1982

Structural analysis aided by interactive computer graphics

David E. Rodgers
Iowa State University

Structural analysis aided by

Interactive computer graphics

by

David E. Rodgers

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Civil Engineering
Major: Structural Engineering

Approved:

In Charge of Major Work

For the Major Department

For the Graduate College

Iowa State University
Ames, Iowa

1982

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In the near future all engineering will be done with the help of a computer, but current general purpose structural engineering computer programs are not adequate for the future needs of the structural engineer.[14] What is needed is a program that reorganizes the approach to problems and uses the latest computer software technology available. Specifically, the program must be straightforward and easy for the engineer to use. Increased interactiveness and improved computer graphics are the software advancements that will be at the heart of the movement.[1*]

Currently, structural engineers have available to them a wealth of computer programs for structural engineering applications. There are computer programs available to automate almost every phase of their structural engineering work.[1] In this wide variety of programs they are able to model almost any type of structural problem imaginable. The limitless combinations of finite and beam elements, support conditions and load types allow the engineer to simulate virtually any physical problem. After the computer has solved the problem, the engineer can extract from the results a variety of information, from support reactions to stress contours. With these results, the structural engineer interprets the numbers and makes decisions accordingly. It should be emphasized that the computer is merely a tool to perform the mathematical operations and it is the responsibility of the engineer to verify the correctness of all computer input and output.[7]

---

*Superscripts refer to entries in the References.

Review of Current Methods

Let us look at a simple example to see how computers can help the structural engineer (see Figure 1.1). This will be done by comparing the steps to solve a problem by two methods: 1) using moment distribution and 2) using the computer program STRUDL.

Moment Distribution

CASE A (SIDESWAY PREVENTED)

- Obtain the relative stiffness of the members
- Calculate the distribution factors
- Calculate the fixed end moments for each member
- Distribute
- Sum the columns/determine the shears
- Determine the horizontal restraining force

CASE B (SIDESWAY INDUCED)

- Apply fixed end moments for sway
- Distribute
- Sum columns/determine the shears
- Calculate the horizontal force that caused the FEM
- Determine the sidesway contribution factor for Case B
- Calculate member end moments
- Calculate shears
- Calculate support reactions through structural equilibrium

As it can be seen, moment distribution usually tends to be quite a lengthy and tedious process to be done manually, just for the simplest of problems.

Conversely, by the computer program STRUDL the steps would be:

- Organize the structural data
    - node coordinates
    - member incidences
    - support conditions
    - member properties
    - loading
- Input the program into the computer
- Run the program
- Interpret the results

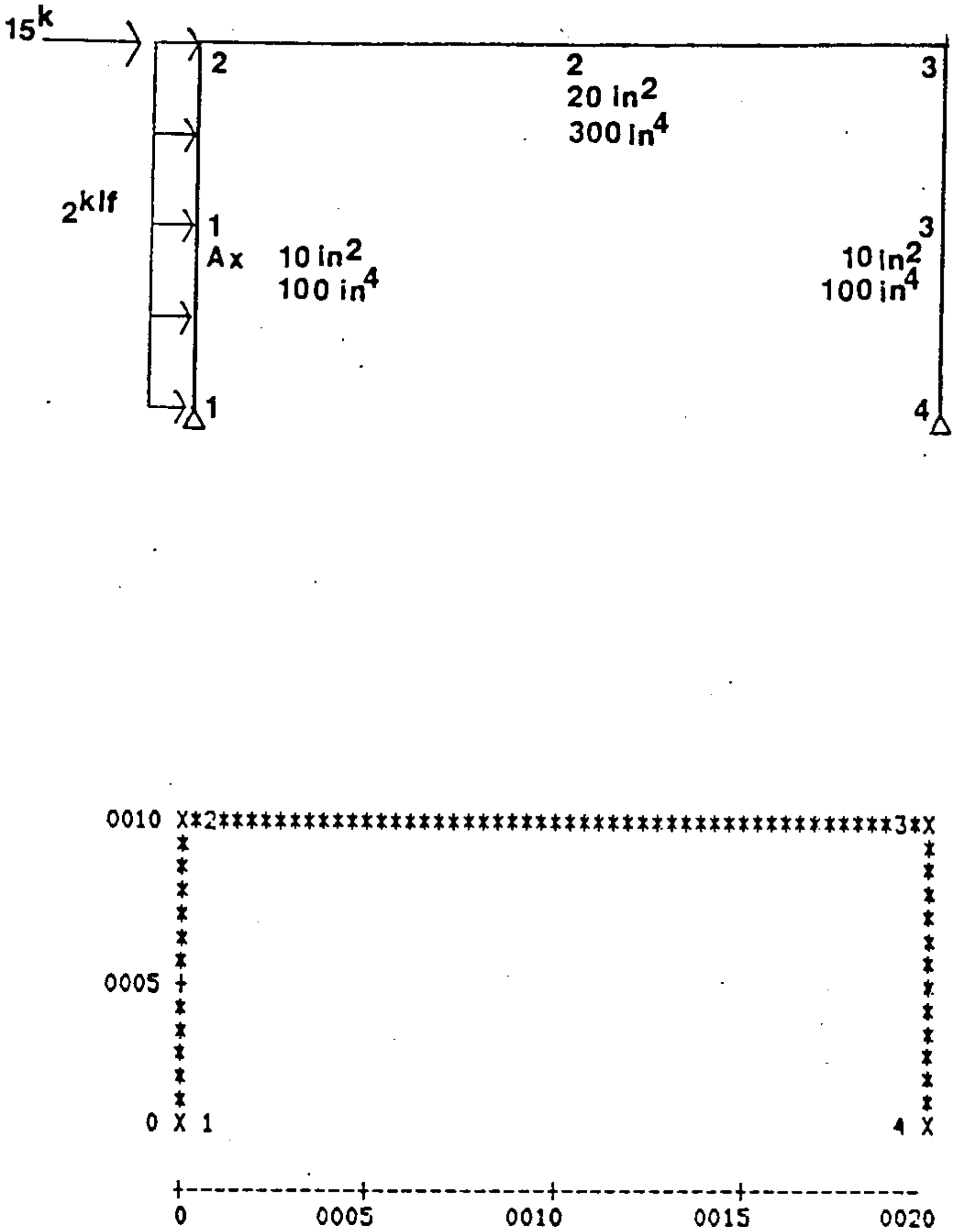


Figure 1.1

Example Problem With STRUDL Frame Representation

Specifically, the STRUDL commands would be:

```
STRUDL 'EXAMPLE -- 3 - MEMBER FRAME'
TYPE PLANE FRAME
UNITS FEET KIPS
JOINT COORDINATES
1    0.0    0.0    S
2    0.0   10.0
3   20.0   10.0
4   20.0    0.0    S
JOINT 1  4 RELEASE MOMENT Z
MEMBER INCIDENCES
1    1      2
2    2      3
3    3      4

UNITS INCH
CONSTANTS E 29000  ALL
MEMBER PROPERTIES
1    3      AX  10.0  IZ  100.0
2           AX  20.0  IZ  300.0
UNITS FEET
LOADING 1  UNIFORM MEMBER LOAD & LATERAL LOAD
MEMBER 2 LOAD FORCE Y UNIFORM W -2.0
JOINT 2 LOAD FORCE X 15.0
LOAD LIST 1
STIFFNESS ANALYSIS
LIST FORCES REACTIONS ALL
UNITS INCH
LIST DISPLACEMENTS ALL
FINISH
```

Table 1 is the computer output from this program. The output includes all forces, reactions, and displacements for all joints and members. Once again, the engineer must carefully inspect all input and output to insure that the correct answers to the correct problem are obtained.

From just this simple example, it is easy to see that the computer can be of great help to the engineer. In fact, large, complex structural problems would be virtually impossible to solve by hand methods, unless many simplifying assumptions are used. The increasing dependence on computers to perform structural analysis is evident by the countless

Table 1
STRUDL Sample Output

ACTIVE UNITS - LENGTH FEET; FORCE KIP; ANGLE RAD; TEMPERATURE DEGF; TIME SEC

ACTIVE UNITS -	LENGTH	FORCE	ANGLE	TEMPERATURE	TIME
	FEET	KIP	RAD	DEGF	SEC

```
**************************************************************************************
*  LOADING - 1            UNIFORM MEMBER LOAD + LATERAL LOAD                           *
**************************************************************************************
```

MEMBER FORCES

MEMBER	JOINT	FORCE: AXIAL	FORCE: SHEAR Y	FORCE: SHEAR Z	MOMENT: TORSIONAL	MOMENT: BENDING Y	MOMENT: BENDING Z
1	1	12.4999905	4.1723003				0.0000000
1	2	-12.4999905	-4.1723003				41.7230072
2	2	10.8276939	12.4999905				-41.7230072
2	3	-10.8276939	27.4999847				-108.2768860
3	4	27.4999847	10.8276939				0.0
3	3	-27.4999847	-10.8276939				108.2768860

RESULTANT JOINT LOADS - SUPPORTS

JOINT		FORCE: X FORCE	FORCE: Y FORCE	FORCE: Z FORCE	MOMENT: X MOMENT	MOMENT: Y MOMENT	MOMENT: Z MOMENT
1	GLOBAL	-4.1723003	12.4999905				0.0000000
4	GLOBAL	-10.8276939	27.4999847				0.0

RESULTANT JOINT DISPLACEMENTS - SUPPORTS

JOINT		DISPLACEMENT: X DISP.	DISPLACEMENT: Y DISP.	DISPLACEMENT: Z DISP.	ROTATION: X ROT.	ROTATION: Y ROT.	ROTATION: Z ROT.
1	GLOBAL	0.0	0.0				-0.0200492
4	GLOBAL	0.0	0.0				-0.0255198

RESULTANT JOINT DISPLACEMENTS - FREE JOINTS

JOINT		FORCE: X FORCE	FORCE: Y FORCE	FORCE: Z FORCE	MOMENT: X MOMENT	MOMENT: Y MOMENT	MOMENT: Z MOMENT
2	GLOBAL	0.1659623	-0.0004310				-0.0096904
3	GLOBAL	0.1655890	-0.0009483				0.0013628

number of structural engineering programs available. These programs are many and varied. Some structural engineering computer programs, such as STRESS and CFRAME, are very specific and limited in their analysis capability. Another class of programs includes broad, general-purpose and high-power analysis programs such as STRUDL, ANSYS and NASTRAN. There are also programs that excel in specific features such as structural design. Examples of special purpose design programs are the PCA concrete design programs and POSTEN. Each of these programs performs well and it would be difficult to improve upon their present features. Therefore, new structural programs should not simply duplicate the capabilities and features that current programs offer. What is needed is a feature that will improve the accessibility of the programs by enhancing the structural input phase. If the man-machine interaction phase of the structural engineering computer programs is improved, the program becomes an even more versatile tool for the engineer to use.

Statement of the Problem

The tasks of the structural analyst are becoming more and more demanding. These pressures are being applied from several sides. There is the need for the engineer to provide analysis in greater depth and more detail, e.g., analysis for offshore and aerospace structures. At the same time, other forces require the engineers to work more effectively and produce structures that are more efficient. Computers, thus far, have become indispensable and irreplaceable in helping engineers keep abreast with the demands placed upon them. Yet as computer technology, (both hardware and software,) advances so must the programs of structural engineering application.[1]

Many factors such as minicomputers, increased interaction and improved graphics, will be at the heart of these new computer programs for structural application. Specifically, it is the man-machine interaction, the input and output phases of the program, that have the most to gain from real time interactive computer graphics. The incorporation of interactive graphics into a structural engineering computer program will open a new dimension in the process used to analyze a structure by computer methods. Interactive computer graphics can be used to automatically generate joints and members. Graphic input is another feature that can help the user enter the structural data into the program. If the interactive conversational dialogue between the user and the program is increased, the program can become still easier and more efficient to use. But simply incorporating these technological advances into our existing programs is not the answer. The opportunity is here to use these advances in order to reorganize the structure of the input and output phases of the structural engineering computer program.

Interactive graphics should be used throughout a structural engineering computer program. First, the geometry of the structure should be entered using graphical input devices. Digitizing tablets, light pens, and graphic cursors can be used to interactively specify joint locations and member incidences. Program generated graphic responses and verification can be used to assure the correct placement of joints, members, support conditions and loads. After the graphical input and solution processes are completed, the graphical post-processing phase should be executed. Graphical representation of the results could aid tremendously in interpreting the structural results. All types of graphic results can be generated by a computer program. Deflected shapes, shear and moment diagrams and stress contours are but a few of the many types of diagrams that a computer can produce.

Interactive computer graphics is the latest advancement in computer technology. Though computer graphics have been around for many years, their use has not been widespread or economical. Only recently have computer graphics become economical and available to the majority of the computer users. Though computer graphics are currently available in some structural programs, never before has a computer program for structural analysis been based on computer graphics from beginning to end. This thesis will briefly discuss the capabilities and features of some current structural analysis computer programs. Then, Chapter 3 proposes a structural analysis computer program with interactive graphic capabilities. The graphic features proposed in this thesis have yet to be implemented in any one computer program. Chapter 4 discusses the general capabilities and graphic features of a computer program that is an actual subset of the program proposed in Chapter 3. This program, written by the author of this thesis, brings together graphic features never before combined in a single structural analysis computer program.

