C
C     2. ENTER NUMBER OF ELEMENTS,NUMBER OF GROUPS, NUMBER
C        OF ELEMENTS IN THE GROUP, NUMBER OF JOINTS,
C        NUMBER OF LOAD CONDITIONS AND NUMBER OF TRAIL
C        STRUCTURE(IN MAIN)
C           NE, NG, NEG, NJ,  NLC, NTS
C
C     3. ENTER THE ALLOWABLE DEFLECTION FOR EACH MEMBER
C        (IN MAIN)
C        AD(I)
C
C     4. ENTER MEMBER INCIDENCES(IN STRUCT)
C           MINC(1,I), MINC(2,I)  I=1,NE
C
C     5. ENTER FOR EACH JOINT CONSTRAINT(IN STRUCT)
C           JNUM, JDIR
C           AFTER LAST JOINT CONSTRAINT ENTER
C           0, 0
C     6. ENTER GCODE ( IN CODES )
C        GCODE( NG,NG )
C
C     7. ENTER JOINT COORDINATES(IN PROP)
C           X(1,J), X(2,J)  J=1,NJ
C
C     8. ENTER MEMBER PROPERTIES(IN PROP)
C        AREA(I), ZI(I), EMOD(I), CTE(I), S(I),  I=1,N
C
C     9. DO FOR EACH LOAD CONDITION
C           IF THERE ARE JOINT LOADS ENTER(IN JLOAD)
C              JNUM, JDIR, FORCE
C              AFTER LAST JOINT LOAD ENTER
C              0, 0, 0.
C           ELSE ENTER
C              0, 0, 0.
C           END IF
C
C           IF THERE ARE MEMBER ACTIONS ENTER(IN MACT)
C              MN, MAT, ACT, DIST1, DIST2
C              AFTER LAST MEMBER ACTION ENTER
C              0, 0, 0, 0, 0.
C           ELSE ENTER
C              0, 0, 0, 0, 0.
C            END IF
```

```
C           END DO
C       10. ENTER T MATRIX ( IN DESIGN )
C           T( NE,NG )
C
C       11. ENTER DESIGN CONVERGENCE PARAMETER AND
C           DEPENDENT SECTION PROPERTY INDICATORIN (IN DESITR)
C           CP, IETN
```

```
C=============================================================
C=              M A I N     P R O G R A M                  =
C=============================================================
C
C FUNCTION- TO SERVE AS THE HIGHEST LEVEL MANAGMENT ROUTINE
C           FOR BOTH ANALYSIS AND DESIGN:
C           A)ESTABLISH ALL ARRAY DIMENTIONS.
C           B)MAKE DECISION FOR ANALYSIS OR DESIGN
C           C)INPUT CONTROLL PARAMETERS;
C           INITIALIZE PARMETERS MX, MXNEQ,LX; READ AND ECHO
C           NE,NJ,NLC,NT IF NE AND NJ ARE LESS THAN OR EQUAL
C           TO MX AND NLC LESS THAN 0 EQUAL TO LX THEN IF
C           NTS 1 THEN CALL FOR ANALYSIS ELSE CALL DESIGN
C           AND STOP ELSE PRINT ERROR MASSEAGE AND STOP.
C
      CHARACTER *(*) TITLE,UNITS,DATE*8
      PARAMETER (MX=40,MXNEQ=3*(MX-1),LX=4,
     $           TITLE='PLANE FRAME ANALYSIS',
     $           UNITS='UNITS:KIP,INCH,RADIAN, FAHRENEHEIT')
      DIMENSION F(6,MX),P(3,MX),SS(MXNEQ,MXNEQ),Q(MXNEQ),
     $          AREA(MX),ZI(MX),EMOD(MX),CTE(MX),ELENG(MX),
     $          C1(MX),C2(MX),S(40),MCODE(6,MX),JCODE(3,MX),
     $          MINC(2,MX),NA(MX),FF(6,40,40),AD(40),G(7),
     $          DKGA(7),DKGI(7),DKS(MXNEQ,MXNEQ),QS(MXNEQ),
     $          ILC(40),QD(MXNEQ,MX),QDG(MXNEQ,40),EM(40),
     $          ZIG(40),T(MX,40),DLTS(40),DE(40,40),SG(40),
     $          DEM(40,40),RM(40),DLEM(40),GCODE(40,10),
     $          IELC(40),E(40),DR(40),AG(40),A(40,MXNEQ),
     $          QQ(60,40)
C
      READ(5,*) DATE
      PRINT 10, TITLE, 'DATE; ', DATE, UNITS
   10 FORMAT('1', T10, 68('*')/ T10, '*', T34, A, T77,
     $       '*'/ T10,68('*')// T64, 2(A)// T10, A/ )
      READ(5,*) NE,NG,NEG,NJ,NLC,NTS
      DO 20  I=1,NG
         READ(5,*) AD(I)
   20 CONTINUE
      PRINT 30,'NUMBER OF ELEMENTS', NE,'NUMBER OF GROUPS',
     $     NG,'MAX. NUMBER OF ELEMENTS IN A GROUP ',NEG,
     $    'NUMBER OF JOINTS',NJ,'NUMBER OF LOAD CONDITIONS',
     $     NLC,'TOTAL NUMBER OF TRIAL STRUCTURES',NTS
   30 FORMAT(1X,6(T10, A,T46, I4/ ))
      IF(NE .LE. MX .AND. NJ .LE. MX .AND. NLC .LE. LX) THEN
        IF(NTS .EQ. 1) THEN
          CALL ANALYS(F,P,Q,SS,GCODE,NG,NEG,AREA,ZI,S,EMOD,
     $                CTE,ELENG,E,DR,C1,C2,MCODE,JCODE,MINC,
     $                NA,ILC,NE,NJ,NEQ,MBD,NLC,MXNEQ,MX,EM,
     $                IELC,RM,A,G,AD)
        ELSE
          CALL DESIGN(FF,QQ,F,P,Q,SS,AREA,ZI,S,EMOD,DKGA,
```

```
     $                  DKGI,DAS,DIS,CTE,ELENG,C1,C2,MCODE,
     $                  JCODE,MINC,NA,ILC,NE,NJ,NG,NEQ,MBD,
     $                  NLC,MXNEQ,NTS,NEG,DKS,QS,QD,QDG,T,DE,
     $                  GCODE,DEM,RM,EM,DLEM,DLTS,A,DR,IELC,
     $                  ZIG,SG,AG,G,E,AD,ITEN)
        END IF
      ELSE
        PRINT 40, 'ERROR MESSAGE; DIMENSION LIMITS EXCEEDED'
40      FORMAT(1X,T10,A/)
      END IF
      STOP
      END
```

```
C*************************************************************
C*                    SUBROUTINE  ANALYS                     *
C*************************************************************
C FUNCTION- MANEGMENT OF ANALYSIS COMPUTAION. IF NE AND NJ
C           ARE LESS THAN OR EQUAL TO MX, CALL FOR EACH LOAD
C           CONDITION DATA, SYSTEM, AND RESULT; ELSE PRINT
C           ERROR MESSAGE AND STOP.
C
      SUBROUTINE ANALYS(F,P,Q,SS,GCODE,NG,NEG,AREA,ZI,S,
     $                  EMOD,CTE,ELENG,E,DR,C1,C2,MCODE,
     $                  JCODE,MINC,NA,ILC,NE,NJ,NEQ,MBD,NLC,
     $                  MXNEQ,MX,EM,IELC,RM,A,G,AD)
      DIMENSION F(6,*),P(3,*),SS(MXNEQ,*),Q(*),AREA(*),E(*),
     $          ZI(*),EMOD(*),CTE(*),ELENG(*),C1(*),C2(*),
     $          MCODE(6,*),JCODE(3,*),MINC(2,*),NA(*),S(*),
     $          EM(*),ILC(*),IELC(*),A(NG,*),AD(*),G(*),
     $          GCODE(NG,*),RM(*),DR(*)
C
      IF(NE .LE. MX .AND. NJ .LE. MX) THEN
        DO 10 LC=1,NLC
          CALL DATA(F,P,Q,AREA,ZI,EMOD,CTE,ELENG,C1,C2,
     $              MCODE,NA,JCODE,MINC,NE,NJ,NEQ,MBD,LC,S,
     $              GCODE,NG,NEG)
          CALL SYSTEM(SS,Q,AREA,ZI,EMOD,ELENG,C1,C2,MCODE,
     $                NE,NEQ,MBD,LC,MXNEQ,G)
          CALL RESULT (F,P,Q,AREA,ZI,EMOD,ELENG,C1,C2,
     $                 MCODE,JCODE,EM,A,MINC,NE,NJ,ILC,LC,
     $                 GCODE,NG,NEG,E,DR,RM,IELC,NEQ,AD)
   10   CONTINUE
        ELSE
          PRINT 20, ' ERROR MESSAGE ;
     $                    AT LEAST NE OR NJ EXCEEDS MX;',
     $                         'INCREASE VALUE OF MX.'
   20   FORMAT(1X,T10, A/ T25, A/ )
        END IF
        RETURN
        END
```

```
C*************************************************************
C*                        SUBROUTINE  DATA                   *
C*************************************************************
C FUNCTION- FOR THE FIRST LOAD CONDTION, LC=1, CALL STRUCT
C           AND LOAD,FOR SUBSEQUENT LOAD CONDTIONS, L>1,
C           CALL LOAD.
C
      SUBROUTINE DATA(F,X,Q,AREA,ZI,EMOD,CTE,ELENG,C1,C2,
     $                MCODE,NA,JCODE,MINC,NE,NJ,NEQ,MBD,LC,
     $                S,GCODE,NG,NEG)
      DIMENSION F(6,*),X(3,*),Q(*),AREA(*),ZI(*),EMOD(*),
     $          ELENG(*),C1(*),C2(*),MCODE(6,*),JCODE(3,*),
     $          MINC(2,*),NA(*),S(*),GCODE(NG,*),CTE(*)
C
      IF(LC.EQ.1)THEN
        CALL STRUCT(X,AREA,ZI,EMOD,CTE,ELENG,C1,C2,MCODE,
     $              JCODE,MINC,NE,NJ,NEQ,MBD,S,GCODE,NG,NEG)
      END IF
      CALL LOAD(AREA,EMOD,CTE,ELENG,C1,C2,MCODE,JCODE,NE,
     $          NEQ,F,Q,NA,LC)
      RETURN
      END
```

```
C*************************************************************
C*                      SUBROUTINE  STRUCT                   *
C*************************************************************
C FUNCTION- READ AND ECHO THE MEMBER INCEDENCES, MINC(L,);
c           INITIALIZE THE ELEMENTS OF THE JOINT CODE MATRIX
c           JCODE,TO UNITY, READ AND ECHO FOR EACH JOINT
c           CONSTRAINT, THE JOINT NUMBERE, JNUM, AND JOINT
C           DIRECTION,JDIR, AND STORE A ZERO IN THE
C           CORRESPONDING LOCATION OF JCODE(END OF DATA
C           MARKER JNUM=0).
C
      SUBROUTINE STRUCT (X,AREA,ZI,EMOD,CTE,ELENG,C1,C2,
     $                   MCODE,JCODE,MINC,NE,NJ,NEQ,MBD,S,
     $                   GCODE,NG,NEG)
      DIMENSION X(3,*),AREA(*),ZI(*),EMOD(*),CTE(*),
     $          C1(*),GCODE(NG,*),C2(*),MCODE(6,*),
     $          MINC(2,*),JCODE(3,*),ELENG(*),S(*)
C
      PRINT *,'M  I N  C :'
      DO 10 I=1,NE
        READ(5,*) MINC(1,I),MINC(2,I)
        PRINT *, MINC(1,I),MINC(2,I)
   10 CONTINUE
      DO 30 J=1,NJ
        DO 20  L=1,3
          JCODE(L,J)=1
   20   CONTINUE
   30 CONTINUE
      PRINT 40
   40 FORMAT(1X,'JOINT COSTRAINTS:',4X,'JNUM',4X,'JDIR')
      READ(5,*) JNUM, JDIR
   50 IF(JNUM.NE.0)THEN
        PRINT 60,JNUM,JDIR
   60   FORMAT(1X,T22,I3,T29,I3)
        JCODE(JDIR,JNUM)=0
        READ(5,*) JNUM, JDIR
        GO TO 50
      END IF
      CALL CODES(JCODE,MINC,NE,NJ,MCODE,NEQ,GCODE,NG,NEG)
      MBD=MBAND(MCODE,NE)
      CALL PROP(MINC,NE,NJ,X,AREA,ZI,EMOD,CTE,ELENG,C1,C2,S)
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  CODES                  *
C*************************************************************
C FUNCTION- GENERATE THE JOINT CODE, JCODE, BY ASSIGNING
C           INTEGERS  IN SEQUENCE,BY COLUMNS, TO ALL NONZERO
C           ELEMENTS OF JCODE FROM 1 TO NEQ; GENERATE THE
C           MEMBER CODE, MCODE, BY TRANSFERRING VIA MINC
C           COLUMNS OF JCODE INTO COLUMNS MCODE.
C
      SUBROUTINE CODES (JCODE,MINC,NE,NJ,MCODE,NEQ,GCODE,NG,
     $                  NEG)
      DIMENSION JCODE(3,*),MINC(2,*),MCODE(6,*),GCODE(NG,*)
C
      PRINT *,'J  C  O  D  E :'
      NEQ=0
      DO 20 J=1,NJ
         DO 10 L=1,3
            IF(JCODE(L,J).NE.0)THEN
            NEQ=NEQ+1
            JCODE(L,J)=NEQ
            ENDIF
 10      CONTINUE
 20   CONTINUE
      DO 30 I=1,3
         PRINT *,(JCODE(I,J), J=1,NJ)
 30   CONTINUE
      DO 50 I=1,NE
        J=MINC(1,I)
        K=MINC(2,I)
        DO 40 L=1,3
          MCODE(L,I)=JCODE(L,J)
          MCODE(L+3,I)=JCODE(L,K)
 40      CONTINUE
 50   CONTINUE
      PRINT *, 'M  C  O  D  E: '
      DO 70 I=1, 6
        PRINT 60, (MCODE(I,J),J=1,NE)
 60     FORMAT(1X,T10,20I5)
 70   CONTINUE
      PRINT *, 'G  C  O  D  : '
      DO 80 JJ=1, NG
            READ(5,*) GCODE(JJ,1),GCODE(JJ,2)
            PRINT *, GCODE(JJ,1),GCODE(JJ,2)
 80   CONTINUE
      RETURN
      END
```

```
C*************************************************************
C*                        FUNCTION  MBAND                    *
C*************************************************************
C FUNCTION- COMPUTE THE HALF BANDWIDTH,MBAND,BY EQ. 6.2: IN
C           EACH COLUMN OF MCODE,THE FIRST AND LAST NONZERO
C           INTEGERS ARE THE SMA11EST AN 1ARGEST NONZERO
C           INTEGERS, RESPECTIVELY, OF THAT COLUMN. MBAND
C           IS THE MAXIMUM DIFFERENCE OF THE NONZERO
C           INTEGERS IN ANY COLUMN OF MCODE.
C
      FUNCTION MBAND(MCODE,NE)
      DIMENSION MCODE(6,*)
C
      MBAND=0
      DO 70 I=1,NE
        L=1
  10    IF(MCODE(L,I).EQ.0)THEN
           L=L+1
           GO TO 10
        END IF
  20    IS=MCODE(L,I)
  30    L=6
  40    IF(MCODE(L,I).EQ.0)THEN
          L=L-1
          GO TO 40
        END IF
  50    IL=MCODE(L,I)
  60    IDIF=IL-IS
        IF(IDIF.GT.MBAND)THEN
          MBAND=IDIF
        END IF
  70  CONTINUE
      PRINT *, ' MBAND  : '
      PRINT *, MBAND
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  PROP                   *
C*************************************************************
C FUNCTION- READ AND ECHO THE JOINT COORDINATES, X(L,J);
C           COMPUTE FOR EACH ELEMENT BY EQ.C.21 THE LENGTH,
C           ELENG(I), AND THE DIRECTION COSINES, C1(I),
C           C2(I);READ FOR EACH ELEMENT The CROSS SECTION
C           AREA,AREA(I), THE MOMENT OF INERTIA ABOUT THE
C           LOCAL Z(3)-AXIS.
C
      SUBROUTINE PROP (MINC,NE,NJ,X,AREA,ZI,EMOD,CTE,ELENG,
     $                 C1,C2,S)
      DIMENSION MINC(2,*),X(3,*),AREA(*),ZI(*),EMOD(*),
     $          C1(*),C2(*),S(*),ELENG(*),CTE(*)
C
      PRINT *, 'J O I N T  C O O R D I N A T E S :'
      DO 10 J=1,NJ
        READ(5,*) X(1,J),X(2,J)
        PRINT *, X(1,J),X(2,J)
   10   CONTINUE
      PRINT *,'                                    '
      PRINT 20
   20 FORMAT(1X,T4,'I',T10,'ELENG',T23,'AREA',T33,'ZI',T42,
C    $     'emod',T52,'CTE',T62,'C1',T70,'C2',T77,'S'//)
      DO 40 I=1,NE
         J=MINC(1,I)
         K=MINC(2,I)
         EL1=X(1,K)-X(1,J)
         EL2=X(2,K)-X(2,J)
         ELENG(I)=SQRT(EL1**2+EL2**2)
         C1(I)=EL1/ELENG(I)
         C2(I)=EL2/ELENG(I)
         READ(5,*) AREA(I),ZI(I),EMOD(I),CTE(I),S(I)
         PRINT 30, I,ELENG(I),AREA(I),ZI(I),EMOD(I),CTE(I),
     $             C1,C2(I),S(I)
   30    FORMAT(1X,T3,I2,T6,F10.2,T17,F10.3,T25,F10.2,T38,
     $          E7.2,T48,E7.2,T58,F5.2,T66,F5.2,T72,F8.2/)
   40 CONTINUE
      RETURN
      END
```

```
C*************************************************************
C*                      SUBROUTINE  LOAD                     *
C*************************************************************
C FUNCTION- INITIALIZE TO ZERO THE JOINT LOAD VECTOR,Q, THE
c           LOCAL ELEMENT(MEMBER) FORCE VECTOR,F, AND THE
C           NUMBER OF ACTIONS VECTOR,NA CALL JLOAD AND MACT.
C
      SUBROUTINE LOAD (AREA,EMOD,CTE,ELENG,C1,C2,MCODE,
     $                 JCODE,NE,NEQ,f,Q,NA,LC)
      DIMENSION AREA(*),EMOD(*),ELENG(*),C1(*),MCODE(6,*),
     $          JCODE(3,*),F(6,*),Q(*),NA(*),CTE(*),C2(*)
C
      DO 10 K=1,NEQ
         Q(K)=0
  10  CONTINUE
      DO 20 I=1,NE
        DO 20 L=1,6
          F(L,I)=0
          NA(I)=0
  20  CONTINUE
      CALL JLOAD(Q,JCODE,LC)
      PRINT 30
  30  FORMAT(1X,'ACTUAL JOINT LOAD:'/)
      DO 50 K= 1,NEQ
        PRINT 40, K,Q(K)
  40   FORMAT(1X,T20,'Q(',I2,')=',F11.4)
  50   CONTINUE
      CALL MACT(F,Q,AREA,EMOD,CTE,ELENG,C1,C2,MCODE,NA,NE,
     $          NEQ)
      RETURN
      END
```