CHAPTER 2

REVIEW OF SEVERAL CURRENT STRUCTURAL ENGINEERING COMPUTER PROGRAMS

In order to look into the future of computer aided structural analysis, a brief look at current and past methods is in order. There are literally hundreds of computer programs for structural engineering applications available today. The following discussion is based on a few programs that are fairly representative of the whole. Structural engineering was one of the first fields in which digital computers were applied.[1] STRESS (Structural Engineering System Solver), developed at Massachusetts Institute of Technology (MIT) in the early 1960s, was very successful and is still widely used today. STRESS provided the engineer with one of the earliest general purpose structural analysis programs. Also, because STRESS is written in a problem oriented language, the user communicates with the program in basic English words, making the need to know computer logic and a programming language unnecessary. As a result of its success, STRESS was used as a springboard for many subsequent structural analysis programs. One of these programs, also developed at MIT, is STRUDL (Structural Design Language). The size, complexity, and capabilities of STRUDL have been expanded by several individual groups; as a result, STRUDL is now the most comprehensive structural analysis and design program available. STRUDL is much more than the analysis tool that STRESS was. Among the added capabilities that STRUDL provided are: finite element analysis, structural design and extended graphic capabilities. Another program is SAP IV, developed at the University of California at Berkeley. SAP IV was preceded by several other SAP versions and is succeeded by newer SAP programs and post-processors. SAP IV is primarily a finite element program and is strictly an analytical tool. One of the newer general purpose programs is ANSYS, developed in the private sector by Swanson Analysis Systems, Inc. ANSYS is an extremely powerful finite element program that has applications beyond just structural engineering. These four programs are currently being used quite extensively throughout the world on small and large computers alike.[4] They represent a spectrum of programs, from the simple beam-element program, to the very high-power general purpose analysis program.

Theory

Almost every structural engineering computer program is based on the stiffness method (also known as the displacement method) of analysis. The matrix formulation of the direct stiffness method is especially suited for computer application.

The basic theory behind the stiffness method of analysis is that each member in a structure has a particular characteristic stiffness. The stiffness of a beam element is dependent upon the E (modulous of elasticity), A (cross-sectional area), I (moment of inertia), L (length) and the end conditions of the member. Stiffness is defined as the force that results due to a unit displacement of a degree of freedom. When several members join to form a frame, this structure has a characteristic stiffness. When the structure is subjected to a particular set of loads, a unique displaced shape results. The global displacements of the structural degree of freedom are the unknown quantities in the stiffness method. See Figure 2.1 for the structural degrees of a 3-D space node. Once the displacements are obtained, all other structural information can be determined.

Mathematically, the matrix analysis of structure is based on the equation:

$$[S] \{D\} = \{A\} \qquad 2.1$$

[S] the overall, assembled stiffness matrix
{D} the set of global displacements of the degrees of freedom
{A} the particular set of global actions at the degrees of freedom

The matrix [S] is an overall structural stiffness matrix that is made up of individual member stiffness matrices. For a plane frame member, the matrix is a 6 x 6 symmetric, positive definite matrix.

$$[S]_{local} = \begin{bmatrix} [S_{ii}] & [S_{ij}] \\ [S_{ji}] & [S_{jj}] \end{bmatrix}$$

The matrix $[S_{ij}]$ is a 3 x 3 matrix. Figure 2.2 illustrates some of the entries in the member stiffness matrix of a plane frame beam element. The member stiffness is originally calculated in the member local coordinate system. Transformation of the member stiffness method into the global coordinate system is accomplished by the equation:

$$[S]_{global} = [R]^t [S]_{local} [R] \qquad 2.2$$

For a two-dimensional plane frame member, the matrix [R] is a 6 x 6 rotation matrix and $[R]^t$ is the notation matrix transposed.

$$[R] = \begin{bmatrix} C & S & 0 & 0 & 0 & 0 \\ -S & C & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & C & S & 0 \\ 0 & 0 & 0 & -S & C & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

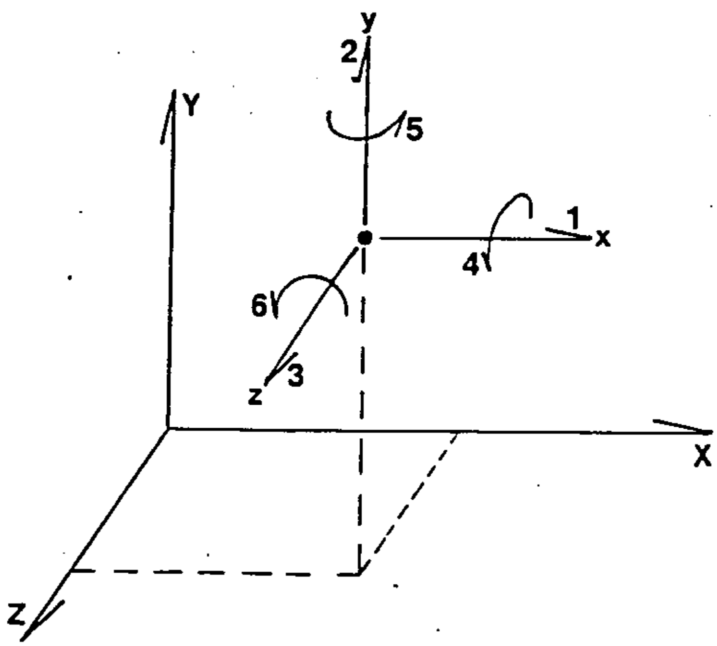


Figure 2.1

Structural Degrees Of Freedom Of A 3-D Node

The entries S and C are the sine and cosine of the member, respectively. (See Figure 2.2.)

To obtain the overall global stiffness matrix, each member stiffness matrix is added to the overall matrix according to the member joint numbers. This overall global stiffness matrix is also symmetric and positive definite.

Structural loads enter into the mathematical model by the global action vector $\{A\}$. This vector has an entry for each degree of freedom in the structure. Loads applied to the members are transformed into equivalent loads occurring at the joints. The resultant joint loads that act on the structure make up the action vector $\{A\}$.

With the overall global stiffness matrix [S] and the global action vector $\{A\}$ determined, Equation 2.1 can be solved for the vector of unknown displacements $\{D\}$. There are many different methods to solve for the unknown vector in Equation 2.1. Probably the simplest and most straightforward method is Gauss Elimination with backward substitution. The stiffness matrix of a structure, modified for support conditions, is always nonsingular; therefore there is always one and only one solution to Equation 2.1.

Program Capabilities

The specific capabilities of just these few programs are too numerous and diverse to discuss in detail but the general capabilities are similar and are discussed below.

Many different types of structural problems can be solved using these programs. Simple, linear small displacement static problems can be solved

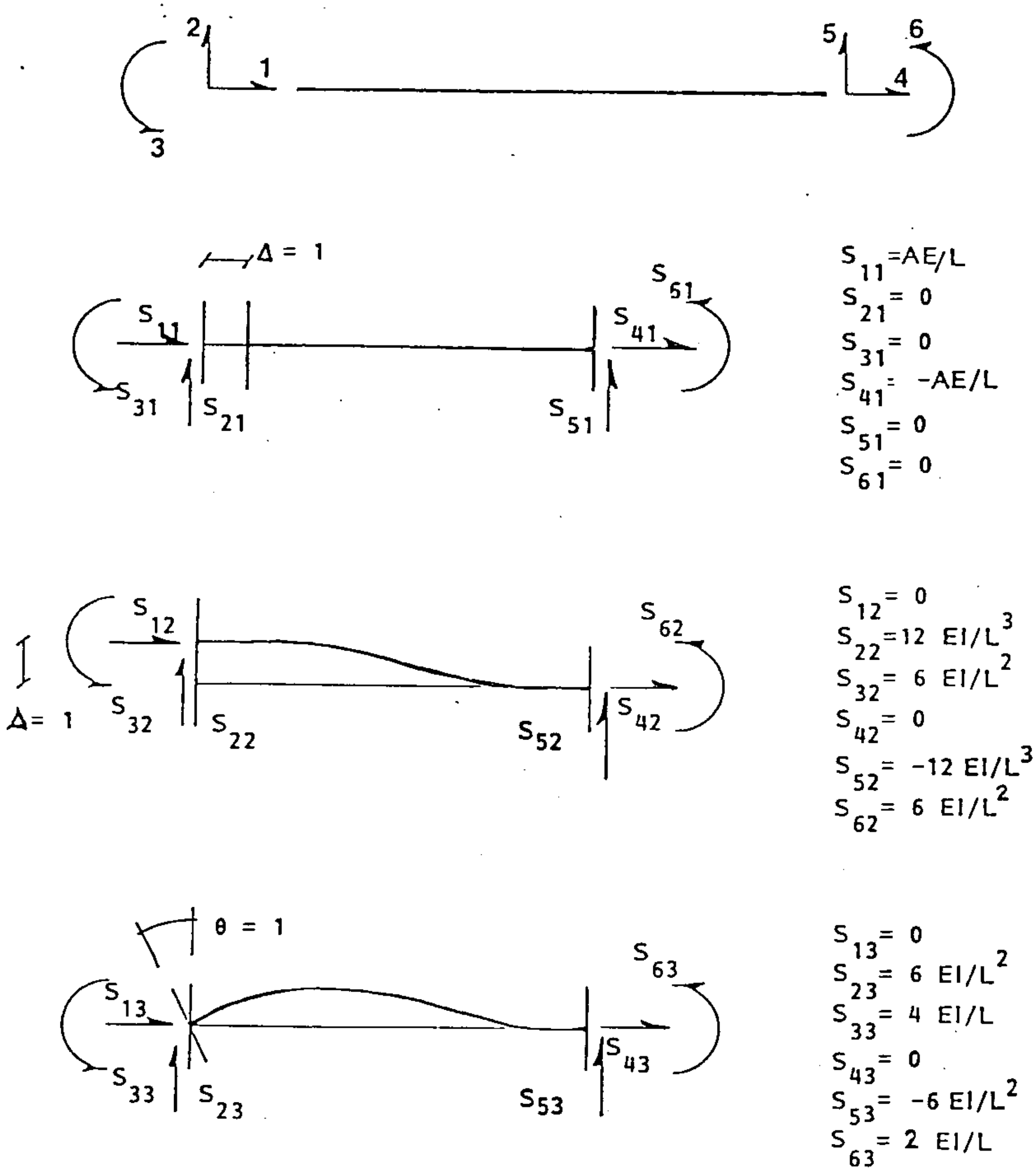


Figure 2.2

Examples Of Member Stiffness

on all of these systems; in fact, that is all that the early STRESS program could handle. Other types of problems that subsequent programs could solve are dynamics, spectral response, large deflection and nonlinear response. Also, the finite element method with its variety of element shapes is able to solve problems of other than structural engineering applications, like heat flow, seepage, and piping.

Just as important as the type of structure, is the structural parameters that the program can accept. The basic structural parameters required to model a structure are the member properties, loads, and support conditions. In structural analysis, the structure is subdivided into discrete members or elements and has idealized loads placed upon it. The movement of the nodes is governed by the degrees of freedom associated with the structure. Structural supports are modeled by constraining specific nodes against specific movements. With this information, the program is able to create a set of matrix equations that mathematically describe the structure. The simplest structural component in each of these programs is the one-dimensional truss element. STRUDL, SAP IV, and ANSYS also allow two-and three-dimensional finite element shapes. These programs permit the mixing of both general finite and beam elements in the same structure; but care must be taken to match structural degrees of freedom at nodes that join several element types.

Many different types of loading conditions are available in these programs. Loads applied to the joints are the only types of loads allowed in the early versions of SAP. The other three programs allow loads to be applied to the members as well as the joints. Concentrated loads and moments and uniform and linear varying loads can be placed on members in order to model actual loading conditions. Some versions of STRUDL allow a uniform load over a two-dimensional element. These force loadings can be applied with respect to the local or global coordinate systems. Also, sloping members and sloping loads can be handled by these programs. In addition to the previously mentioned loading types, all four programs allow thermal loading of members and specified joint displacements in order to create forces in a structure.

Individual loads are grouped to form a load case. Load combinations are created by adding independent load cases. A loading combination is a group of load cases that simultaneously act on the structure. For example:

```
LOAD 1     DEAD LOAD
LOAD 2     LIVE LOAD
LOAD 3     WIND LOAD FROM NORTH
LOAD 4     WIND LOAD FROM SOUTH
LOAD 5     SUPPORT SETTLEMENT OF JOINT 2
```

A load combination is created by combining several of these individual load cases. For example:

```
LOAD 6     'ULTIMATE LOAD'
COMBINE 6  1  1.4  2  1.7
```

These STRUDL statements direct the program to create Load Case 6 consisting of 1.4 times the result of Case 1 plus 1.7 times Load Case 2.

These programs model structural supports as specified joints that have certain degrees of freedom restrained from movement. Care must be taken to insure that the computer model accurately represents the physical problem. All four programs allow a fully and partially restrained joint as a support. Some programs allow a support to act on an inclined surface (one not parallel to one of the global axes). Also, elastic supports and joints are allowed by some programs. In addition, the ends of members can be released from transmitting forces and/or moments. The user must be sure that an unstable structure does not exist due to too many joint and member releases. With the endless combinations of elements, load types and support conditions, almost any basic structural problem can be modeled on the computer.

These structural parameters: element type, load condition, member releases, and support conditions, determine the type of problem that a computer program can solve. The entering of these parameters into the program can be done by one of two methods - batch or interactive. Batch is defined as typing into a computer file (or a deck of punched cards) all the steps necessary to obtain the problem solution. This total file is then submitted to the computer for uninterrupted input, processing, solution and output. The interactive process is the process of entering one command directly into the computer program and letting the program execute this statement before the user enters the next command. This process of directly interacting with the computer is repeated until the problem solution is obtained and the results output. In earlier years, all programs were batch run; but currently, a few programs have been developed to take advantage of the interactive process.

Program Features

In any program, it is the man-machine interaction, the input/output phase of the problem, which is the most critical. A poorly implemented input phase in a program can cause difficulty to the user. The more user-friendly and verification-oriented the program, the greater a tool it becomes for the engineer.

When using either the batch or interactive process, the user must communicate with the program in a language that it can understand. Thus, the user is required to learn the commands and language conventions of the program. To become proficient with any computer program usually requires many hours of practice. Some programs, said to be user-friendly, are easier to learn than others. These programs are always free format and allow the sequence of commands to be input in any order. Another user-friendly feature provided in some programs is default values for certain structural parameters. This feature eliminates the need for the user to issue the commands that individually set each of these parameter values. For instance, in STRUDL, the command "MATERIAL CONCRETE" will initialize an entire table of standard values. To set each of these values would otherwise require one statement each. See Table 2 for a list of these parameters. Some programs also allow the user to change the units of the input parameters, for instance, from feet to inches. Another feature, available in some programs, is the CHANGE/ DELETE command. With this feature, the user is able to correct structural data within the program itself and not have to start the program over.

Some batch and interactive programs provide graphical verification of the structural input. All programs, except SAP IV, can give a graphical representation of the structural configuration. These representations are usually very crude pictures produced on a line printer. (See STRUDL

Table 2
STRUDL Concrete Parameters

CONSTANT	Assumed Value	Explanation	ACI 318-63
FCP	4000 psi	Compressive strength of concrete, f'_c	301
FY	60000 psi	Yield strength of reinforcement, f_y	301
WC	145 pcf	Unit weight of plain concrete	1102a
DENSITY	150 pcf	Unit weight of reinforced concrete[1]	
FC	0.45(FCP)	Allow. compr. stress in concrete, f_c	1002a
VU	$10\sqrt{FCP}$	Ult. shear stress in beam with web reinf.[2]	1705b
V	$5\sqrt{FCP}$	Allow. shear stress in beam with web reinf.	1205b
RFSP	6.67	Splitting ratio, F_{sp}.[3]	505
FYST	1.0(FY)	Yield strength of stirrups	1703
FYSP	1.0(FY)	Yield strength of spiral	913b
FS	0.4(FY)	Allow. tens. stress in primary reinf.	1003a
FSC	0.4(FY)	Allow. comp. stress in column reinf.[4]	1003b
FV	0.4(FY)	Allow. tens. stress in stirrups	1203
PHIFL	0.90	Flexure } Capacity reduction factor	1504
PHISH	0.85	Shear } Capacity reduction factor	1504
PHIBO	0.85	Bond } Capacity reduction factor	1504
PHISP	0.75	Spiral column } Capacity reduction factor	1504
PHITI	0.70	Tied column } Capacity reduction factor	1504
BLFR	0.75	Ratio of max p, (p-p') or (p_w-p_f) to p_{bal}	Chap. 16
PMAXCO	0.08	Max. allow. reinf. ratio in columns	913a
PMINCO	0.01	Min. allow. reinf. ratio in columns	913a
PMINFL	200/FY	Min. allow. reinf. ratio in flexural members	911a
DEFC	0.18	A deflection control coefficient	1507
ES	29×10^6 psi	Modulus of elasticity for reinf. steel	
EC	$33(WC)^{1.5}\sqrt{FCP}$	Modulus of elasticity for concrete	1102a
EU	0.003	Ult. strain in concrete at extreme comp. fibre	1503c

Notes:

1. The constant 'DENSITY' is the STRUDL constant of the same name which has been set to a value of 150 pcf for reinforced concrete. The constant is not used in the current version.
2. VU is multiplied by PHISH internally.
3. Values for v_c are calculated using Eq. 12-9 to 12-11 and 17-8 to 17-10 of ACI 318-63. The assumed value, F_{sp}=6.67 makes these equivalent to the equations in Sections 1201 and 1701.
4. The assumed value of FSC is also limited to 30,000 psi. maximum.

output in Figure 1.1.) Only ANSYS and STRUDL offer interactive graphics. Furthermore, ANSYS offers representation of the structural support conditions and loading data. Since structural engineers are dealing with real, physical problems, the advantage of graphical verification of the sometimes complex structures is essential to a quick and accurate solution. In the output mode, after the solution routine, graphics can once again be a great help to the engineer. Interpreting pages of numerical results is a tedious and error-prone task. Graphical output of the deformed shape of the structure and member performance plots (shear and moment diagrams) can be invaluable in helping the engineer.[15]

After the structural information has been entered into the computer program and the matrix equations solved, information about the response of the structure to the loads can be obtained. All programs output the same basic information, i.e. the reactions at all supports, the forces acting at each joint of each member, and the displacements of all joints. Some programs can give additional information including the forces, displacements and stresses at specified sections within a member. All of these programs require the user to request all output. In this manner, the engineer can selectively request only that information that is necessary. The STRUDL command,

LIST SECTION FORCE MEMBER 1 2 SECTION FRACTIONAL DS 0.0 0.1

requests the forces and moments at 1/10 points of members 1 and 2 to be printed. Another option the user may have is whether the format of the printed material will be grouped according to loading case or members. Output features, though not critical to the performance of the computer program, are an important aspect of the program. A poorly designed output section can make the task of checking and interpreting results tedious and difficult.

Another capability, which is really a post-processing phase, is the structural design phase. Only available on a handful of structural engineering computer programs, the structural design phase can help the engineer to achieve a more economical solution. Most programs are strictly an analysis-only tool. In order for a program to design structural members, a structural design code or specification must be incorporated into the program itself. Specifications such as the American Institute of Steel Construction Specification for Structural Steel Buildings and the American Concrete Institute Building Code for Reinforced Concrete are available in some versions of STRUDL. This feature allows the program to choose an appropriate structural member based on the analysis done previously in the program. This feature - automated structural design - can save the engineer a lot of time, but at the same time can produce additional difficulties. When using automated structural design, the engineer must examine not only the forces in each structural member but the adequacy and appropriateness of each member designed. Many a problem may be created by merely accepting the computer output as correct.[11] Within STRUDL, all structural steel rolled shapes and pipes are available from which the program can select members. In concrete design, the engineer must be more precise in specifying the design parameters. Many types of concrete members are available to choose from, including rectangular beams, Tee and Ell beams, one- and two-way slabs and columns of all shapes and reinforcement patterns.

Summary

As it can be seen, within the structural programs currently available to the engineer, the range of capabilities is wide. Within this range, almost every aspect of structural analysis work has been automated. An examination of the capabilities of just a few programs: STRESS, STRUDL, SAP IV and ANSYS, has shown that almost any type of structural problem can be solved by computer-aided methods. A wide variety from simple, linear static problems to large, finite element dynamics problems can be handled. The specific type of problem that can be solved by a particular computer program is dependent on the type of structural parameters that the program can accept. Element types, loading conditions and joint conditions are the basic structural parameters. The number of ways in which these parameters can be combined dictates the limits of the program. Other features that can determine the success of a program are simplified input and user-friendliness. The type of output available and the format in which it can be presented is also important. These features affect the process that the engineer must go through in order to interpret the computer results. One of the most important features a structural engineering computer program can provide is the interactive graphic process that allows the user to interact directly with the program. A simple and straightforward input phase is essential for the quick and correct entering of the structural parameters. This one feature, interactive graphics can make a structural engineering computer program a more useful tool for the structural engineer.

CHAPTER 3

PROPOSAL FOR AN INTERACTIVE GRAPHIC STRUCTURAL ENGINEERING COMPUTER PROGRAM

As a result of the recent advances in computer technology, computer programs of structural engineering applications should be rewritten to keep abreast. Advances such as increased computational speeds, graphics and improved interactive capabilities will revolutionize the way an engineer can solve a problem by computer-aided methods.[1] Following is a proposal for a structural engineering computer program that not only takes advantage of these recently developed capabilities, but also reorganizes the approach that an engineer takes to evaluate a structure by computer-aided methods. This proposal for a structural engineering computer program will limit itself to only those capabilities needed to analyze a two-dimensional plane frame small displacement linear static structural problem. The principles presented can easily be extended to handle larger, three-dimensional problems of dynamic, large displacement and nonlinear response. Also, finite element and structural design capabilities could be added to the program to produce a comprehensive computer package.

The Interactive Feature

From the beginning, the program should be completely interactive. Interactive means that the user and the computer should converse throughout the input and output process. Currently, most programs available are completely batch and require pre-programming in order to model a structure. ANSYS and some versions of STRUDL currently offer some interactive capabilities, but more capabilities are needed than the limited ones that either of these programs offers. The program should be of conversational capability. It need not necessarily contain word and sentence dialogue, but must at least contain prompts and responses. Also, "help" sections should be provided to guide the user through the program and give information about the commands available.[11] Interactive conversation should go further than just post-verification of the structural data. The user must be able to base his next command as a result of the computer's immediate response. This includes the listing and verification of any piece of structural data and the ability to change or delete any of this datum. The solution phase cannot and should not be interactive[11] but the output and post-processing parts of the program must be. It would be a waste of the user's time and computer time to run a program in batch, obtain the printed results and discover a minor error which renders the entire solution worthless. A frequent error in batch run programs is the misspelling of a command or the mistyping of a value. Either of these types of errors could void an entire computer run. To help eliminate this problem, the computer program should be more user-friendly and interactive in its input and output phases.

The Graphics Feature

Graphics is another area in which the new structural computer program should excel.[1,11,12] Since the engineer is modeling an actual physical structure, graphics are essential to verify the sometimes complex nature of the structure. Pictures of the structure can confirm the connectivity of the members, conditions of the supports, locations of the loads and just about every other structural parameter. Lines, arrows and symbols can be used to represent the parameters directly on a graphic display terminal. (See Figure 3.1.) As a result of the display, the user may elect to make changes in the structural data. In the post-processing and output phase of the program, shear and moment diagrams can be produced to explain and clarify printed numerical data. Not only single line diagrams, but envelope diagrams of several load cases could be superimposed on the same display. This gives the engineer the ability to quickly interpret the output data. Another diagram that can illustrate the behavior of the structure is the shape of the deformed structure. Nodes of extreme displacement and points of unusual deformation can be identified immediately and investigated more thoroughly. Graphics can clarify and emphasize many interesting points that might not be detected had printed data been the only form of output.

These two features, interactiveness and graphics, will place a structural computer program in the forefront of computer software technology. A new structural engineering computer program should reorganize the approach used to enter the structural data into the program. Current structural programs require the user to create nodes in space and specifies members to span between nodes. Special attention must be given to node and member numbering and the local and global coordinate systems. This new program takes a different approach. A structure is made up of members which are attached to each other at locations called nodes. Perhaps a subtle difference, but in this approach the user need not be concerned directly with node locations and numbers.

This approach depends on the recent computer technology advances of interactive graphics in order to create and draw the structure in real time. As the structure is being created, joints and members are automatically numbered. This feature frees the user from the task of calculating joint coordinates and laying out the connectivity of the structure beforehand. Even in two of the most advanced programs currently available, STRUDL and ANSYS, graphics is only a tool to verify the connectivity of the structure after it has been entered. This new structural computer program should be able to graphically display the structure as it is created. Furthermore, this program should graphically display the location of each joint and member load. Support conditions and member end releases should also be represented on the graphic terminal screen. This graphic verification of the structural data is actually producing line diagrams similar to the sketches that engineers draw to help them understand the actual conditions on the physical structure. (See Figure 3.1.) Also, a sketch of the deformed structure can help the engineer to better understand the behavior of the structure. (See Figure 3.2.) The computer can use the solution results to draw the deformed structure. A wealth of information can be gained from examining member shear and moment diagrams and envelopes. The program should be able to produce shear and moment diagrams in hard copy form accurate enough to scale-off values. This could eliminate pages upon pages of printed computer output. Increased interaction with the program can only improve the process which the engineer must go through in order to solve a structural problem. The concept behind interactive graphics and