```
C*************************************************************
C*                    SUBROUTINE  JLOAD                      *
C*************************************************************
C FUNCTION- READ THE JOINT NUMBER, THE JOINT DIRECTION,JDIR,
C           AND THE APPLIED FORCE;WHILE JNUM.NE.0,PRINT JNUM
C           FORCE,STORE FORCE IN Q,AND READ JNUM,JDIR,FORCE.
C
      SUBROUTINE JLOAD(Q,JCODE,LC)
      DIMENSION Q(*),JCODE(3,*)
C
      PRINT 10, LC
  10  FORMAT(1X,'LOAD CONDITION = ',I3/)
      READ(5,*) JNUM,JDIR,FORCE
      IF(JNUM.NE.0)THEN
        PRINT *,'                                        '
        PRINT *,'          JNUM       JDIR      FORCE'
      ELSE
        PRINT *, 'NO JOINT LOAD'
      END IF
  20  IF(JNUM.NE.0)THEN
         PRINT *, JNUM,JDIR,FORCE
         K=JCODE(JDIR,JNUM)
         Q(K)=FORCE
         READ(5,*) JNUM,JDIR,FORCE
         GO TO 20
      END IF
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  MACT                   *
C*************************************************************
C FUNCTION- READ THE MEMBER NUMBER, MN, THE, MEMBER ACTION
C           TYPE, MAT THE ACTION, ACT,AND THE DISTANCE,DIST;
C           WHILE MN.NE.0 PRINT MN, MAT, ACT, DIST,INCREMENT
C           THE ACTION COUNTER, COMPUTE AND ACCUMULATE THE
C           FIXED-END FORCES, AND READ MN, MAT,ACT,DIST;CALL
C           ASSEMF .
C
      SUBROUTINE MACT(F,Q,AREA,EMOD,CTE,ELENG,C1,C2,MCODE,
     $                NA,NE,NEQ)
      DIMENSION F(6,*),Q(*),AREA(*),EMOD(*),CTE(*),ELENG(*),
     $          c1(*),C2(*),MCODE(6,*),NA(*)
      REAL L
C
      READ(5,*) MN,MAT,ACT,DIST1,DIST2
      IF(MN.NE.0)THEN
        PRINT 10
   10   FORMAT(1X,'MEMBER ACTIONS :',T22,'MN',6X,'MAT',6X,
     $         'ACT',6x,'DIST1',6X,'DIST2')
      ELSE
      PRINT 20
   20 FORMAT(1X,'NO MEMBER ACTIONS')
      END IF
   30 IF(MN.NE.0) THEN
         PRINT 40,  MN,MAT,ACT,DIST1,DIST2
   40    FORMAT(1X,T21,I2,6X,I2,6X,F6.2,5X,F5.2,5X,F6.2)
         NA(MN)=NA(MN)+1
         P=ACT
         L=ELENG(MN)
         S1=DIST1/L
         S2=DIST2/L
         A1=S2**2+S2*S1+S1**2
         B1=S2**3+S2**2*S1+S2*S1**2+S1**3
         C=3*S2**2+2*S2*S1+S1**2
         D1=4*S2**3+3*S2**2*S1+2*S2*S1**2+S1**3
C
      IF(MAT.EQ.1) THEN
            F2=P*(-1-S1**2*(2*S1-3))
            F3=P*(-L*S1*(1-S1)**2)
            F5=P*S1**2*(2*S1-3)
            F6=P*L*S1**2*(1-S1)
            F(2,MN)=F(2,MN)+F2
            F(3,MN)=F(3,MN)+F3
            F(5,MN)=F(5,MN)+F5
            F(6,MN)=F(6,MN)+F6
C
      ELSE IF(MAT.EQ.2) THEN
            F2=P*L*(-0.5*(1-S1**4+2*S1**3-2*S1))
            F3=P*L*(-(L/12)*(1-3*S1**4+8*S1*3-6*S1**2))
```

```
            F5=P*L*(-0.5*(1+S1**4-2*S1**3))
            F6=P*L*((L/12)*(1+3*S1**4-4*S1**3))
            F(2,MN)=F(2,MN)+F2
            F(3,MN)=F(3,MN)+F3
            F(5,MN)=F(5,MN)+F5
            F(6,MN)=F(6,MN)+F6
C
      ELSE IF(MAT.EQ.3) THEN
            F1=-P*(1-S1)
            F4=-P*S1
            F(1,MN)=F(1,MN)+F1
            F(4,MN)=F(4,MN)+F4
C     T=P
      ELSE IF(MAT.EQ.4) THEN
            F1=EMOD(MN)*AREA(MN)*CTE(MN)*P
            F4=EMOD(MN)*AREA(MN)*CTE(MN)*(-P)
            F(1,MN)=F(1,MN)+F1
            F(4,MN)=F(4,MN)+F4
C
      ELSE IF(MAT.EQ.5) THEN
            F2=-(P*L/2)*(S2-S1)*(2-2*A1+B1)
            F3=-(P*L**2/12)*(S2-S1)*(6*(S2+S1)-8*A1+3*B1)
            F5=-(P*L/2)*(S2-S1)*(2*A1-B1)
            F6=(P*L**2/12)*(S2-S1)*(4*A1-3*B1)
            F(2,MN)=F(2,MN)+F2
            F(3,MN)=F(3,MN)+F3
            F(5,MN)=F(5,MN)+F5
            F(6,MN)=F(6,MN)+F6
C
      ELSE IF(MAT.EQ.6) THEN
            F2=-(P*L/20)*(S2-S1)*(10-5*C+2*D1)
            F3=-(P*L**2/60)*(S2-S1)*(10*(2*S2+S1)-10*C+3*D1)
            F5=-(P*L/20)*(S2-S1)*(5*C-2*D1)
            F6=(P*L**2/60)*(S2-S1)*(5*C-3*D1)
            F(2,MN)=F(2,MN)+F2
            F(3,MN)=F(3,MN)+F3
            F(5,MN)=F(5,MN)+F5
            F(6,MN)=F(6,MN)+F6
C
      ELSE IF(MAT.EQ.7) THEN
            F1=-P*L/2
            F4=-P*L/2
            F(1,MN)=F(1,MN)+F1
            F(4,MN)=F(4,MN)+F4
      END IF
      READ(5,*) MN,MAT,ACT,DIST1,DIST2
      GO TO 30
      END IF
      CALL ASSEMF(F,Q,C1,C2,MCODE,NA,NE,NEQ)
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  ASSEMF                 *
C*************************************************************
C FUNCTION- TRANSFORMATION AND ASSEMBLE THE LOCAL FIXED-END
C           FORCES F(L,I),TO PRDUCE THE EQUIVALENT JOINT
C           LOAD VECTOR,Q,BY EQS. 2.34,2.37,3.92,3.94 AND
C           THE FORCE TRANSFORMATION (sec. 2.4).
C
      SUBROUTINE ASSEMF(F,Q,C1,C2,MCODE,NA,NE,NEQ)
      DIMENSION F(6,*),Q(*),C1(*),C2(*),MCODE(6,*),NA(*)
C     STATMENT FUNCTION
      FG(C1I,C2I,FLX,FLY)=C1I*FLX+C2I*FLY
C
       DO 20 I=1, NE
          IF (NA(I) .NE.0) THEN
           DO 10 L=1, 6
            K= MCODE(L,I)
            IF (K .NE.0 ) THEN
             IF (L .EQ. 1) THEN
                Q(K) = Q(K)-FG(C1(I),-C2(I),F(1,I),F(2,I))
             ELSE IF (L .EQ.2) THEN
                Q(K) = Q(K)-FG(C2(I),C1(I),F(1,I),F(2,I))
             ELSE IF (L .EQ.3) THEN
                Q(K) = Q(K)-F(3,I)
             ELSE IF (L .EQ.4) THEN
                Q(K) = Q(K)-FG(C1(I),-C2(I),F(4,I),F(5,I))
             ELSE IF (L .EQ.5) THEN
                Q(K) = Q(K)-FG(C2(I),C1(I),F(4,I),F(5,I))
            ELSE
                Q(K) = Q(K)-F(6,I)
             END IF
          END IF
   10    CONTINUE
        END IF
   20  CONTINUE
       PRINT *, '                '
       PRINT *, 'LOCAL FIXED-END FORCES'
       PRINT 30
   30  FORMAT(1X,T2,'ELE.',T14,'F1',T26,'F2',T39,'F3',T52,
     $         'F4',T65,'F5',T78,'F6'/)
       DO 50 I=1,NE
          PRINT 40, I,(F(L,I),L=1,6)
   40     FORMAT(1X,T2,I2,T10,5(G12.3,5X),G12.3/)
   50  CONTINUE
       PRINT *, 'EQUIVALENT JOINT LOAD VECTOR '
       DO 70 K=1, NEQ
          PRINT 60, K,Q(K)
   60     FORMAT(1X,'Q(',I2,')=',G11.4)
   70  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C*                      SUBROUTINE  SYSTEM                   *
C*************************************************************
C FUNCTION- FOR THE FIRST LOAD CONDITION ,LC=1,CALL STIFF
C           AND SOLVE FOR SUBSEQUENT LOAD CONDITIONS, LC>1,
C           CALL SOLVE.
C
      SUBROUTINE SYSTEM(SS,Q,AREA,ZI,EMOD,ELENG,C1,C2,MCODE,
     $                  NE,NEQ,MBD,LC,MXNEQ,G)
      DIMENSION SS(MXNEQ,*),Q(*),AREA(*),ZI(*),EMOD(*),
     $          C1(*),C2(*),MCODE(6,*),G(*),ELEMG(*)
C
      IF(LC.EQ.1) THEN
        CALL STIFF(AREA,ZI,EMOD,ELENG,C1,C2,MCODE,NE,NEQ,
     $             MBD,MXNEQ,SS,G)
      END IF
      CALL SOLVE(SS,Q,NEQ,MBD,LC,MXNEQ)
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  STIFF                  *
C*************************************************************
C FUNCTION- INITIALAIZE THE SYSTEM STIFFNESS MATRIX, SS TO
C           ZERO; FOR EACH ELEMENT CALL ELEMS AND ASSEMS .
C
      SUBROUTINE STIFF(AREA,ZI,EMOD,ELENG,C1,C2,MCODE,NE,
     $                 NEQ,MBD,MXNEQ,SS,G)
      DIMENSION AREA(*),ZI(*),EMOD(*),ELENG(*),C1(*),C2(*),
     $          SS(MXNEQ,*),MCODE(6,*),G(*)
C
      MO=MBD+1
      DO 20  I=1,MO
         DO 10  J=1,NEQ
            SS(I,J)=0
  10     CONTINUE
  20  CONTINUE
      DO 40  N=1,NE
         CALL ELEMS(AREA,ZI,EMOD,ELENG,C1,C2,N,G)
         PRINT 30, (I,G(I),I=1,7)
  30     FORMAT(1X,' G',I2,'=',F12.4/)
         CALL ASSEMS(SS,G,MCODE,MBD,MXNEQ,N)
  40  CONTINUE
      PRINT *, 'CHECK FOR BANDED STIFFENESS MATRIX :'
      DO 60 K=1,MO
         PRINT 50, (SS(K,J), J=1,NEQ)
  50     FORMAT(1X,T3,9F12.4/)
  60  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C*                      SUBROUTINE  ELEMS                    *
C*************************************************************
C FUNCTION- FOR ELEMENT N, COMPUTE THE GLOBAL STIFFNESS
C           COEFFICIENTS, g(7).
C
      SUBROUTINE ELEMS(AREA,ZI,EMOD,ELENG,C1,C2,N,G)
      DIMENSION AREA(*),ZI(*),EMOD(*),ELENG(*),C1(*),C2(*),
     $          G(*)
C
      A =EMOD(N)*ZI(N)/ELENG(N)**3
      B =AREA(N)*ELENG(N)**2/ZI(N)
      PRINT *,'                '
      G(1)=A*(B*C1(N)**2+12*C2(N)**2)
      G(2)=A*C1(N)*C2(N)*(B-12)
      G(3)=A*(B*C2(N)**2+12*C1(N)**2)
      G(4)=-A*6*ELENG(N)*C2(N)
      G(5)=A*6*ELENG(N)*C1(N)
      G(6)=A*4*ELENG(N)**2
      G(7)=A*2*ELENG(N)**2
      RETURN
      END
```

```
C*************************************************************
C*                      SUBROUTINE  ASSEMS                   *
C*************************************************************
C FUNCTION- INITIALIZE INDEX BY EQ. 7.4; ASSIGN STIFFNESS
C           COEFFICIENTS, G(L),OF ELEMENT N TO THE SYSTEM
C           STIFFNESS BAND MATRIX,SS, BY INDEX,MCODE,EQ.6.7.
C
      SUBROUTINE ASSEMS(SS,G,MCODE,MBD,MXNEQ,N)
      DIMENSION SS(MXNEQ,*),G(*),MCODE(6,*),INDEX(6,6)
      DATA INDEX /1,2,4,-1,-2,4, 2,3,5,-2,-3,5, 4,5,6,-4,
     $ -5,7,-1,-2,-4,1,2,-4, -2,-3,-5,2,3,-5, 4,5,7,-4,-5,6/

      IF(N.EQ.1)THEN
         PRINT *,'CHECK INDEX MATRIX  :'
         DO 10 J=1,6
            PRINT *, (INDEX(J,I),I=1,6)
            PRINT *, '                 '
 10      CONTINUE
      END IF
      DO 30 JE=1,6
      J= MCODE(JE,N)
        IF(J.NE.0) THEN
           DO 20 IE=1,JE
           I=MCODE(IE,N)
              IF(I.NE.0) THEN
              K=I-J+MBD+1
              L=INDEX(IE,JE)
                   IF(L.GT.0)THEN
                   SS(K,J)=SS(K,J)+G(L)
                   ELSE
                   SS(K,J)=SS(K,J)-G(-L)
                   END IF
              END IF
 20       CONTINUE
        END IF
 30   CONTINUE
      RETURN
      END
```

```
C*************************************************************
C                       SUBROUTINE  SOLVE                     *
C*************************************************************
C FUNCTION- FOR THE FIRST LOAD CONDITION , LC=1, CALL SPBFA
C           AND SPBSL; FOR SUBSEQUENT LOAD CONDITION, LC>1,
C           CALL SPBSL.
C
      SUBROUTINE SOLVE (SS,Q,NEQ,MBD,LC,MXNEQ)
      DIMENSION SS(MXNEQ,*),Q(*)
C
      IF(LC.EQ.1) THEN
        CALL SPBFA(SS,MXNEQ,NEQ,MBD,INFO)
        IF (INFO.NE.0) THEN
           PRINT *, 'ERROR MESSAGE;SINGULARITY;'
           STOP
        END IF
      END IF
      CALL SPBSL(SS,MXNEQ,NEQ,MBD,Q)
      PRINT *, 'DISPLACEMENT VECTOR'
      PRINT 10,(I,Q(I),I=1,NEQ)
   10 FORMAT(1X,'Q',I2,'=',F11.5/)
      RETURN
      END
```

```
C*************************************************************
      SUBROUTINE SAXPY(N,SA,SX,INCX,SY,INCY)
C     CONSTANT TIMES A VECTOR PLUS A VECTOR.
C     USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      REAL SX(1),SY(1),SA
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      IF(N.LE.0)RETURN
      IF (SA .EQ. 0.0) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        SY(IY) = SY(IY) + SA*SX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
   20 M = MOD(N,4)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        SY(I) = SY(I) + SA*SX(I)
   30 CONTINUE
      IF( N .LT. 4 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
        SY(I) = SY(I) + SA*SX(I)
        SY(I + 1) = SY(I + 1) + SA*SX(I + 1)
        SY(I + 2) = SY(I + 2) + SA*SX(I + 2)
        SY(I + 3) = SY(I + 3) + SA*SX(I + 3)
   50 CONTINUE
      RETURN
      END
C
      REAL FUNCTION SDOT(N,SX,INCX,SY,INCY)
C
C     FORMS THE DOT PRODUCT OF TWO VECTORS.
```

```
C     USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      REAL SX(1),SY(1),STEMP
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      SDOT = 0.0E0
      STEMP = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        STEMP = STEMP + SX(IX)*SY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      SDOT = STEMP
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        STEMP = STEMP + SX(I)*SY(I)
   30 CONTINUE
      IF( N .LT. 5 ) GO TO 60
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        STEMP = STEMP + SX(I)*SY(I) + SX(I + 1)*SY(I + 1)
            SX(I+2)*SY(I+2)+SX(I+3)*SY(I+3)+SX(I+4)*SY(I+4)
   50 CONTINUE
   60 SDOT = STEMP
      RETURN
      END
C
      SUBROUTINE SPBFA(ABD,LDA,N,M,INFO)
      INTEGER LDA,N,M,INFO
      REAL ABD(LDA,1)
C
C     SPBFA FACTORS A REAL SYMMETRIC POSITIVE DEFINITE
```

```
C     MATRIX STORED IN BAND FORM.
C
C   SPBFA IS USUALLY CALLED BY SPBCO, BUT IT CAN BE CALLED
C   DIRECTLY WITH A SAVING IN TIME IF  RCOND  IS NOT NEEDED.
C
C   ON ENTRY
C
C     ABD  REAL(LDA, N)
C       THE MATRIX TO BE FACTORED.  THE COLUMNS OF THE UPPER
C          TRIANGLE ARE STORED IN THE COLUMNS OF ABD AND THE
C          DIAGONALS OF THE UPPER TRIANGLE ARE STORED IN THE
C          ROWS OF ABD .  SEE THE COMMENTS BELOW FOR DETAILS
C
C        LDA INTEGER
C            THE LEADING DIMENSION OF THE ARRAY  ABD .
C            LDA MUST BE .GE. M + 1 .
C
C        N   INTEGER
C            THE ORDER OF THE MATRIX  A .
C
C        M   INTEGER
C            THE NUMBER OF DIAGONALS ABOVE THE MAIN DIAGONAL
C                0 .LE. M .LT. N .
C
C   ON RETURN
C
C      ABD   AN UPPER TRIANGULAR MATRIX  R , STORED IN BAND
C                FORM, SO THAT  A = TRANS(R)*R .
C
C      INFO   INTEGER
C             = 0  FOR NORMAL RETURN.
C             = K  IF THE LEADING MINOR OF ORDER  K  IS NOT
C               POSITIVE DEFINITE.
C
C     BAND STORAGE
C
C       IF  A  IS A SYMMETRIC POSITIVE DEFINITE BAND MATRIX,
C       THE FOLLOWING PROGRAM SEGMENT WILL SET UP THE INPUT.
C
C                   M = (BAND WIDTH ABOVE DIAGONAL)
C                   DO 20 J = 1, N
C                      I1 = MAXO(1, J-M)
C                      DO 10 I = I1, J
C                         K = I-J+M+1
C                         ABD(K,J) = A(I,J)
C               10     CONTINUE
C               20  CONTINUE
C
C  LINPACK.  THIS VERSION DATED 08/14/78 .
C  CLEVE MOLER, UNI. OF NEW MEXICO,ARGONNE NATIONAL LAB.
C
```

```
C  SUBROUTINES AND FUNCTIONS
C
C  BLAS SDOT
C  FORTRAN MAX0,SQRT
C
C  INTERNAL VARIABLES
C
      REAL SDOT,T
      REAL S
      INTEGER IK,J,JK,K,MU
C     BEGIN BLOCK WITH ...EXITS TO 40
C
C
         DO 30 J = 1, N
            INFO = J
            S = 0.0E0
            IK = M + 1
            JK = MAX0(J-M,1)
            MU = MAX0(M+2-J,1)
            IF (M .LT. MU) GO TO 20
            DO 10 K = MU, M
             T=ABD(K,J)-SDOT(K-MU,ABD(IK,JK),1,ABD(MU,J),1)
             T = T/ABD(M+1,JK)
             ABD(K,J) = T
             S = S + T*T
             IK = IK - 1
             JK = JK + 1
   10       CONTINUE
   20       CONTINUE
            S = ABD(M+1,J) - S
C     ......EXIT
            IF (S .LE. 0.0E0) GO TO 40
            ABD(M+1,J) = SQRT(S)
   30    CONTINUE
         INFO = 0
   40 CONTINUE
      RETURN
      END
C
      SUBROUTINE SPBSL(ABD,LDA,N,M,B)
      INTEGER LDA,N,M
      REAL ABD(LDA,1),B(1)
C
C     SPBSL SOLVES THE REAL SYMMETRIC POSITIVE DEFINITE
C     BAND SYSTEM  A*X = B
C     USING THE FACTORS COMPUTED BY SPBCO OR SPBFA.
C
C     ON ENTRY
C
C        ABD   REAL(LDA, N)
C              THE OUTPUT FROM SPBCO OR SPBFA.
```

```
C
C         LDA    INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  ABD .
C
C         N      INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C         M      INTEGER
C                THE NO. OF DIAGONALS ABOVE THE MAIN DIAGONAL.
C
C         B      REAL(N)
C                THE RIGHT HAND SIDE VECTOR.
C
C      ON RETURN
C
C         B        THE SOLUTION VECTOR  X .
C
C      ERROR CONDITION
C
C       A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR
C       CONTAINS A ZERO ON THE DIAGONAL.  TECHNICALLY THIS
C       INDICATES SINGULARITY BUT IT IS USUALLY CAUSED BY
C       IMPROPER SUBROUTINE ARGUMENTS.  IT WILL NOT OCCUR IF
C       THE SUBROUTINES ARE CALLED CORRECTLY AND  INFO.EQ.0 .
C
C      TO COMPUTE  INVERSE(A) * C  WHERE  C  IS A MATRIX
C      WITH  P  COLUMNS
C            CALL SPBCO(ABD,LDA,N,RCOND,Z,INFO)
C            IF (RCOND IS TOO SMALL .OR. INFO .NE. 0) GO TO..
C            DO 10 J = 1, P
C               CALL SPBSL(ABD,LDA,N,C(1,J))
C         10 CONTINUE
C
C      LINPACK.  THIS VERSION DATED 08/14/78 .
C      CLEVE MOLER, UNI. OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C      SUBROUTINES AND FUNCTIONS
C
C      BLAS SAXPY,SDOT
C      FORTRAN MIN0
C
C      INTERNAL VARIABLES
C
       REAL SDOT,T
       INTEGER K,KB,LA,LB,LM
C
C      SOLVE TRANS(R)*Y = B
C
       DO 10 K = 1, N
          LM = MIN0(K-1,M)
          LA = M + 1 - LM
```