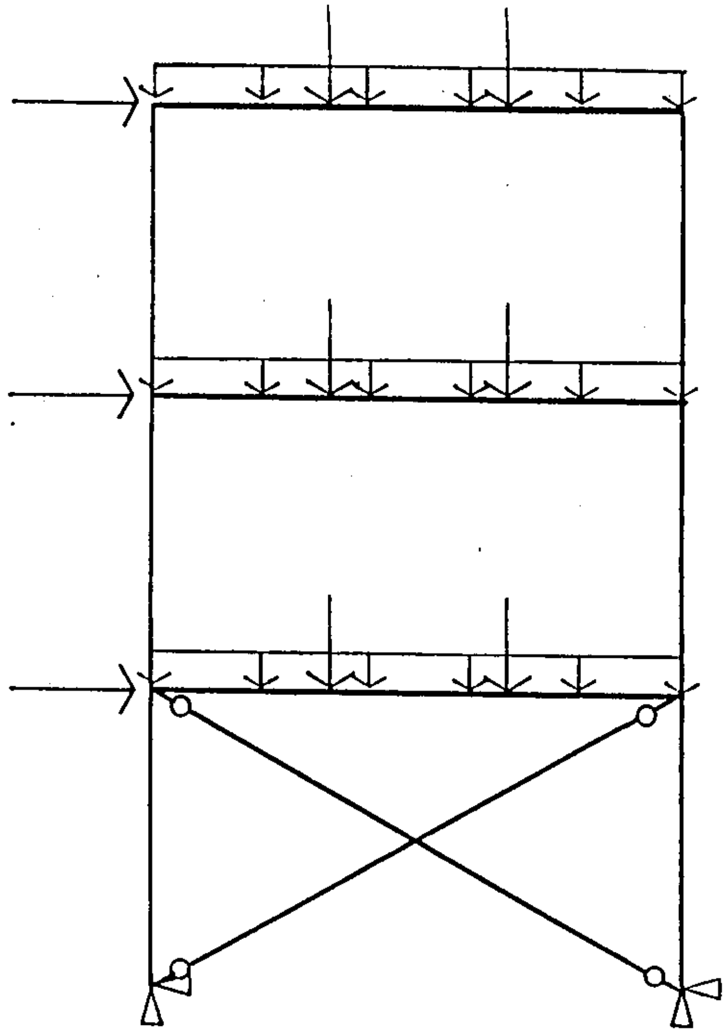


Figure 3.1
Computer Representation Of A Structural Frame

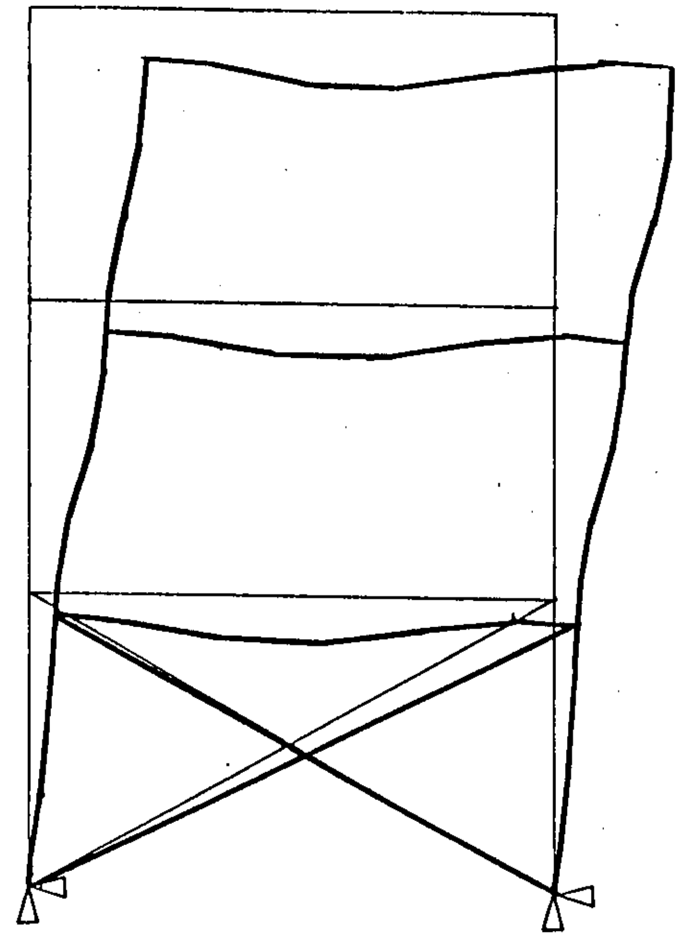


Figure 3.2
Computer Representation, Of A Deformed Structure

user-friendly dialogue is to correctly analyze the correct structure the first time.

The User-Friendly Feature

A user friendly program should communicate with the user in a simple, high-level language. There should not be too many commands and the spelling of a command should relate directly to the function of the command. For example, the word "redraw" could command the program to execute the routine that would clear the screen and redisplay on the screen the structure with all the latest input and changes incorporated. The logic of a user-friendly program should be easy to follow. The structure of the program should be such that any portion of the program is always accessible. The exception is that the program should prevent the user from entering routines that he/she should not be in. For example, the user should not be allowed inside a routine to place loads on members if there are no members currently active. In other words, the computer program should be "idiot-proof".

Other user-friendly features the program should provide are generation and duplication routines. The user should be able to create and duplicate data in an easy fashion. The beginning-end-increment feature is ideal for this purpose. For instance, consider a structure in which all the beams in a particular level are even-numbered from 2 to 20. In order to give all the beams the same member of properties, the user would only need to give a command similar to:

MEMBERS 2 TO 20 BY 2 PROPERTIES AX 10.0 IZ 100.0.

The user should also be able to create the structure in a minimum of steps. A command, "BAYS", that will create a whole line of structural bays identical to one given could drastically reduce the time required to input the structural geometry. This is what user-friendly programs are all about, having simple, easy, unambiguous commands perform a lot of work with a minimum of user input.

In addition to graphical displays, the program should list in printed form, on-line, all information concerning the structure. Naturally, some information is not suitable for graphic representation. It would be more appropriate for parameter values such as the member constant E (modulus of elasticity) and properties I (moment of inertia) and A (member cross-sectional area) to appear only in list fashion in printed form. Another list of printed information that should be available to the user is an on-line "help" section. These "help" sections will not replace the User's Manual but will be an on-line reference resource about the program. It should be noted that these "help" sections are not meant to be a full-fledged documentation of the program, though in some instances, short texts concerning certain commands would be appropriate. These sections should be provided to merely refresh the user's memory to the commands available from his/her particular location in the program. These interactive and graphic features will produce a better structural analysis computer program.

Auto-explanatory prompts that lead the user through the program are essential for better man-machine interaction. These prompts should instruct the user what parameter values to input into the program. Also, error-messages are needed to flag incorrect and ambiguous data. One of the purposes of an interactive program is to immediately communicate to the user which commands the program does and does not understand. Messages should also be in the solution phase during the validity check of the structure. This check is to guard against a fatal error when solving solving the matrix equations. Errors such as missing member properties or unstable joints should be called to the user's attention for immediate correction. To solve a structural problem requires communicating with the program. It is the user-friendly feature that facilitates this man-machine interaction.

Program Capabilities

The structural modeling capabilities of this program should be comparable to most other structural programs. Limiting the discussion to the one-dimensional beam element, the program should be able to accept concentrated loads and moments as well as linear and uniformly distributed loads. Other methods to induce forces in a structure should also be provided. Thermal loadings, both the uniform temperature and the temperature gradient loading of members must be included. Also, routines to handle specified joint displacements and misfitting members must be provided in this computer program. Support modeling must also be able to simulate what could happen in a real structure. All program provide the fully fixed and partially restrained joints as a support. The linear elastic support and the support on an inclined surface, though not used frequently, are needed in order to correctly model structures with these conditions. Joint and member end releases are also needed. Member end releases for shear and slotted connections have their place in real structures and should thus be included in a structural modeling program.

With these capabilities, the program should be able to simulate just about any condition possible in a simple, real, static structure.

Summary

This proposal for an interactive graphic structural engineering computer program should use the latest in computer technology. This will provide the engineer with a program that is state-of-the-art. In addition, these technological advances can be used to reorganize the approach used to solve a structural problem. The key is real time interactive computer graphics. The structure is built with members that connect with each other at joints. Each joint and member is automatically given a number by the program as it is created. Almost every command in the structural input phase should have an immediate automatic graphic response. The types of problems this program should be able to solve should be comparable to most current structural computer programs. It is the completely interactive graphic input and output that will make this program special. With this feature, the engineer can quickly input the problem and more efficiently interpret the results.

CHAPTER 4

A SUBSET PROGRAM OF THE PROPOSED INTERACTIVE GRAPHIC STRUCTURAL PROGRAM

This chapter describes a computer program, written by the author of this thesis, that is a direct subset of the interactive graphic program previously proposed. The specific capabilities of this program are discussed below. A user's manual and example problems are provided in the Appendix.

Capabilities

This program is able to solve two-dimensional plane frame static linear problems and is not limited to orthogonal members. The standard three degrees of freedom, translation X, translation Y and rotation Z, are given to every joint. (See Figure 4.1.) One-dimensional beam elements of constant cross-section can be used to model a structure with both joint and member loads. The joint loads can be concentrated forces in the global X or Y direction and moments about the global Z axis. Member loads can be either concentrated or distributed. Point loads can be applied anywhere along the member but must be directed along one of the member's local axes. Distributed loads must be uniform but can be applied over the whole member as well as over just a portion of the member. Both the local X and local Y directions are available for distributed loads. Supports can be modeled by restraining any combination of the three degrees of freedom for any joint. (See Figure 4.2.) These restraints must act in the principal global directions. The ends of any individual member may be

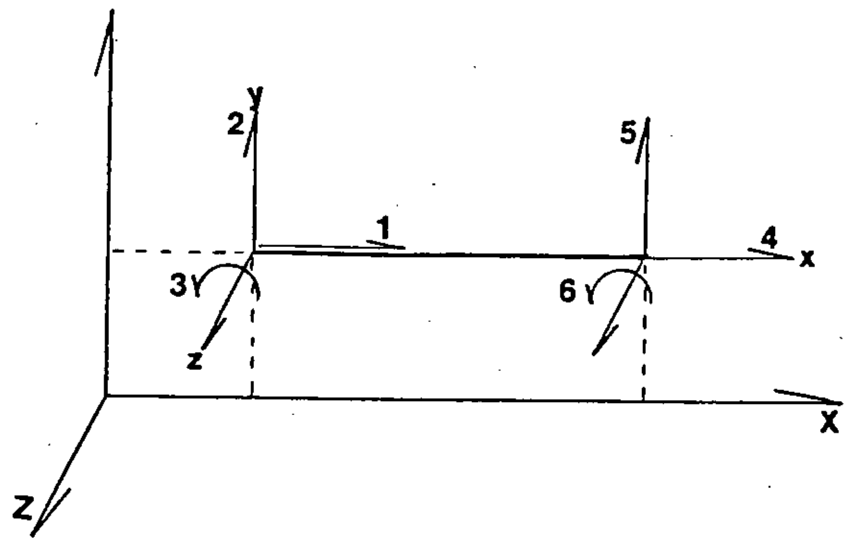


Figure 4.1

Degrees Of Freedom For A 2-d Beam Element

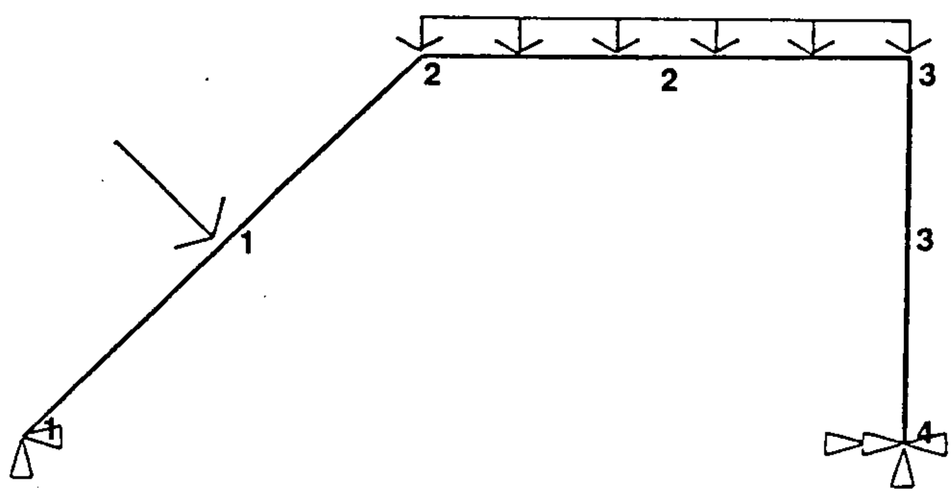


Figure 4.2

Computer Representation Of A Simple Frame

released from transmitting moment. With just these few limited capabilities and some innovation from the user, practically any two-dimensional plane frame structure can be modeled with this program.

User-Friendly Features

Conversation with the program is facilitated by the interactive user-friendly features. Most of the user-friendly features mentioned in the previous chapter are included in this program. Dialogue with the program is in the form of commands issued by the user and prompts and messages issued by the program. In order to minimize repetitive user input, a "copy" command and a "begin-end-increment" command are provided. Also, a validity check of the structural model is made prior to formulation of the matrix equations. Not only are the commands user-friendly but the structure of the whole program is meant to be user-friendly. There is no rigid input sequence, input is free format and the approach to the problem is straightforward and practical.

Structural Input

The innovations in this program rely on its interactive graphic features. The features, many of which were discussed previously, will simplify the input phase and clarify the structural output. Since entering the correct data into the computer program is essential for a correct run, this program was created to be very user-friendly. A variety of methods is available in which to build the structure member by member. In each method, each step is verified with an immediate graphic response on the terminal screen. One method, of course, is to specify joint coordinates and member incidences. Also, the program is able to duplicate a specified structural bay or story. Another input feature is the ability to input the structure via a digitizing tablet. Once again, this program uses current computer technology in order to simplify the user's input task.

The input of other structural data is also simplified. Member end releases, constants, properties and supports can be specified in the conventional manner. The user may repeat these structural data by copying the conditions or values from one joint or member to many other joints or members. With the use of the begin-end-increment command, the user has great control over the copying process. Loading conditions can be specified by accessing the load subroutine. By answering the auto-explanatory prompts, loads can be placed on members and joints. Verification of correct placement of the loads is quickly and easily obtained by the graphic display. As with most other structural programs, several different loading cases can be specified. Each load case is completely independent. Because the program assumes a linear elastic structure, the load cases can be factored, superimposed and added to each other to create additional dependent load combinations. These interactive and graphic features make it easy to enter the structural data and quickly verify its correctness.