```
            LB = K - LM
            T = SDOT(LM,ABD(LA,K),1,B(LB),1)
            B(K) = (B(K) - T)/ABD(M+1,K)
   10    CONTINUE
C
C        SOLVE R*X = Y
C
         DO 20 KB = 1, N
            K = N + 1 - KB
            LM = MINO(K-1,M)
            LA = M + 1 - LM
            LB = K - LM
            B(K) = B(K)/ABD(M+1,K)
            T = -B(K)
            CALL SAXPY(LM,T,ABD(LA,K),1,B(LB),1)
   20    CONTINUE
         RETURN
         END
```

```
C*************************************************************
C*                      SUBROUTINE  RESULT                   *
C*************************************************************
C FUNCTION- INITIALIZE THE JOINT FORCE MATRIX,P, TO ZERO ;
C           CALL FORCES AND OUTPUT.
C
      SUBROUTINE RESULT (F,P,Q,AREA,ZI,EMOD,ELENG,C1,C2,
     $                   MCODE,JCODE,EM,A,MINC,NE,NJ,ILC,LC,
     $                   GCODE,NG,NEG,E,DR,RM,IELC,NEQ,AD)
      DIMENSION F(6,*),Q(*),AREA(*),ZI(*),EMOD(*),ELENG(*),
     $          MCODE(6,*),JCODE(3,*),MINC(2,*),P(3,*),
     $          GCODE(NG,*),E(*),DR(*),RM(*),IELC(*),AD(*),
     $          C1(*),C2(*),EM(*),A(NG,*),ILC(*)
C
      DO 20  J=1,NJ
        DO 10  L=1,3
           P(L,J)=0
  10    CONTINUE
  20  CONTINUE
      CALL FORCES (F,P,Q,AREA,ZI,EMOD,ELENG,C1,C2,MCODE,
     $             MINC,NE)
      CALL DEFLE(Q,JCODE,GCODE,MINC,NG,NEG,E,DR,LC,C2,ILC,
     $           RM,IELC,EM,A,NEQ,AD)
      CALL OUTPUT(F,P,Q,JCODE,NE,NJ)
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  FORCES                 *
C*************************************************************
C FUNCTION- FOR EACH ELEMENT CALL ELEMF AND JOINTF.
C
      SUBROUTINE FORCES(F,P,Q,AREA,ZI,EMOD,ELENG,C1,C2,
     $                  MCODE,MINC,NE)
      DIMENSION F(6,*),P(3,*),Q(*),AREA(*),ZI(*),ELENG(*),
     $          C1(*),C2(*),MCODE(6,*),MINC(2,*),EMOD(*)
C
      DO 10 I=1,NE
         CALL ELEMF(F,Q,AREA,ZI,EMOD,ELENG,C1,C2,MCODE,I)
         CALL JOINTF(F,P,C1,C2,MINC,I)
  10  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  ELEMF                  *
C*************************************************************
C FUNCTION- COMPUTE THE LOCAL FORCES OF I, F(6,I): DETERMINE
C           THE GLOBAL ELEMENT DISPLACEMENTS, D(6), FROM THE
C           JOINT DISPLACEMENTS VECTOR,Q, VIA MCODE; COMPUTE
C           THE LOCAL FORCES AT THE A-END OF THE ELEMENT BY
C           EQS. 2.34, 2.36, 3.59;USE EQUILIBRIUM TO COMPUTE
C           THE LOCAL FORCES AT THE B-END OF THE ELEMENT AND
C           EQ. 3.95 TO COMPUTE THE ACTUAL ELEMENT FORCES .
C
      SUBROUTINE ELEMF(F,Q,AREA,ZI,EMOD,ELENG,C1,C2,MCODE,I)
      DIMENSION F(6,*),Q(*),AREA(*),ZI(*),EMOD(*),ELENG(*),
     $          C1(*),C2(*),MCODE(6,*),D(6)
C
      DO 10 L=1,6
       K=MCODE(L,I)
        IF(MCODE(L,I).EQ.0) THEN
          D(L)=0
        ELSE
       D(L)=Q(K)
       END IF
  10  CONTINUE
      A=(EMOD(I)*ZI(I))/(ELENG(I)**3)
      B=AREA(I)*(ELENG(I)**2)/ZI(I)
      D11=C1(I)*D(1)+C2(I)*D(2)
      D12=-C2(I)*D(1)+C1(I)*D(2)
      D13=D(3)
      D14=C1(I)*D(4)+C2(I)*D(5)
      D15=-C2(I)*D(4)+C1(I)*D(5)
      D16=D(6)
      F1=A*B*(D11-D14)
      F2=A*(12*(D12-D15)+6*ELENG(I)*(D13+D16))
      F3=A*(6*ELENG(I)*(D12-D15)+2*ELENG(I)**2*(2*D13+D16))
      F(1,I)=F(1,I)+F1
      F(2,I)=F(2,I)+F2
      F(3,I)=F(3,I)+F3
      F(4,I)=F(4,I)-F1
      F(5,I)=F(5,I)-F2
      F(6,I)=F(6,I)+F2*ELENG(I)-F3
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  JOINTF                 *
C*************************************************************
C FUNCTION- TRANSFORM THE LOCAL FORCES OF ELEMENT I, F(6,I)
C           TO GLOBAL FORCES AND ASSIGN THEM TO THE JOINT
C           FORCES MATRIX, P, BY EQS. 2.29, 2.34, 2.37, AND
C           MINC .
C
      SUBROUTINE JOINTF(F,P,C1,C2,MINC,I)
      DIMENSION F(6,*),P(3,*),C1(*),C2(*),MINC(2,*)
      FG(C1I,C2I,FLX,FLY)=C1I*FLX+C2I*FLY
C
      J= MINC(1,I)
      K= MINC(2,I)
      P(1,J)= P(1,J)+FG(C1(I),-C2(I),F(1,I),F(2,I))
      P(2,J)= P(2,J)+FG(C2(I),C1(I),F(1,I),F(2,I))
      P(3,J)= P(3,J)+F(3,I)
      P(1,K)= P(1,K)+FG(C1(I),-C2(I),F(4,I),F(5,I))
      P(2,K)= P(2,K)+FG(C2(I),C1(I),F(4,I),F(5,I))
      P(3,K)= P(3,K)+F(6,I)
      RETURN
      END
```

```
C*************************************************************
C*                        SUBROUTINE  DEFLE                  *
C*************************************************************
C FUNCTION- GENERATE THE RELATIONSHIP MATRIX BETWEEN JOINTS
C           DISPLACEMENTS AND MEMBERS DEFLECTIONS, THEN
C           COMPUTE THE MEMBER DEFLECTION AND DEFLECTION
C           RATIO FOR EACH LOAD CONDITION. SAVE THE CURRENT
C           LARGEST DEFLECTION EM(I) AND KEEP A RECORD OF
C           CORRESPONDING LOAD CONDITION INDICATOR, IELC(I)
C           CALL MAXDR.
C
      SUBROUTINE DEFLE(Q,JCODE,GCODE,MINC,NG,NEG,E,DR,LC,C2,
     $                 ILC,RM,IELC,EM,A,NEQ,AD)
      DIMENSION Q(*),JCODE(3,*),GCODE(NG,*),MINC(2,*),E(*),
     $          ILC(*),RM(*),EM(*),IELC(*),A(NG,*),AD(*),
     $          C2(*),DR(*)
C
      IF(LC.EQ.1) THEN
         DO 20 I=1, NG
            E(I)=0.0
            DR(I)=0.0
            DO 10 J=1, NEQ
               A(I,J)=0
 10         CONTINUE
 20      CONTINUE
      END IF
      DO 40 JE=1, NG
         IF(GCODE(JE,NEG).EQ.0) THEN
            I=GCODE(JE,1)
            J=MINC(1,I)
            K=MINC(2,I)
            L=JCODE(1,J)
            N=JCODE(1,K)
            IF(C2(I).GE.0) THEN
               A(JE,N)= 1
               A(JE,L)=-1
               E(J)=Q(N)-Q(L)
            ELSE
               A(JE,N)=-1
               A(JE,L)= 1
               E(JE)=Q(L)-Q(N)
            END IF
            DR(JE)=ABS(E(JE)/AD(JE))
         ELSE
            DO 30 M=1, NEG
               I=GCODE(JE,M)
               J=MINC(1,I)
               K=MINC(2,I)
               L=JCODE(2,J)
               N=JCODE(2,K)
               IF(M.NE.NEG) THEN
```

```
                  A(JE,L)=-0.5
                  A(JE,N)= 1
                  EE=Q(N)-0.5*Q(L)
               ELSE
                  A(JE,N)=-0.5
                  E(JE)=EE-0.5*Q(N)
                  DR(JE)=ABS(E(JE)/AD(JE))
                  EE=0.0
               END IF
30          CONTINUE
         END IF
         IF(LC.EQ.1) THEN
            EM(JE)=E(JE)
            IELC(JE)=LC
         ELSE
            IF(ABS(E(JE)).GT.ABS(EM(JE))) THEN
                EM(JE)=E(JE)
                IELC(JE)=LC
            END IF
         END IF
40    CONTINUE
      PRINT 50
50    FORMAT(1X,//' A ( NG , NEQ ) '/)
      IF(LC.EQ.1)THEN
        DO 70 IN=1,NG
           PRINT 60, (A(IN,J),J=1,NEQ)
60         FORMAT(1X,T3,9F4.1/)
70       CONTINUE
      END IF
      CALL MAXDR(LC,ILC,DR,NG,RM)
      IF(LC.EQ.3) THEN
        PRINT 80
80      FORMAT(1X,'MEMBER DEFLE',T35,'MAX. DEFLE',T60,
     $          'IELC(I)'/)
       DO 100 I=1, NG
          PRINT 90, I,E(I),I,EM(I),I,IELC(I)
90        FORMAT(1X,'E(',I2,')=',F12.4,T35,'EM(',I2,')=',
     $           F12.4,T60,'IELC(',I2,')=',I2)
100   CONTINUE
          PRINT *, '                    '
          PRINT 110
110       FORMAT(1X,'DEFLE RATIO',T35,'MAX. DEFLE RATIO',T60,
     $           'ILC(I)'/)
       DO 130 I=1, NG
          PRINT 120, I,DR(I),I,RM(I),I,ILC(I)
120       FORMAT(1X,'DR(',I2,')=',F12.4,T35,'RM(',I2,')=',
     $           F12.4,T60,'ILC(',I2,')=',I2)
130   CONTINUE
      END IF
      RETURN
      END
```

```
C*************************************************************
C                      SUBROUTINE  MAXDR                      *
C*************************************************************
C FUNCTION- SAVE THE CURRENT LARGEST VALUE AS THE CURRENT
C           DESIGN DEFLE. RATIO DR(I) AND KEEP A RECORD OF
C           CORRESPONDING LOAD CONDITION INDICATOR(ILC(I)).
C
      SUBROUTINE MAXDR(LC,ILC,DR,NG,RM)
      DIMENSION ILC(*),DR(*),RM(*)
C
      DO 10 I=1, NG
         IF(LC.EQ.1) THEN
            RM(I)=DR(I)
            ILC(I)=LC
         ELSE
            IF(DR(I).GT.RM(I)) THEN
               RM(I)=DR(I)
               ILC(I)=LC
            END IF
         END IF
  10  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C*                      SUBROUTINE  OUTPUT                   *
C*************************************************************
C FUNCTION- USE THE JOINT DISPLACEMENT VECTOR, Q, AND JCODE
C           TO PRINT THE JOINT DISPLACEMENTS ( INCLUDING
C           JOINT CONSTRAINTS); PRINT THE LOCAL ELEMENT
C           FORCES, F(6,NE);PRINT THE JOINT FORCES, P(3,NJ).
C
      SUBROUTINE OUTPUT(F,P,Q,JCODE,NE,NJ)
      DIMENSION F(6,*),P(3,*),JCODE(3,*),U(3),Q(*)
C
      PRINT 10
  10  FORMAT(1X,'JOINT DISPLACEMENTS: JOINT  DIRECTION 1
     $       DIRECTION2      DIRECTION3')
      DO 40 J=1,NJ
         DO 20 L=1,3
            K=JCODE(L,J)
            IF(K.EQ.0) THEN
                U(L)=0
            ELSE
                U(L)=Q(K)
            END IF
  20     CONTINUE
      PRINT 30,J,(U(L),L=1,3)
  30  FORMAT(1X,T21,I3,T28,G11.4,7X,G11.4,14X,G11.4)
  40  CONTINUE
      PRINT *, '                                '
      PRINT 50
  50  FORMAT (1X,'ACTUAL LOCAL ELEMENT FORCES:'/'ELE.    F1',
     $         13X,'F2',12X,'F3',13X,'F4',14X,'F5',13X,'F6')
      DO 70 I=1,NE
        PRINT 60,I,(F(L,I),L=1,6)
  60    FORMAT(1X,' ',T2,I2,T7,5(G11.4,4X),G11.4)
  70  CONTINUE
      PRINT *, '                '
      PRINT 80
  80  FORMAT(1X,'JOINT FORCES:'/' JOINT    DIRECTION1
     $         DIRECTION2       DIRECTION3')
      DO 100 J=1,N
        PRINT 90, J,(P(L,J),L=1,3)
  90    FORMAT(1X,'  ',T3,I3,T9,G11.4,6X,G11.4,6X,G11.4)
  100 CONTINUE
      PRINT 110
  110 FORMAT(1X,/)
      RETURN
      END
```

```
C*************************************************************
C*                      SUBROUTINE  DESIGN                   *
C*************************************************************
C FUNCTION- TO SET UP ALL DATA REQUIRED FOR THE DESIGN
C           ITERATION PROBLEM (STRUCTURAL PROPERTIES AND
C           LOADS), AND SET THE PERMENENT STORAGE LOACTIONS
C           FOR FIXED END FORCES AND EQUIVALENT JOINT FORCES
C           ONE SET FOR EACH LOAD CONDITION.
C
      SUBROUTINE DESIGN(FF,QQ,F,P,Q,SS,AREA,ZI,S,EMOD,DKGA,
     $                  DKGI,DAS,DIS,CTE,ELENG,C1,C2,MCODE,
     $                  JCODE,MINC,NA,ILC,NE,NJ,NG,NEQ,MBD,
     $                  NLC,MXNEQ,NTS,NEG,DKS,QS,QD,QDG,
     $                  T,DE,GCODE,DEM,RM,EM,DLEM,DLTS,A,DR,
     $                  IELC,ZIG,SG,AG,G,E,AD,ITEN)
      DIMENSION SS(MXNEQ,*),QQ(60,*),F(6,*),P(3,*),Q(*),
     $          AREA(*),ZI(*),EMOD(*),CTE(*),ELENG(*),EM(*),
     $          AD(*),C1(*),C2(*),MCODE(6,*),JCODE(3,*),
     $          NA(*),SG(*),E(*),ILC(*),FF(6,40,*),DKGA(*),
     $          DKS(MXNEQ,*),DLTS(*),G(*),QD(MXNEQ,*),QS(*),
     $          T(NE,*),A(NG,*),DE(NG,*),ZIG(*),GCODE(NG,*),
     $          DEM(NG,*),IELC(*),DLEM(*),DKGI(*),RM(*),
     $          AG(*),DR(*),MINC(2,*),S(*),QDG(MXNEQ,*),
C
C     PRINT *, '*******************************************'
C     PRINT *, '               D  E  S  I  G  N            '
C     PRINT *, '*******************************************'

      DO 40 LC=1, NLC
         CALL DATA(F,P,Q,AREA,ZI,EMOD,CTE,ELENG,C1,C2,MCODE,
     $             JCODE,MINC,NA,NE,NJ,NEQ,MBD,LC,S,GCODE,
     $             NG,NEG)
          DO 20 I=1, NE
             DO 10 J=1, 6
                FF(J,I,LC)=F(J,I)
10           CONTINUE
20        CONTINUE
          DO 30 K=1, NEQ
             QQ(K,LC)=Q(K)
30        CONTINUE
40    CONTINUE
      PRINT *, 'FIXED END FORCES, FF(J,I,LC) '
      DO 80 LC=1, NLC
       PRINT 50, LC
50     FORMAT(1X,'FIXED END FORCES FOR LOAD CONDITION=',I2/)
       PRINT *, '----------------------------------------'
       DO 70 I=1,NE
          DO 60 J=1,6
            PRINT 55, J,I,LC,FF(J,I,LC)
55          FORMAT(1X,T3,'FF(',I2,',',I2,',',I2,')= ',F10.3)
60        CONTINUE
```

```
 70  CONTINUE
 80 CONTINUE
    PRINT *,'APPLIED LOAD VECTOR,QQ(K,LC),ONE FOR EACH Lc'
    PRINT *,'-------------------------------------------'
    PRINT *, '                    '
    DO 110 LC =1, NLC
      DO 100 K=1 ,NEQ
        PRINT 90, K,LC,QQ(K,LC)
 90     FORMAT(1X,T3,'QQ(',I2,',',I2,')= ',F10.4)
100   CONTINUE
110 CONTINUE
    DO 130 I=1, NE
       DO 120 J=1, NG
          READ(5,*) T(I,J)
120    CONTINUE
130 CONTINUE
    PRINT *, ' T ( NE , NG ) : '
    DO 150 I=1, NE
      PRINT 140, (T(I,J),J=1,NG)
140   FORMAT(1X,T3,20F5.1)
150 CONTINUE
    CALL DESITR (FF,QQ,SS,P,AREA,ZI,S,EMOD,ELENG,C1,C2,
   $             MCOODE,GCODE,MINC,ILC,NE,NJ,NEQ,MBD,NLC,
   $             MXNEQ,NTS,F,Q,DKGA,dkgi,DAS,DIS,NG,DKS,
   $             QS,QD,QDG,A,DE,ITEN,NEG,DEM,RM,,DLEM,
   $             DLTS,T,EM,DR,IELC,ZIG,SG,AG,G,E,AD,GCODE)
    RETURN
    END
```

```
C*************************************************************
C*                        SUBROUTINE DESITR                  *
C*************************************************************
C FUNCTION- READ THE ITRATION PARAMETERS; FOR EACH TRAIL
C           STRUCTURE SET UP TEMPORARY F, Q VECTORS FOR A
C           PARTICULAR LOAD CONDITION AND COMPUTE THE SYSTEM
C           SOLUTION AND RESPONSE THEN CALL SENS AND SENSD
C           TO COMPUTE THE SENSITIVITY DATA REQUIRED FOR
C           DEPENDENT SECTION PROPERTIES COMPUTATIONS. CALL
C           CONVERGE AND TEST STRUCTURE FOR CONVERGENCE, IF
C           NOT COMPUTE NEW TRIAL STRUCTURE AND ASSIGN NEW
C           MEMBER PROPERTIES TO EACH ELEMENT. ITRATION
C           TERMINATES EITHER BY CONVERGENCE OR BY EXCEEDING
C           THE NUMBER OF TRIAL STRUCTURES, NTS.
C
      SUBROUTINE DESITR (FF,QQ,SS,P,AREA,ZI,S,EMOD,ELENG,C1,
     $                   C2,MCODE,JCODE,MINC,ILC,NE,NJ,NEQ,
     $                   MBD,NLC,MXNEQ,NTS,F,Q,DKGA,DKGI,
     $                   DAS,DIS,NG,DKS,QS,QD,QDG,A,DE,ITEN,
     $                   NEG,DEM,RM,DLEM,DLTS,T,EM,DR,IELC,
     $                   ZIG,SG,AG,G,E,AD,GCODE)
      DIMENSION FF(6,40,*),QQ(60,*),SS(MXNEQ,*),P(3,*),Q(*),
     $          S(*),F(6,*),EMOD(*),ELENG(*),C1(*),C2(*),
     $          MCODE(6,*),JCODE(3,*),MINC(2,*),ILC(*),
     $          DKS(MXNEQ,*),QS(*),QD(MXNEQ,*),QDG(MXNEQ,*),
     $          AG(*),T(NE,*),A(NG,*),DE(NG,*),GCODE(NG,*),
     $          RM(*),G(*),E(*),EM(*),DLEM(*),DLTS(*),DR(*),
     $          IECL(*),DEM(NG,*),ZIG(*),DKGI(*),SG(*),
     $          AREA(*),ZI(*),DKGA(*),AD(*)
C
      READ(5,*) CP, ITEN
      PRINT 8
    8 FORMAT(1X,/)
      PRINT *, '      T R I A L   S T R U C T U R E S      '
      PRINT *, '   ======================================  '
      M=1
   10 IF(M .LE. NTS) THEN
         DO 100 LC = 1,NLC
           DO 30 I = 1,NE
               DO 20 J =1,6
                  F(J,I)=FF(J,I,LC)
   20          CONTINUE
   30       CONTINUE
            DO 40 K = 1, NEQ
               Q(K)=QQ(K,LC)
   40       CONTINUE
            IF(LC.EQ.3) THEN
              PRINT 50, M
   50         FORMAT(1X,T17,'M(ITERATION COUNTER) = ',I3/T15,
     $              32('='))
              PRINT 60, LC
```