In addition to the graphic verification, the program provides routines to list in text form any or all information concerning the structure. Information can also be changed or deleted at any time. Using the change and delete sections, any piece of data can be altered or erased.

Output Features

After the structural data is entered, the solution phase is invoked. The solution results may now be reviewed. In the post-processing phase, the engineer can use interactive graphics to examine both the original and deformed structure. Also, the structural supports and loads can be displayed in order to help interpret and understand the structure's response. Individual shear and moment diagrams and envelopes can be produced with accuracy that makes scaling acceptable. Of course, printed structural results can be displayed on-line or sent to disk storage for printing later. Also, a "save" command is available in order to store the structural data on disk for later recall into the program.

Summary

All users, both the engineer and the nontechnical user, will benefit from this program. Because the format of the program is interactive and conversational, no computer programming knowledge is required. In order to use this program, the user must learn the commands and standard conventions used by the program. Also, the program eliminates the need for the user to be a proficient typist. Input prompts and graphic input responses help to keep typing to a minimum. These features, along with the others mentioned previously help to make this program a simple, easy to use tool for all users.

All in all, this program is a direct subset of the previously proposed interactive graphic program. All of the basic capabilities needed to solve a simple two-dimensional linear static problem are provided. The program is actually only a springboard for a larger, more comprehensive program. Many of the routines could have been handled in a more optimum way. The command handling routines especially, could use the expertise of a computer scientist. But the program does serve its purpose: to show the structural engineering community that it is time to rewrite current structural engineering computer programs to take greater advantage of the latest computer technology. In doing so, we can create interactive graphic structural computer programs that are far more user-friendly and efficient. These programs of interactive graphic capabilities, will play an increasingly important role in the engineer's computing power and productivity.

CHAPTER 5

CONCLUSION

This thesis has presented an overview of the general capabilities of a few current structural engineering computer programs. Generally, these programs can be used to model a wide range of structural problems. A thorough discussion of how these programs can be applied to specific structural engineering problems is beyond the scope of this thesis. The specific capabilities within a program determine what type of problems a particular program can solve.

Chapter 2 discussed the basic capabilities and features of four popular structural engineering computer programs: STRESS, SAP IV, STRUDL and ANSYS. All classes of problems, (static, dynamic, finite element, linear and nonlinear,) can be solved using these programs. The basic parameters such as element types, loading conditions, and support conditions determine the type of problem each program can handle. Program features such as user-friendliness and interactiveness, during the input/output phase, determine how easy a problem solution can be obtained. Chapter 2 also mentioned one special program feature - graphics. It was pointed out that most current engineering computer programs utilize graphics very little and that interactive graphic input is virtually nonexistent.

Therefore, Chapter 3 proposed a structural engineering computer program that would be completely based on interactive computer graphics. The recent advancements in computer technology and the growing availability of computer graphics now make it feasible for programs to be based on interactive graphics. The class of problem that this program can solve is limited to a 2-dimensional, static, plane frame problem; though the basic idea can be applied to programs that can handle 3-dimensional, finite element, dynamic and nonlinear problems. The method used to enter the structural parameters was also reorganized. Interactive graphic techniques would be used to enter the structural geometry. This means that every command would generate an automatic graphic response. With this method of input, the structural data are graphically verified as they are created. Using this approach, it will be virtually impossible for incorrect structural data, such as node location, member incidences or support conditions to slip by undetected. Interactive graphics would also be used to enhance the output of the structural results. Member performance plots and the shape of the deformed structure can be calculated from the problem solution.

Chapter 4 presented the general capabilities of a program that is an actual subset of the interactive graphic program proposed in Chapter 3. This program, written by the author of this thesis, can solve 2-dimensional, static, plane frame structures. Most of the interactive graphic features proposed in Chapter 3 are included in this program. Also included are many user-friendly features such as member generation routines and a duplicate command.

The appendices are devoted to a detailed description of the program presented in Chapter 4. Appendix A is a user's manual. Covered in this appendix are the program conventions, structural conventions and an explanation of each command. Appendix B discusses the actual FORTRAN code of the program. The function and execution of each subroutine is discussed here. A logic flow diagram and selected flow charts are presented in Appendix C. Appendix D contains two example problems. The interactive man-machine dialogue used to set up the problems and actual computer generated graphics are included in this appendix. The reader is encouraged to obtain hands-on experience with the program in order to appreciate the true interactive nature of the program. Appendix E closes this thesis with the actual FORTRAN code.

As computers become more and more economical and computer graphics become more and more available, their impact within the structural engineering community cannot be overlooked. This thesis has shown that interactive computer graphics should be the driving force behind the next generation of structural engineering computer programs. Not only will these programs use computer graphics but the approach used to input the problem and output the results will be based on interactive computer graphics. With the growing availability of computer graphics, programs should be written to use this valuable feature. This thesis has shown that it is possible and is now feasible to base a structural engineering computer program on interactive computer graphics. With the implementation of the interactive graphic feature, structural engineering programs will be abreast with current available computer technology.

In the near future, as new computer technology becomes readily available, computer programs of structural engineering application will once again need to be revised. Color graphics is already here but its economic feasibility is out of the reach of most computer users. Also, dynamic/refresh graphics, (which could be used to demonstrate modal shapes and progressive collapse,) is still not available to the general engineering community. Other computer advances such as array processors and super computers, with their phenomenal computational speeds and direct application to matrices, will have an even bigger impact on the structural engineering environment. The task of upgrading computer programs of structural engineering application is an ongoing process that must be maintained in order to keep structural engineering at its utmost level of professional competence.

APPENDIX A

PROGRAM USER'S MANUAL

The appendices discuss in detail the program presented in Chapter 4. Appendix A is a user's manual for the program. Appendix B is program documentation that discusses the FORTRAN program. Flow charts and logic flow diagrams are provided in Appendix C. Appendix D includes sample problems solved using this program. Appendix E is a listing of the FORTRAN program.

Following is a user's manual for a structural analysis program aided by interactive computer graphics. Written by the author of this thesis, this program includes many of the features discussed in the proposal for an interactive graphic structural analysis computer program. Though illustrations are provided, the user is encouraged to obtain hands-on experience with the program to appreciate the interactive graphic process.

Introductory Remarks

Briefly, the program will solve a plane frame linear elastic static structure. The reader is referred to Chapter 4 of this thesis for a more complete discussion of the program capabilities. Members are limited to one-dimensional beams of constant cross-section. These members are not required to be orthogonal to the global axes. Joint and member loads must be applied in the global and local coordinate systems respectively. The capabilities within this program should allow the engineer to model most simple structures.

Compared to existing structural analysis computer programs, this program is very interactive. There are many user-friendly routines incorporated into the program to make execution smooth and trouble-free. Fundamentally, program prompts are provided wherever user input is required. These prompts request user to enter specific parameters as well as inform the user of his/her location within the program. For example, all requests for main menu commands are prompted by the word "COMMAND?". Input prompts are not the only messages issued by the program. The program flags words and commands that it cannot interpret and sends an error message to the terminal. For example, misspelling the main menu command "SUPPORT" as "SUPORT" will cause the message "COMMAND NOT FOUND" and will return the user to the main menu. Likewise, trying to specify member properties for member #5 when only four members have been created will yield the message "INVALID MEMBER NUMBER". Bad parameter values are also flagged and the user informed. Not all bad parameter values are flagged. In the list routine, the user may specify a list range larger than the actual member or joint range. The program recognizes the larger value and adjusts the range to include only the correct list range. Graphics is another user-friendly feature. Most commands will cause an immediate graphic response to verify the user's input. Also, special graphic pictures can be requested. For example, the shear and moment diagrams of a member or the shape of the deformed structure can be requested.

Program Conventions

In order to use this program, the user must observe several program conventions. These conventions are, for the most part, similar to most other structural analysis computer programs.

Input Conventions

The language conventions are simple and flexible. All commands must be at least four characters in length. If more than four characters are entered, the extra characters are ignored by the program. The responses "yes" and "no" may be shortened to one character. When numerical values are requested, they may be entered in free format (no specific column format or decimal point required). A list of numerical values may be separated by a comma, a blank space or a carriage return. If an extra list value is accidentally given it will be ignored. All in all, the input conventions are very simple. The user should be warned of several hazards. Currently, the program does not parse all input. The entering of a real value or an alphanumeric where an integer value is requested will cause a computer system error and the program is aborted. The same result will occur if an alphanumeric character is given where a real value is requested.

The program uses the basic English units of inch, feet, kips and degrees. Joint coordinates and member lengths are in feet. The member constants and properties, E, A and I, have are the units KSI, IN^2 and IN^4, respectively. Structural loads and force output are expressed in KIPS, FT-KIPS and KLF. Joint displacements are given in INCHES and RADIANS. These units must be maintained as they are preset within the program.

Structural Conventions

The concept of global and local axes can best be related to the terms structural and member axes, respectively. The global axes constitute the coordinate system that refers to the total structure. (See Figure A.1.) The coordinates of the joints are given with respect to this Cartesian coordinate system. Joints and supports (and therefore, joint loads and support reactions) are always in reference to the global system. The member local axes requires further explanation.

Every member has its own unique local coordinate system. The local X axis of a member is always in line with the length of the member and originates at the start of the member. The other axis that is always in plane with the structure is the local Y axis. As always, this axis is perpendicular to the member's local X axis. In keeping with standard-right-hand rule, the local Z axis is perpendicular to the terminal screen (the plane of the structure) and extends out toward the user. (See Figure A.2.) With these conventions established, the only variable left is start and end of the member. This program uses the convention that the local Y axis will always have a component in the global positive Y direction. This uniquely specifies the start and end of a member and, thus, the local coordinate system. One exception to this is a perfectly vertical member (a column). In this case, the program assigns the side of the member with the lower global X coordinate as the start. (See Figure A.3.)

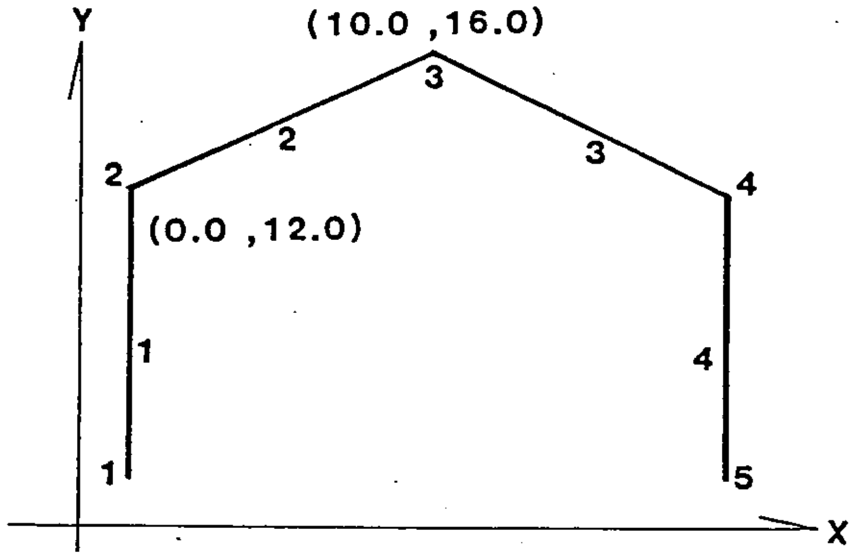


Figure A.1

Global (structural) Coordinate System

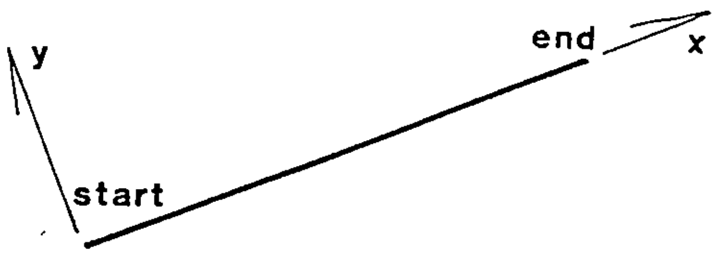


Figure A.2

Local (member) Coordinate System

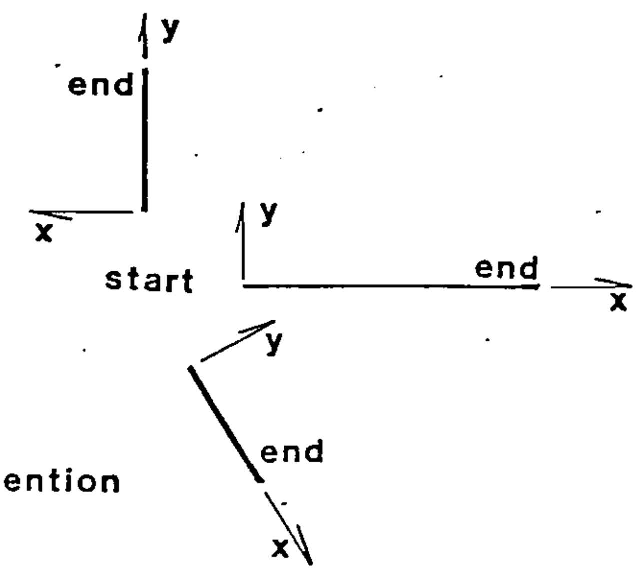


Figure A.3

Member Start-End Convention

Program Commands

Main Menu

The user enters the program via an introduction routine. Some introductory remarks are printed and then the user is placed at the main menu. The main menu is a location in the executive program where any major routine can be accessed. See Appendix C for a list of main menu commands. After the execution of any main menu routine, the user is returned to the main menu. Return to the main menu is always verified by the prompt "COMMAND?". When the user has completed his/her problem, exit from the program is gained by the main menu command QUIT.