```
 60         FORMAT(1X,T20,'LOAD CONDITION=',I2/T18,30('='))
            PRINT 70
 70         FORMAT(1X,/T3,'ELE.',T14,'AREA(I)',T29,'S(I)',
   $              T3,'ZI(I)')
            DO 90 I=1,NE
              PRINT 80, I,AREA(I),S(I),ZI(I)
 80           FORMAT(1X,T2,I3,T8,F15.4,T27,F15.4,T41,F15.4)
 90         CONTINUE
          END IF
          CALL SYSTEM(SS,Q,AREA,ZI,EMOD,ELENG,C1,C2,MCODE,
   $                  NE,NEQ,MBD,LC,MXNEQ,G)
          CALL RESULT (F;P,Q,AREA,ZI,EMOD,ELENG,C1,C2,
   $                   MCODE,JCODE,EM,A,MINC,NE,NJ,ILC,LC,
   $                   GCODE,NG,NEG,E,DR,RM,IELC,NEQ,AD)
          CALL SENS(C1,C2,EMOD,ELENG,NE,DKGA,DKGI,DAS,S,
   $               DIS,Q,NEQ,MXNEQ,SS,MCODE,JCODE,MINC,
   $               NG,DKS,QS,QD,LC,MBD,ITEN)
          CALL SENSD(LC,NE,NG,NEQ,QD,QDG,T,A,DE,MXNEQ,DEM,
   $                 EM,NEG,GCODE,IELC,M)
100     CONTINUE
        CALL SOLVED(DEM,EM,DLEM,NG,DLTS,AD)
        IF(M.EQ.1) THEN
           DO 120 IJ=1,NG
              IM=GCODE(IJ,1)
              SG(IJ)=S(IM)
              ZIG(IJ)=ZI(IM)
              AG(IJ)=AREA(IM)
              PRINT 110,IJ,SG(IJ)
110           FORMAT(1X,T10,'SG(',I2,')=',F10.4)
120        CONTINUE
        END IF
        CALL CONVRG(RM,AG,ZIG,SG,CP,NG,NTS,M,ILC,DLTS,ITEN)
        CALL ASSIGN(AREA,S,ZI,NG,GCODE,NEG,ZIG,SG,AG)
        GO TO 10
     ELSE
        STOP
     END IF
     RETURN
     END
```

```
C*************************************************************
C                        SUBROUTINE  SENS                     *
C*************************************************************
CFUNCTION- FOR EACH ELEMENT CALL DSM, ASSEMK, QSENS, AND
C          SPBSL THEN STORE THE JOINT DISPLACEMENT
C          DERIVATIVES W.R.T. ELEMENT DESIGN VARIABLES IN
C          VECTOR (QD).
C
      SUBROUTINE SENS(C1,C2,EMOD,ELENG,NE,DKGA,DKGI,DAS,S,
     $                DIS,Q,NEQ,MXNEQ,SS,MCODE,JCODE,MINC,
     $                NG,DKS,QS,QD,LC,MBD,ITEN)
      DIMENSION C1(*),C2(*),EMOD(*),ELENG(*),DKGA(*),S(*),
     $          SS(MXNEQ,*),MCODE(6,*),JCODE(3,*),MINC(2,*),
     $          QD(MXNEQ,*),DKS(MXNEQ,*),Q(*),QS(*),DKGI(*)
C
      DO 50 I=1, NE
       CALL GDR(C1,C2,EMOD,ELENG,I,DKGA,DKGI,DAS,DIS,S,ITEN)
       CALL ASSEMK(I,NEQ,MCODE,DKGA,DKGI,DAS,DKS,MXNEQ,DIS)
         CALL QSENS(Q,DKS,NEQ,QS,LC,MXNEQ,I)
         CALL SPBSL(SS,MXNEQ,NEQ,MBD,QS)
         PRINT 10
  10     FORMAT(1X,//'  Q D (NEQ,I)'/)
         DO 20 JK=1,NEQ
            QD(JK,I)=0.0
  20     CONTINUE
         DO 40 JE=1, NEQ
            QD(JE,I)=QS(JE)
            PRINT 30, JE,I,QD(JE,I)
  30        FORMAT(1X,T10,'QD(',I2,',',I2,')=',F15.4)
  40     CONTINUE
  50  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C                          SUBROUTINE  GDR                     *
C*************************************************************
C FUNCTION- DIFFERENTIAT THE SYSTEM STIFFNESS MATRIX, k,
C           W.R.T. AREA (DKGA) AND MOMENT OF INERTIA
C           (DKGI).ALSO DIFFERENTIAT DEPENDENT SECTION
C           PROPERTIES EQUATIONS W.R.T. SECTION MODULUS, S
C
      SUBROUTINE GDR(C1,C2,EMOD,ELENG,I,DKGA,DKGI,DAS,DIS,S,
     $                ITEN)
      DIMENSION C1(*),C2(*),EMOD(*),ELENG(*),DKGA(*),S(*),
     $          DKGI(*)
C
      REAL L
      L=ELENG(I)
      C11=C1(I)
      C22=C2(I)
      E=EMOD(I)
      S1=S(I)
      DAS=0.0
      DIS=0.0
      DO 10 JS=1,7
          DKGA(JS)=0.0
          DKGI(JS)=0.0
  10  CONTINUE
C DK/DA
      DKGA(1)=(E*C11**2)/L
      DKGA(2)=(E*C11*C22)/L
      DKGA(3)=(E*C22**2)/L
      DKGA(4)=0
      DKGA(5)=0
      DKGA(6)=0
      DKGA(7)=0
C DK/DI
      DKGI(1)=(12*E*C22**2)/L**3
      DKGI(2)=(-12*E*C11*C22)/L**3
      DKGI(3)=(12*E*C11**2)/L**3
      DKGI(4)=(-6*E*C22)/L**2
      DKGI(5)=(6*E*C11)/L**2
      DKGI(6)=(4*E)/L
      DKGI(7)=(2*E)/L
C DA/DS(DERIVATIVES OF TRAYNORS EQUS.)
      IF(ITEN.EQ.1) THEN
         IF(S1.LE.503) THEN
            DAS=0.0298025*(2*S1+580)/(S1**2+580*S1)**0.5
            DIS=(S1+290)/30.3
         ELSE
            DAS=0.072309
            DIS=18.5111
         END IF
```

```
C DI/DS
      ELSE IF(ITEN.EQ.2) THEN
         DAS=0.119637-(19.41765E-05*S1)+(16.02478E-08*S1**2)
         IF(S1.LT.50.0) THEN
            DIS=7.98
         ELSE
            DIS=10.90819+(0.0368155*S1)-(3.228723E-5*S1**2)
         END IF
      END IF
        PRINT 20, DAS,DIS
  20    FORMAT(1X,/T10,'DAS=',F10.4,T50,'DIS=',F10.4/)
      DO 40 IE=1,7
        PRINT 30,  IE,DKGA(IE),IE,DKGI(IE)
  30    FORMAT(1X,T10,'DKGA(',I2,')=',F12.4,T50,'DKGI(',I2,
     $          ')=',F12.4)
  40  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C                        SUBROUTINE  ASSEMK                    *
C*************************************************************
C FUNCTION- INITIALIZE INDEX, THEN COMPOSE THE DERIVATIVE
C           STIFFNESS MATRIX W.R.T. SECTION MODULUS, S(DKS)
C
      SUBROUTINE ASSEMK(N,NEQ,MCODE,DKGA,DKGI,DAS,DKS,MXNEQ,
     $                  DIS)
      DIMENSION MCODE(6,*),DKGA(*),DKGI(*),DKS(MXNEQ,*),
     $          INDEX(6,6),DKG(7)
      DATA INDEX /1,2,4,-1,-2,4,2,3,5,-2,-3,5,4,5,6,-4,-5,7,
     $        -1,-2,-4,1,2,-4,-2,-3,-5,2,3,-5,4,5,7,-4,-5,6/
C
      DO 10 JG=1,7
         DKG(JG)=0.0
  10  CONTINUE
      DO 30 JE=1, 7
         DKG(JE)=DKGA(JE)*DAS+DKGI(JE)*DIS
C        PRINT 20, JE,DKG(JE)
  20     FORMAT(1X,T10,'DKG(',I2,')=',F18.4)
  30  CONTINUE
      DO 50 II=1, NEQ
         DO 40 JJ=1,NEQ
            DKS(II,JJ)=0
  40     CONTINUE
  50  CONTINUE
      DO 70 JE=1, 6
         J=MCODE(JE,N)
         IF(J.NE.0) THEN
            DO 60 IE=1, 6
               I=MCODE(IE,N)
               IF(I.NE.0) THEN
                  L=INDEX(IE,JE)
                  IF(L.GT.0) THEN
                      DKS(I,J)=DKG(L)
                  ELSE
                      DKS(I,J)=-DKG(-L)
                  END IF
               END IF
  60        CONTINUE
         END IF
  70  CONTINUE
      PRINT *, '   D   K   S   ( N E Q , N E Q )  '
      DO 90 I=1, NEQ
           PRINT 80, (DKS(I,J),J=1,NEQ)
  80       FORMAT(1X,T3,9F12.4/)
  90  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C                        SUBROUTINE  QSENS                    *
C*************************************************************
C FUNCTION- COMPOSE THE LOAD VECTOR [DK/DX].{Q}= {QS} FOR
C           SENSITIVITY ANALYSIS COMPUTATIONS.
C
      SUBROUTINE QSENS(Q,DKS,NEQ,QS,LC,MXNEQ,II)
      DIMENSION Q(*),DKS(MXNEQ,*),QS(*)
C
      DO 10 JU=1,NEQ
         QS(JU)=0.0
  10  CONTINUE
      DO 40 I=1, NEQ
         SV=0.0
         SVE=0.0
         DO 20 J=1, NEQ
            SV=DKS(I,J)*Q(J)
            SVE=SVE+SV
  20     CONTINUE
         QS(I)=-SVE
         PRINT 30, I,QS(I)
  30     FORMAT(1X,T10,'QS(',I2,')=',F12.4)
  40  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C                      SUBROUTINE  SENSD                      *
C*************************************************************
C FUNCTION- COMPUTE THE JOINT DISPLACEMENT DERIVATIVES W.R.T
C           MEMBER DESIGN VARIABLES AND THE DEFLECTION
C           DERIVATIVES W.R.T. MEMBER DESIGN VARIABLE. ALSO,
C           KEEP A RECORD FOR DEFLECTION DERINATIVES THAT
C           CORRESPOND TO THE CONTROLLING LOAD CONDITIONS.
C
      SUBROUTINE SENSD(LC,NE,NG,NEQ,QD,QDG,T,A,DE,MXNEQ,DEM,
     $              EM,NEG,GCODE,IELC,M)
      DIMENSION QD(MXNEQ,*),T(NE,*),A(NG,*),DE(NG,*),EM(*),
     $          DEM(NG,*),GCODE(NG,*),IELC(*),QDG(MXNEQ(*)
C
      DO 30 JJ=1, NG
         DO 20 JEE=1, NEQ
            SM=0.0
            SN=0.0
            DO 10 IR=1, NE
               SN=QD(JEE,IR)*T(IR,JJ)
               SM=SM+SN
   10       CONTINUE
            QDG(JEE,JJ)=SM
   20    CONTINUE
   30 CONTINUE
      DO 60 I=1, NG
         DO 50 JE=1, NG
            DD= 0.0
            EE=0.0
            DO 40 J=1,NEQ
               EE=A(JE,J)*QDG(J,I)
               DD=DD+EE
   40       CONTINUE
            DE(JE,I)=DD
   50    CONTINUE
   60 CONTINUE
      IF(LC.EQ.3) THEN
      PRINT *, ' Q D G (NEQ,NG) '
      PRINT *, '                '
      DO 80 II=1,NEQ
      PRINT 70, (QDG(II,JJ),JJ=1,NG)
   70 FORMAT(1X,T3,9F12.4/)
   80 CONTINUE
      PRINT *, '  D E (NG,NG) '
      PRINT *, '                '
      DO 100  IJ=1,NG
         PRINT  90, (DE(IJ,J),J=1,NG)
   90    FORMAT(1X,T3,9F12.4/)
  100 CONTINUE
      END IF
```

```
      DO 120 JE=1, NG
         IL=IELC(JE)
         IF(IL.EQ.LC) THEN
             DO 130 I=1, NG
                  DEM(JE,I)=DE(JE,I)
120          CONTINUE
         END IF
130   CONTINUE
      IF(LC.EQ.3) THEN
         PRINT 140
140      FORMAT(1X,/' DEM (NG , NG)'/)
         DO 160 JJ=1,NG
            PRINT 150, (DEM(JJ,II),II=1,NG)
150         FORMAT(1X,T3,9F10.6/)
160      CONTINUE
      END IF
      RETURN
      END
```

```
C*************************************************************
C                        SUBROUTINE  SOLVED                    *
C*************************************************************
C FUNCTION- SOLVE FOR THE DESIGN VARIABLE INCREMENT.
C
      SUBROUTINE SOLVED(DEM,EM,DLEM,NG,DLTS,AD)
      DIMENSION DEM(NG,*),EM(*),DLEM(*),DLTS(*),
     $          WKAREA(1000),AD(*)

      PRINT *, '       DLEM(NG)    '
      DO 20 JE=1, NG
            IF(EM(JE).LT.0) THEN
               DLEM(JE)=(-AD(JE)-EM(JE))
            ELSE
               DLEM(JE)=(AD(JE)-(EM(JE)))
            END IF
         PRINT 10, JE,DLEM(JE)
 10      FORMAT(1X,T10,'DLEM(',I2,'=)',F9.4)
 20   CONTINUE
      IDGT=4
      MM=1
      CALL LEQT2F(DEM,MM,NG,NG,DLEM,IDGT,WKAREA,IER)
      PRINT *, ' D L T S ( I ) : '
      DO 30 J=1, NG
         DLTS(J)=DLEM(J)
 30   CONTINUE
      DO 50 J=1, NG
         PRINT 40, J,DLTS(J)
 40      FORMAT(1X,T10,'DLTS(',I2,')=',F18.4)
 50   CONTINUE
      RETURN
      END
```

```
C*************************************************************
C                        SUBROUTINE  CONVRG                    *
C*************************************************************
C FUNCTION- MANAGE THE CONVERGENCE COMPUTATIONS AND CHECKS,
C           IF CONVERGENCE iS NOT ACHIEVED, DEVELOP A NEW
C           TRIAL STRUCTURE; IF CONVERGENCE IS ACHIEVED ,
C           PRINT OUT THE RESULTS OF THE FINAL DESIGNE.
C
      SUBROUTINE CONVRG (RM,AG,ZIG,SG,CP,NG,NTS,M,ILC,DLTS,
     $                   ITEN)
      DIMENSION RM(*),AG(*),ZIG(*),SG(*),ILC(*),DLTS(*)
C
      IC=0
      CALL EVALDE(RM,CP,IC,NG)
      IF(IC .EQ. 0) THEN
         M=M+1
         CALL TRIAL(AG,ZIG,SG,NG,DLTS,ITEN)
      ELSE
         CALL OUTDES(AG,ZIG,SG,RM,M,NG,ILC,CP)
         M= NTS+1
      END IF
      RETURN
      END
```

```
C*************************************************************
C                      SUBROUTINE  EVALDE                     *
C*************************************************************
C FUNCTION- EVALUATE THE CURRENT DEFLECTION STATE AND CHECK
C           CONVERGENCE.
C
      SUBROUTINE EVALDE(RM,CP,IC,NG)
      DIMENSION RM(*)
C
      C=1-CP
      PRINT 10, CP
   10 FORMAT(1X,/' CP(TOLERANCE) = ',F6.3/)
      DRMAX=ABS(RM(1)-C)
      PRINT 15, DRMAX
   15 FORMAT(1X,' DRMAX FOR MEMBER  (1) = ',F20.8/)
      DO 30 I=2, NG
         A=ABS(RM(I)-C)
         PRINT 20,I,A
   20    FORMAT(1X,' A FOR MEMBER (',I2,') = ',F20.8)
         DRMAX=MAX(DRMAX,A)
   30 CONTINUE
      PRINT 40, DRMAX
   40 FORMAT(1X,' DRMAX = ',F20.8)
      IF(DRMAX .LE. CP) THEN
        IC=1
      ELSE
        PRINT *, 'NON CONVERGENCE'
      END IF
      RETURN
      END
```

```
C*************************************************************
C                      SUBROUTINE  TRIAL                      *
C*************************************************************
C FUNCTION- UNDER CONDITIONS OF NON CNVERGENCE, DEVELOP A
C           NEW TRIAL STRUCTURE FOR THE NEXT ITERATION CYCLE
C
      SUBROUTINE TRIAL(AG,ZIG,SG,NG,DLTS,ITEN)
      DIMENSION AG(*),ZIG(*),SG(*),DLTS(*)
C
      DO 30 I=1, NG
            ZZ=SG(I)+DLTS(I)
            SG(I)=ZZ
            ZZ=0.0
            IF(SG(I).LE.0) THEN
               SG(I)=20.0
               PRINT 10, I,SG(I)
  10           FORMAT(1X,T10,'ACTION HAS BEEN TAKEN----SG(',
     $                 I2,')=',F11.2/)
            END IF
            IF(ITEN.EQ.2) THEN
                 IF(SG(I).GT.1110) THEN
               SG(I)=600.0
               PRINT 20, I,SG(I)
   20          FORMAT(1X,T10,'ACTION HAS BEEN TAKEN----SG(',
     $                 I2,')=',F11.2/)
            END IF
          END IF
         AG(I)=PROPA(ITEN,SG(I))
         ZIG(I)=PROPI(ITEN,SG(I))
   30 CONTINUE
       PRINT *, 'I T E N = ',ITEN
      RETURN
      END
```

```
C*************************************************************
C                          FUNCTION  PROPA                     *
C*************************************************************
C FUNCTION- COMPUTE NEW SEC. AREA VALUES (USED TWO SETS)
C
      FUNCTION PROPA(N,X)
C
      IF(N .EQ. 1) THEN
         IF(X.LE.503) THEN
             TAREA=0.464*(((290+X)**2-84100)/60.6)**0.5
         ELSE
             TAREA=(18.5111*X+1988.9336)/256
         END IF
       ELSE IF(N.EQ.2) THEN
C 0<S<1100IN3
       TAREA=3.62415+(0.119637)*X-(9.70882E-5)*X**2
     $       (5.34160E-8)*X**3
       END IF
      PROPA=TAREA
      RETURN
      END
```

```
C*************************************************************
C                          FUNCTION   PROPI                    *
C*************************************************************
C FUNCTION- COMPUTE NEW MOMENT OF INERTIA VALUES.
C
      FUNCTION PROPI (N,SG)
C
      IF(N.EQ.1) THEN
        IF(SG.LE.503) THEN
           TZI=((290+SG)**2-84100)/60.6
        ELSE
           TZI=18.5111*SG-311.0664
        END IF
      ELSE IF(N.EQ.2) THEN
        IF(SG .LT. 50) THEN
           TZI=(7.98)*SG
        ELSE
           TZI=(-191.3096)+(10.90819)*SG+(0.01840775)*SG**2-
     $          (1.076241E-5)*SG**3
        END IF
      END IF
      PROPI=TZI
      RETURN
      END
```

```
C*************************************************************
C                        SBROUTINE  OUTDES                    *
C*************************************************************
C FUNCTION- TO PRINT THE FINAL DESIGN DETAILS AFTER THE
C           PROBLEM HAS CONVERGED.
C
      SUBROUTINE OUTDES(AG,ZIG,SG,RM,M,NG,ILC,CP)
      DIMENSION AG(*),ZIG(*),SG(*),RM(*),ILC(*)
      PRINT 10
  10  FORMAT(1X,//T20,'THE FINAL DESIGN VALUES ARE :'//)
      PRINT 20
  20  FORMAT(1X,T1,'ELE.',T9,'AREA',T19,'ZI',T31,'S',T38,
     $       'RM',T51,'ILC',T57,'M',T70,'CP'/)
      DO 40 I= 1,NG
         PRINT 30,I,AG(I),ZIG(I),SG(I),RM(I),ILC(I),M,CP
  30     FORMAT(1X,T2,I2,T6,F7.2,T14,F10.2,T27,F10.2,T40,
     $           F6.4,T50,I3,t55,I3,T67,F8.5)
  40  CONTINUE
      RETURN
      END
```

```
C*************************************************************
C                        SUBROUTINE  ASSIGN                    *
C*************************************************************
C FUNCTION- ASSIGN MEMBER SECTION PROPERTIES TO EACH ELEMENT
C           IN THE GROUP.
C
      SUBROUTINE ASSIGN(AREA,S,ZI,NG,GCODE,NEG,ZIG,SG,AG)
      DIMENSION AREA(*),S(*),ZI(*),GCODE(NG,*),ZIG(*),SG(*),
     $          AG(*)
C
      DO 20 J=1, NG
         DO 10 IA=1, NEG
            I=GCODE(J,IA)
            IF(I.NE.0) THEN
               AREA(I)=AG(J)
               ZI(I)=ZIG(J)
               S(I)=SG(J)
            END IF
  10     CONTINUE
  20  CONTINUE
      RETURN
      END
```

The vita has been removed from the scanned document