Create Mode

Upon entering the program, the user usually goes directly into the create mode. The create mode can be divided into two sections - structure and data. These sections contain the following commands:

CREATE	
STRUCTURE	DATA
DIGITIZE	CONSTANTS
BUILD	PROPERTIES
BAYS	SUPPORTS
STORIES	RELEASE MEMBER
	LOADS

Note: When specifying any of these commands only the first four letters need to be given.

DIGITIZE:

The structure can be created by any of the routines in the structure section. To create a new structure, either the BUILD or DIGITIZE routine must be used. As mentioned earlier, all joints and members are numbered automatically by the computer as they are created. The program recognizes when the user is referring to a joint that has already been created and verifies this accordingly.

DIGITIZE allows the user to input the structure by tracing, on a digitizing tablet, a scaled drawing of the structure. DIGITIZE uses a repetitive process to input the structural configuration. Because of the repetitive process, there are no commands to be issued by the user. Upon entering the routine, the user is instructed to square the drawing with the internal grid of the tablet. A routine is provided to assist in this process. Next, a routine is run for the user to communicate to the computer the actual scale of the drawing. Then, a repetitive process is used to build the structure member by member. Joints and members are numbered sequentially in the order they are created. Exit from the DIGITIZE routine is gained by digitizing one last point far to the right on the tablet.

BUILD:

To create a frame from scratch, with no scaled drawing prepared in advance, the user must go into the BUILD routine. The frame is "built" by creating members. Members are created by identifying a member start and end. Three methods are available to identify the start of a member and five methods for the end. Method one, available to both start and end is, obviously, to specify the cartesian X and Y coordinates of the point. Method two requests the user to specify the number of a node that has already been created. The third method available to both start and end is to locate, with the graphic crosshair cursor, the location (X, Y coordinate) of the point desired. This is aided by the accurately scaled and labeled tic marks on the screen's perimeter. Methods four and five are available only to identify the end of a member, as they refer to the member end relative to the member start. In method four, the user enters an angle, in degrees, either positive or negative, and a length. The member end is defined by a radial line segment extending from the member start, at the specified angle, for the distance specified by the user. A stepping process is used for method five. The typewriter keyboard keys U, D, R, L are used to step the graphic cursor up, down, right, and left respectively. The location of the graphic cursor at the time the key E (for enter) is pressed becomes the location of the end of the member.

These five methods are accessed by a question and answer process with the computer program. The user may use any or all of the five methods to build the structural frame. Remember, every user response invokes an automatic graphic verification. Following is a sample of a structural input sequence using the BUILD routine. (See Figure A.1.)

> END1	Program prompt for method number to identify member start.
1	User selects Method 1
> X Y?	Program prompt for X and Y coordinate
0.0 0.0	for the member start node location
	User input X Y for member start
> END2	Prompt for method number for member end
1	User selects method 1
> X Y?	Program prompt
0.0 12.0	X, Y location for member end
>ANOTHER MEMBER?	Program prompt
Y	User responds "yes"
> END1	Program prompt
2	User selects method 2
> NODE #	Program prompt
2	User input Node number that the member start will be located

```
> END 2                  etc...
   1
> X  Y?
  10.0  16.0
> ANOTHER MEMBER?
  .
  .

  .                      NOTE:  during program execution, the
                         input sequence illustrated is
                         interactive and verified with
                         automatic graphic response.
```

The user should note that any of the five different methods presented may be used to attach a member to a joint previously created. As each point is created, its X and Y coordinates are compared with all of the other previously created nodes to see if it is within a specified tolerance of another joint. If so, the appropriate joint is re-identified (drawn) and the member attached. Exit from the BUILD routine can be gained at any time by responding N (no) to the prompt "ANOTHER MEMBER?". From here, the user may enter any other routine in the program, except DIGITIZE and RESTORE. It is inconceivable that the user would need to digitize the rest of a structural frame after a portion has already been created. Also, it would be virtually impossible for the user to line up the drawing on the tablet with the image of the frame already on the screen. But access from DIGITIZE to BUILD, BAYS, STORIES or any other routine is possible.

BAYS and STORIES:

The routines BAYS and STORIES provide two more ways in which members can be created. These routines create members that must attach to previously created members and must lie on an orthogonal grid. These routines prompt for user input and accept graphical input responses. Here again, no commands are issued by the user. As always, the members start and end and the joint and member numbering are taken care of internally, by the computer. As the title indicates, BAYS will create additional bays from one previously defined bay. Story height and bay width are determined by the previously defined bay. Although the original bay may not have been perfectly orthogonal, all subsequent bays are corrected to be exactly orthogonal. The additional bays may extend to the right or the left of the original bay. STORIES is similar in intent to BAYS but produces a structure in the vertical direction. Here, the user is prompted for the number of additional stories to be created, the number of bays per story, and the height of these stories. With this information, the program is able to automatically create and number the additional stories. These methods, BUILD, DIGITIZE, BAYS and STORIES are presently the only methods available to create a structure.

Other routines in the create mode accept the input for other structural data. Structural loads, member constants, member properties, supports and member end releases are each handled in separate routines in the Data section. Available through the main menu, these routines may be accessed in any order. Each of these Data routines provide auto-explanatory input prompts and help sections throughout.

CONSTANTS and PROPERTIES:

The specifying of member constants and properties, though in separate routines, are identical in operation. The member constant required by the program is E (the modulus of elasticity). The member properties needed are, A (cross sectional area) and I (moment of inertia) for the axis of bending. The entering of these parameters is aided by input prompts and error-messages. Additionally, a copy feature may be used to duplicate parameter values from one member to a list of members. A sample input sequence for member properties is given below:

> COMMAND?	Main menu prompt
PROPERTIES	Command to execute routine to enter member properties
> MEMBER PROPERTIES: > Az Iz	Message to inform the user of his/her location in the program and properties required
> MEMBER NUMBER	Program prompt
1	User specifies member 1
> PROPERTIES	Program prompt
10.0 100.0	User input properties
> MEMBER NUMBER	Program prompt
-1	User input to exit this routine
> COMMAND?	Main menu program prompt

<u>SUPPORT and RELEASES:</u>

When initially created, all joints have freedom of movement in all three degress of freedom. When two members attach to the same joint, the connection is assumed to be rigid. In order to alter either of these default conditions the user must use the supports and/or member end releases routines. All joints that are to be structural supports must be declared by the user. Upon entering the SUPPORT routine, via the main menu, the program prints the SUPPORT command options available:

SUPPORT OPTIONS

COMMAND	DESCRIPTION
TX	release support to translate in global X direction
TY	release support to translate in global Y direction
RZ	release support to rotate around Z axis
TT	release supports to translate in both X and Y
XR	release support translation X and rotation Z
YR	release support translation Y and rotation Z
NO	fully fixed support (release no degrees of freedom)

The user enters the number of the joint that will become a support followed by one of the seven support options. The specifying of member end releases is very similar to the specifying of support conditions. A member end can only be released from transmitting moment Z. After entering the number of the member whose end is to be released, the user enters one of the end release options; Start, End or Both. Exit from either of these routines is gained by responding "-1" to the prompt for the next joint or member number.

LOAD:

Currently, only a limited variety of uniform and concentrated loads is available. Diagrams representing the inventory of load types are given in the accompanying illustration (Figure A.4). This section has its own subset of commands. Each command is a mnemonic for a particular load condition. After the user enters the load type and the member or joint number, the program responds with the prompt:

MAG, LOC, LOC, LOC

This prompt is requesting the user to input load magnitude and location data. For joint loads, only the magnitude data are needed; but three dummy values of the zero must also be entered to satisfy the program. In placing concentrated loads on a member, the location of the load along the member must also be specified. A decimal fraction scheme is used to specify the load location. The member start is 0.0 and the member end is 1.0. Up to three separate locations along the member can be loaded with one execution of the command. To satisfy the program, three locations must be given. If fewer than three locations are desired, the user should

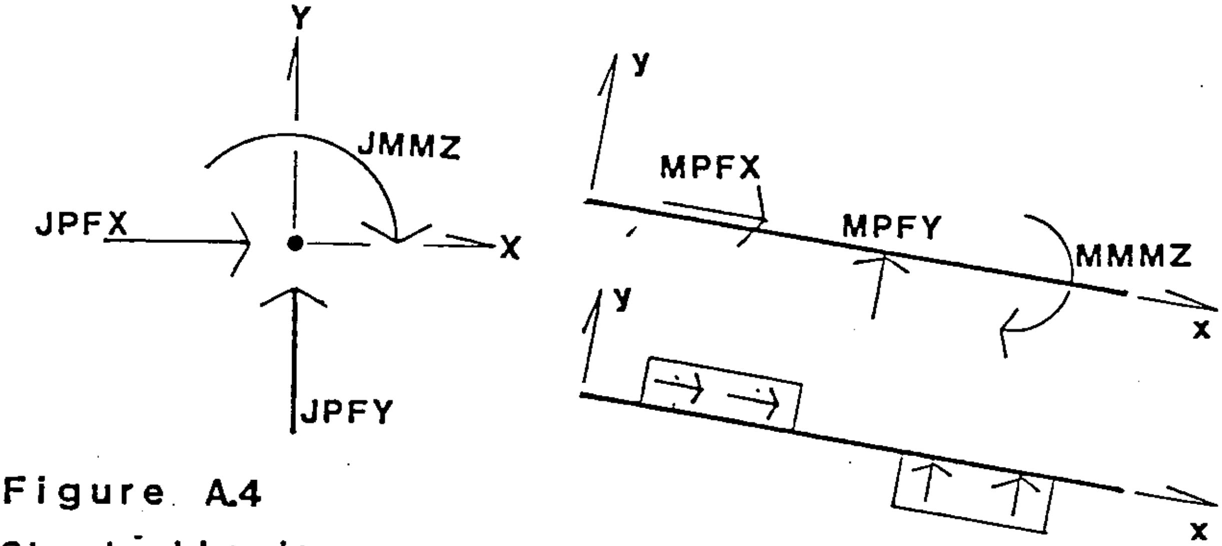
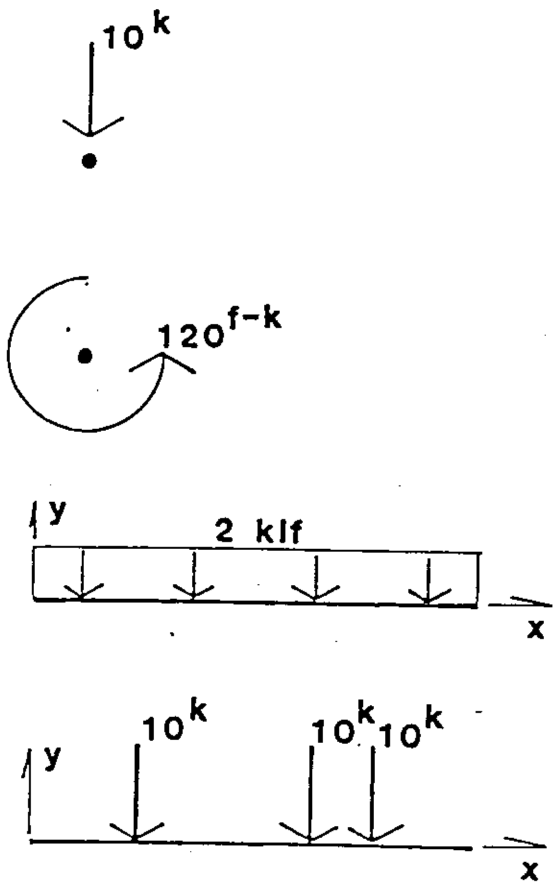


Figure A.4
Structural Loads



command	mag	loc	loc	loc
JPFY	-10	0	0	0
JMMZ	-120	0	0	0
MWFY	-2	0	1	0
MPFY	-10	.25	.66	.75

Figure A.5
Examples of Structural Loads

input 0.0 for the extra locations. The program ignores locations of 0.0 for concentrated loads. Uniform member loads are also speciifed via the same prompt. Once again, decimal fractions are used to specify the start and end locations of the uniform load. Examples of the loading commands are shown in figure A.5. The user exits the routine by answering "EXIT" in the response to the prompt "NEXT LOAD OR EXIT".

<u>LOADCASE</u>:

Different load cases and combinations are created by the subroutine LOADCASE. Listed below is the subset of commands available within this subroutine.

CREATE	To create a new load case
RENAME	To change the title given to a load case or combination
LIST	To list on the terminal screen the number and title for all current load cases and combinations
STORE	To store the current working load case
SWITCH	Switches load cases to make a different load case the active working load case
COMBINE	To produce a load combination
ACTIVATE	To specify which load cases are to be active in the post-processing phase
HELP	To list LOADCASE help section
EXIT	To exit LOADCASE subroutine

Up to five independent load cases and five dependent load combinations can be handled by this program. Load cases are numbered sequentially as they are created by the command CREATE. Because there can be only one active working load case at one time, a SWITCH command is provided. This command allows the user to specify which load case is the current active working load case. Load combinations are created by issuing the command COMBINE. Auto-explanatory prompts lead the user through the process of naming the load combination and specifying the load case factors that make up this load combination. The ACTIVE command allows the user to specify which load cases are to be active during post-processing. For post-processing, two additional load combinations are automatically created. The names of these load combinations are MAXIMUM ENVELOPE and MINIMUM ENVELOPE. These two special load combinations are created by extracting maximum and minimum data results from the load combinations and the active load cases.

Graphic Routines

COMMANDS

REDRAW	Redraws the basic structure of member numbers and joint members
SETUP	To enter the routine to set the flags associated with the STRUCTURE command
STRUCTURE	Redraws the structure according to the flag set in SETUP
ZMIN	To execute the routine to zoom in on the display
ZMIO	To execute the routine to zoom in or out on the display
DEFORMATION	Draw the deformed shape of the structure

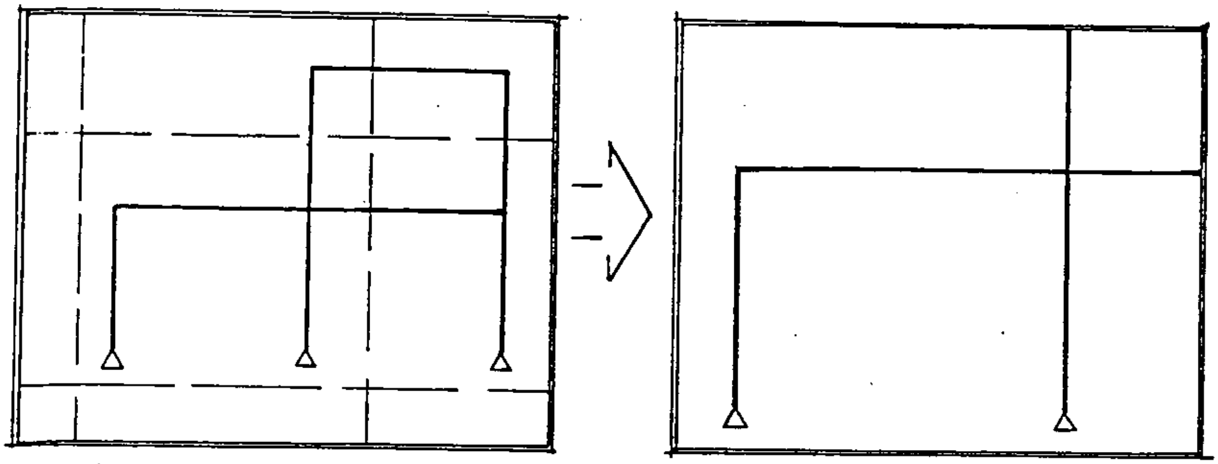
As mentioned earlier, interactive graphics is the dimension that is missing from current structural analysis computer programs. Therefore, this program provides automatic graphic response to commands as well as five routines that are devoted to the graphic verification of the structural data.

The whole structure can be redrawn by two different redraw commands - REDRAW and PLOT. Accessed from the main menu, REDRAW will erase the screen, and redraw the structure with all of the latest updated data incorporated. Only members and joints and the global axes are redrawn with this command. PLOT can draw a more comprehensive picture of the structure. In conjunction with the SETUP command, more control over the display is obtained using the PLOT command. The routine SETUP offers the user six options. These commands are a switch command and will either activate or deactivate the function.

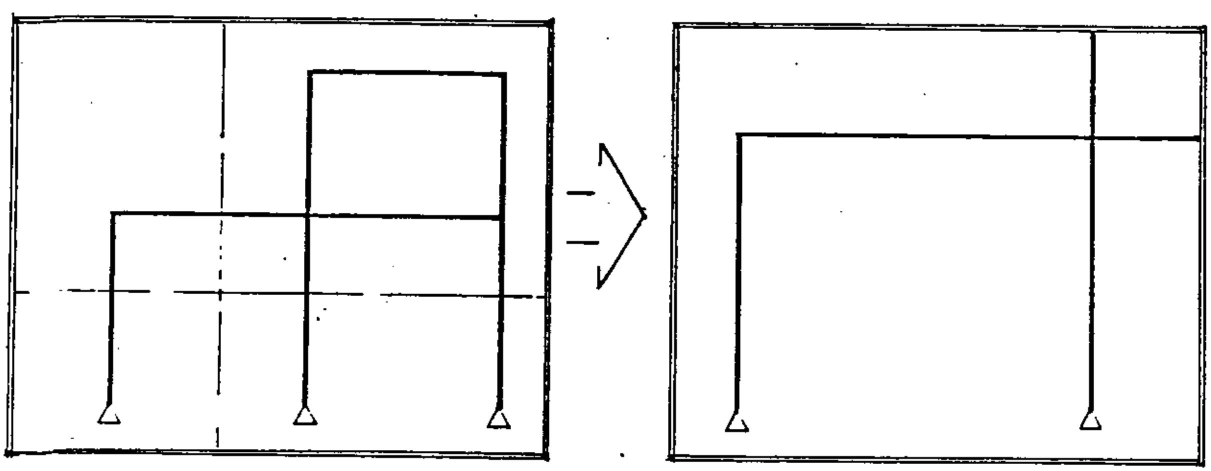
JNUM	Display joint numbers
MNUM	Display member numbers
MEMB	Display members
LOAD	Display loads
RELE	Display member end releases
SUPP	Display support conditions

The command STRUCTURE is the acutal command that will invoke the redrawing of the structural frame. With SETUP and PLOT, the display any piece of structural information may be turned on or off.

Another graphic feature is the zoom command. The first zoom command, ZMIN can only zoom in on the display. Upon entering this routine, the graphic cross hairs are displayed. With the graphic cross hairs, the user locates the bounds for the lower left and the upper right corners of the next display. ZMIO allows the user to zoom either in or out. This routine utilizes a scaling factor. The user inputs a factor less than one or greater than one to represent how much of the current display he/she wishes to be present in the next display. Then, the user selects the point on the current display he/she would like to place at the center of the next display. (See figure A.6.) These routines allow the user to more closely inspect or refine a portion of the structure at any enlarged scale. Between the graphic commands to zoom and redraw the structure, the user has complete control over the display.



ZMIN



0.5 factor

ZMIO

Figure A.6

Zoom Functions

Structural Data Manipulating

The next group of commands, all available from the main menu, are the commands to print structural data. These routines cover a wide range of functions but they all deal with the structural data in a printed list form. The commands in this section are: LIST, CHANGE, DELETE, DATA, RESULTS, ANSWERS, SAVE and RESTORE.

LIST:

LIST, as the name implies, will list in printed form any information about the structural data. A subset of commands used in list are:

NODE	List joint X,Y coordinates
MEMB	List member incidences and lengths
SUPP	List support conditions
RELE	List member end releases
JLOA	List joint loads
MLOA	List member lodds
CONST	List member constants
PROP	List member properties
EXIT	Exit LIST section
HELP	Print LIST help section

After issuing one of these commands the program will prompt the user to enter the range (beginning and end numbers). All printed lists are clearly labeled with title and units.

CHANGE:

Similar to LIST, CHANGE can alter the value or conditions of a specific parameter. Not all structural data can be altered. For instance, rather than change the value of a load on a member, it would be more appropriate for the incorrect load to be deleted and a new load to be respecified. The data that can be manipulated in the Change section are:

NODE	Change joint X,Y coordinates
MEMB	Change member incidences
SUPP	Change a support condition
RELE	Change the end release condition of a member
CONST	Change the value of E for a member
PROP	Change the value of I or a for a member
EXIT	Exit change section
HELP	Print change help section

For the user's convenience, prompts and information messages are issued throughout the CHANGE routine.

DELETE:

Delete can manipulate the parameters that CHANGE cannot. The structural items that may be deleted are:

NODE	Delete a node from the structure
MEMB	Delete a member from the structure
SUPP	Return a joint to free
RELE	Delete all end releases for a member
JLOA	Delete a load on a joint
MLOA	Delete a load on a member
EXIT	Exit delete section
HELP	Print delete help section

Program consistency checks are provided in the DELETE section in order to always keep all the structural information valid. For instance, when deleting a member, all of the loads on that member are also deleted. Also, when deleting a node, the program checks to see if there are any members attached to this node. If a member is attached to the node, the to the node, the deletion process is not carried out and the user is informed of the situation. All of these routines: LIST, CHANGE and DELETE, provide informational messages and prompts throughout. Also, help sections are provided. Exit from any of these routines is obtained by issuing the command EXIT.

DATA:

The other routines in the data manipulation section deal with external storage files. From the main menu, the command DATA will cause an external file to be set up and all of the current structural data to be sent to this file. (See Table A.1 for an example of the data output.) The program prompts the user to enter a name for the new external file.

RESULTS and ANSWERS:

After the program solution, the results can be sent to an external file using the RESULT command. Lists of numeric data are output for all load cases. This output is in fixed format. (See Table A.2.) Specifically, the information the user may have printed to an external storage file is:

FORC	Write out the forces (in local) at the end of each member
SUPP	Write out the support reactions for each support
DISP	Write out the displacements (in global) for each joint
SECF	Write out the sectional forces (at 1/10 points) for a member
EXIT	Exit this section
HELP	Print results help section

NOTE: All load cases and combinations are written out for each command.

The routine ANSWERS is identical to RESULTS except the data is immediately printed on the terminal screen.

SAVE and RESTORE:

Lastly, a SAVE/RESTORE command is available. With this feature, a user is able to save data that describes the structure; then at a later

Table A.1
Sample DATA Output

```
JOB TITILE:
**************************************************************
*                SAMPLE PROBLEM - HOIST FRAME                *
**************************************************************

NUMBER OF MEMBERS      ->   6
NUMBER OF JOINTS       ->   5
NUMBER OF LOAD CASES   ->   1
NUMBER OF LOAD COMB    ->   0

NODE #______X___________Y__
   1      0.000       0.000
   2     10.000       0.000
   3     20.000       0.000
   4      0.000      10.000
   5     10.000       5.000

MEMBER #___BEGIN____END____LENGTH
   1          1.     2.     120.00
   2          2.     3.     120.00
   3          4.     5.     134.16
   4          5.     3.     134.16
   5          1.     5.     134.16
   6          2.     5.      60.00

SUPPORT JOINT #_____FIXED
        1           TX TY
        4           TX TY

MEMBER #____RELEASES

MEMBER #___ E ksi ___
   1       29000.
   2       29000.
   3       29000.
   4       29000.
   5       29000.
   6       29000.

MEMBER #___ AREA ___ Iz ___
   1        4.00     100.00
   2        4.00     100.00
   3        4.00     100.00
   4        4.00     100.00
   5        4.00     100.00
   6        4.00     100.00

********************** LOAD CASE #  1 *********************

  LOAD CASE TITLE ---> JOINT LOAD AT JOINT 3
JOINT #___DIRECTION___MAGNITUDE
   3         FY         -20.000

MEMBER #___TYPE_____DIR____MAGNITUDE__BEG___END

LOAD COMBINATION DATA

************ ACTIVE POST-PROCESSING LOAD CASES ***********
   1      JOINT LOAD AT JOINT 3
  11      MAXIMUM ENVELOPE
  12      MINIMUM ENVELOPE
$
```

Table A.2
Sample RESULTS Output

```
JOB TITILE:
**************************************************************
*              SAMPLE PROBLEM - HOIST FRAME                  *
**************************************************************
```

JOINT DISPLACEMENTS

JOINT #	LOAD CASE	TRANS X	TRANS Y	ROTATE Z
1				
	1	0.0000	0.0000	0.0004
2				
	1	-0.0562	-0.1299	-0.0041
3				
	1	-0.1123	-0.5852	-0.0028
4				
	1	0.0390	-0.1233	-0.0026
5				
	1	0.0000	0.0000	0.0003

* MEMBER END FORCES *

MEMBER #	LOAD CASE	JOINT #	AXIAL	SHEAR	MOMENT
1					
	1	1.	54.30	-1.83	0.00
		2.	-54.30	1.83	-18.30
2					
	1	2.	54.30	10.87	18.30
		3.	-54.30	9.13	-9.58
3					
	1	4.	-61.59	1.77	10.21
		3.	61.59	-1.77	9.58
4					
	1	5.	-77.81	-0.91	0.00
		4.	77.81	0.91	-10.21
5					
	1	1.	17.55	0.00	0.00
		4.	-17.55	0.00	0.00
6					
	1	2.	-12.70	0.00	0.00
		4.	12.70	0.00	0.00

* SUPPORT REACTIONS *

JOINT #	LOAD CASE	FORCE X	FORCE Y	MOMENT Z
1				
	1	70.2574	6.1493	0.0000
5				
	1	-71.1476	34.5364	0.0000

date, restore the information into the program and continue with the problem solution. All information about the problem is saved, from joint coordinates to structural loads. To execute these procedures the user need only answer the auto-explanatory prompts within the routines.

Solution and Structural Output

SOLVE:

After all of the structural data have been entered into the program and the user has verified its correctness, the solution routine may be invoked. The main menu command SOLVE will cause the program to begin the solution process.

First, the structural data are reviewed to insure that all required data is present. This includes making sure all members have an E, I, and an A. Any of these problems will cause errors in the subsequent matrix equations. If the data pass all checks, the formulation of the matrix equation is performed. Lastly, the post-processing calculations are performed.

INDIVIDUAL and DEFORMATION:

After the solution process, the user is once again returned to the main menu. From the main menu, the user may access the post-processing routines. INDIVIDUAL is the command that invokes the displaying of each member's shear and moment diagrams and envelopes. Once again, the user need only answer the prompts in order to display these member performance plots. The other post-processing graphic routine is DEFORMATION. This routine will display the structure with displaced joints and members. When the output from DEFORMATION is used in conjunction with the PLOT command, a clear picture of the structure's response to the loads can be seen. The RESULT command, another post-processing command, was discussed previously.

Main menu At this stage, most of the other computer programs are done; this program is not. From here, after the results have been interpreted, the user may once again change the structural data. The purpose of this feature is to allow the user direct interactive control over the structural data in order to investigate several options for the structure. With the use of the commands DATA, RESULTS and SAVE, the current structural alternatives can be saved in an external file before the next structural design is begun.

Concluding Remarks

With this program, the process of entering, interpreting and refining a structure has been revolutionized. With the interactive graphics and the automatic numbering of joints and members, the user's input process is greatly simplified. The program prompts and error-messages also help the the user through the input phase. In the post-processing phase, interactive graphics help the engineer to interpret the results. The next logical step is also provided. With the results reviewed and interpreted, refinements to the structure can be immediately incorporated into the structural model and a new refined problem solution obtained.

APPENDIX B

PROGRAM DOCUMENTATION

Introduction

This Appendix describes in detail each subroutine in the FORTRAN program. The structure of the program is very straightforward. Most main menu commands execute one and only one subroutine. The names of the variables relate to the data stored within. For example, the variable TALLY is a counter that contains the current number of members in the structure. Basically, the logic flow of the program is simple and direct. Because the program is organized around one executive program that calls many subroutines, modifying and expanding the program is easy.

The hardware required to run the program is basic. This program was created on a DIGITAL VAX 11/780 computer running under the VAX/VMS operating system. Either the TEKTRONIX 4014 or the TEKTRONIX 4051 storage tube grahpics terminal can be used to access the program. In order to use all of the features within the program, a 4014 is required. Features such as four different character sizes and the digitizing tablet are only available through the TEKTRONIX 4014. Optional hardware features are: a TEKTRONIX 4594 digitizing tablet, a TEKTRONIX 4631 hardcopy unit, and a TEKTRONIX 4663 interactive digital plotter. The required hardware of a DIGITAL VAX cpu and a TEKTRONIX grahpics terminal is fairly standard and can be found in installations around the world.

The software required to run the program is basic and is not site dependent. The program is written in standard DIGITAL version of FORTRAN-77. The program does utilize a few character handling routines that are unique to DIGITAL FORTRAN-77. (See the subroutine BUILD for an example of these routines.) Except for these few routines, any FORTRAN-77 compiler will suffice. The graphics package used in this program is the TEKTRONIX Interactive Graphics Library (IGL). Only the basic IGL package is needed. No other graphic software options are required.

The organization of the program is simple and straightforward. See Appendix C for flow charts and logic flow diagrams. Conceptually, the program is founded around a central data base. This data base is a collection of variables that contain the structural data to describe the frame. All of the variables originate in the driver program (main menu) and are shared throughout the other subroutines via FORTRAN common blocks. Therefore, whenever any piece of information is added, changed, or deleted the central data base is automatically updated. The main menu resides in the driver program. From the main menu any major subroutine can be accessed. The user is required to initiate every step of the problem set-up and post-processing by issuing a command to execute procedures. For a list of the main menu commands see the logic flow chart in Appendix C.

Subroutines

PARSE:

calls: BUILD, STORIES, BAYS, DIGI, PROP, CONST, SUPP, RELE, LOAD, ERASE, CHAN, GRAPHIK, REDR, LIS, JCASEACT MCASEACT, LOCASE, GLOBSTIF, BNASMBL, SOLVE, CASEFORC, CASEMOSH, INDIV, ZOOM, LOADCASE, OUT, RESULTS RECASE, ENVEL, FACTOR, SAVE, RESTORE, ZERO, DEFLECT, ANSWERS

called by: INTRO

This routine is the driver program for the rest of the subroutines. The main menu resides in this program. All major routines are accessed from this program. The central data base is founded in PARSE. All of the data needed to describe the structure resides in this subroutine.

The program is organized around a menu. This menu contains all of the main menu commands that a user is able to issue. When the user issues a command the main menu tries to resolve the command against a table of valid commands. If the command issued by the user matches a command in the table, program execution is transferred to the subroutine associated with that command. If the command is not matched, an error message is printed on the terminal screen requesting the user to try again. All of the routines used to create the structural data, solve the problem, and post process the results are accessed via this program. Exit out of the whole program is gained by issuing the main menu command QUIT.

BUILD:

call: SAMENODES, REDR
called by: PARSE

BUILD is the main subroutine used to create structural members and nodes. The screen size and rounding increment are set-up by this subroutine. A member is created by specifying the location (coordinates) of each member start and end. Three methods are available to locate the member start and five methods available for the member end. This subroutine uses one special routine. The statement "IRET=SYS$QIOW(,..." is a FORTRAN read statement. This is a VAX routine that will accept one character from the keyboard (without printing it on the terminal screen) and will automatically generate a character return.

The subroutine is organized around five options.

1 - X Y coordinate
2 - node #
3 - locate
4 - angle and length
5 - step

These are the five routines to locate the ends of the member. A response "N" to the prompt "ANOTHER MEMBER" will exit the user from this subroutine.

The main variables in BUILD are as follows:

NLOC	real	(40,2)	holds up to 40 pairs of (X,Y) node coordinate locations
MT	real	(40,12)	holds the information that describes each member (up to 40 members), 12 registers are used for each member 1 node number of member start 2 node number of member end 3 not used 4 not used 5 member length, in inches 6 constant E, in KSI 7 Ax cross-sectional area, in IN^2 8 Iz moment of inertia, in IN^4 9 not used 10 not used 11 not used 12 not used
NT	integer		counter for the number of nodes that have been created
TALLY	integer		counter for the number of members that have been created

<u>DIGI</u>:

calls: SQUARE, PAGE, SAMENODE
called by: PARSE

DIGI will allow the user to input the structural members into the program via a digitizing tablet. Screen size and rounding increment are specified by the user via routines within the subroutine. This subroutine uses a repetitive process of digitizing each member by locating the start and end of each member. Exit from the routine is gained by digitizing a point on the far right side of the tablet.

Main Variable Accessed: MT, NLOC, TALLY, NT

<u>BAYS</u>:

calls: SAMENODES
called by: PARSE

BAYS will automatically generate additional structural bays identical to one that has already been created. The user identifies a previously created structural bay, of three members, by identifying the four nodes that correspond to this bay. The user then specifies whether the new bays are to extend to the right or left of the original bay and the number of additional bays that are to be generated. The program calculates the bay height and width and the new joint and member numbers. Note that even though the members of the original bay may not have been orthogonal the new bays are created orthogonal to the global axes. The routine is automatically exited after the new bays are created.

Main Variables Accessed: MT, NLOC, TALLY, NT

<u>STORIES</u>:

calls: SAMENODES
called by: PARSE

STORIES will automatically generate the structural members for additional stories. The user is prompted for the number of additional stories, floor height, and the number of bays in the next story. Next, the user identifies the nodes at the base of the first new story. The subroutine automatically generates the joint locations and numbers and the member incidences and numbers. The subroutine is automatically exited after the new stories are generated.

Main Variables Accessed: MT, NLOC, TALLY, NT

PROP:

calls: none
called by: PARSE

PROP is the subroutine that accepts the values for the member properties. Currently only the properties Ax and Iz (cross-sectional area and moment of inertia, respectively) are required by the program.

Two methods are available to specify the properties for the members. The first method is to specify the member number then the properties. The second method will copy the properties from one member to many other members. This method is accessed by answering "0" to the prompt "MEMBER NUMBER". The members to receive the properties are determined by the FORTRAN Do loop. The begin, end and increment parameters are given by the user. PROP places the member property values for each member directly into the variable MT. Exit from the program is gained by answering "-1" to the prompt "MEMBER NUMBER".

Main Variables Accessed: MT, TALLY

CONST:

calls: none
called by: PARSE

The subroutine CONST accepts the constant E for each structural member. This subroutine is identical in operation to the subroutine PROP.

Main Variables Accessed: MT, TALLY

RELE:

calls: none
called by: PARSE

The subroutine RELE allows the user to specify the members whose ends cannot transmit moment. RELE is also identical in operation to PROP. Instead of specifying a parameter value, the user specifies one of the letters S, E, or B. The letters correspond to releasing the member start, end or both ends from carrying moment. The program code for member end releases is: 0 - no releases, 1 - both ends released, 2 - only start released, 3 - only end release. Unless changed by the user, all members have both ends rigidly connected to the joints.

Main Variables Accessed: TALLY

MBREL	integer	40	holds the code that specifies the member end release for each member

SUPP:

calls: none
called by: PARSE

This subroutine allows the user to specify which nodes are to be structural supports. Within this subroutine, the user specifies which degrees of freedom, if any, are to be released for each support. For a list of the support options and the syntax used to specify each of these options see the user's manual, Appendix A. This subroutine is identical in operation to PROP. The program code for a structural support is as follows. A 3 digit integer is used; each digit can be either a "1" or a "0". Each digit place (hundreds, tens and ones) in the number correspond the nodal degrees of freedom, translation X, translation Y, and rotation

Z, respectively. The digit "1" in one of the positions represents the degree of freedom is restrained from movement, e.g., 111 - all 3 degrees of freedom are fixed, 110 - only the rotation Z is released for this support. If the code for a node is 000 the joint is not a support.

Main Variables Accessed: NT

SREL	integer	40	holds the code that specifies the support condition

LOAD:

calls: none
called by: PARSE

Structural loads, applied to both the joint and member, are specified using the subroutine LOAD. LOAD works only on the load case that is currently active. For a list of the mnemonic of the loading types, see the user's manual, Appendix A. To specify a load on the structure, the user issues a LOAD command from the menu. This command is compared to a table of valid LOAD commands. If the command is matched, execution is passed to the part of the subroutine that requests the loading magnitude and location. A two part code is used to specify the loading condition. Part 1 is the load type. The integer digits 1, 2, and 3 represent the loads types: concentrated, uniform, and applied moment, respectively. Part 2 is the direction which the load is applied: again the digits 1, 2, and 3 represent the direction X, Y and Z respectively. Exit is gained by issuing the "EXIT" command to the prompt "NEXT LOAD OR EXIT".

Main Variables Accessed: TALLY, NT

MLOAD	real	(40,6)		holds the specifications for each member load
			1	number of the member where this load is applied
			2	code for load type
			3	code for load direction
			4	magnitude of load, in Kips or F-K
			5	location of concentrated load or start of uniform load (in decimal fashion)
			6	location of end of uniform load along member
MLTALLY	integer			counter for the number of member loads
JLOAD	real	(40,3)		holds the specifications for the joint loads
			1	number of the joint where this load is applied
			2	code for load direction
			3	magnitude of load, in Kips or F-K
JLTALLY	integer			counter for the number of joint loads

LOADCASE:

calls: none
called by: PARSE

This subroutine allows the user to create and manipulate load cases and load combinations. The program can handle up to 5 independent load cases and 5 dependent load combinations. To execute a particular routine in this subroutine the user issues one of the LOADCASE commands. See the user's manual, Appendix A, for a list of the commands available.

Main Variables Accessed:

CASES	integer		counter for the number of independent load cases created
LCASE	integer		holds the number of the load case currently active
MCASE	integer	5	holds the number of member loads specified for each of the 5 load cases
JCASE	integer	5	holds the number of joint loads specified for each of the 5 load cases
NMCASE	real	(5,40,6)	holds all of the member loads for all 5 load cases - format same as MLOAD
NJCASE	real	(5,40,3)	holds all of the joint loads for all 5 load cases - format same as JLOAD
NCOMB	integer		counter for the number of dependent load combinations created

COMB	real	(5,5)	holds the factors that specify the dependent load combinations -each row of the array corresponds to a different load combination -each column corresponds to one of the 5 load cases.
ACASES	integer		counter for the number of load cases and combinations that are to be active in the post-processing phases
ACTLIST	integer	(10)	holds the number of the load cases and combinations active for post-processing
NAME	char*30	(10)	holds the name, given by the user, for each load case and combination

LIS:

calls: none
called by: PARSE

This subroutine will list, in printed form, any or all of the data about the structure. For a list of the commands available, see the user's manual, Appendix A. LIS is a menu driven program similar to PARSE and LOAD. The user is prompted for a list name and a list range. The subroutine automatically accesses the correct variable in the cental data base and prints the requested information on the screen. Before printing the information, all program codes are converted to user oriented standard, english mnemonics.

Main Variables Accessed: MT, NLOC, TALLY, NT, SREL, MREL, JLOAD, MLOAD, MLTALLY, JLTALLY

CHAN:

calls: SAMENODE
called by: PARSE

CHAN allows the user to change the values of certain structural parameters. See the user's manual, Appendix A, for the list of commands available. The operation of CHAN is identical to LIS.

Main Variables Accessed: MT, NLOC, TALLY, NT, MREL, SREL

<u>ERASE</u>:

calls: none
called by: PARSE

Similar to LIS and CHAN, ERASE will alter the central data base by deleting specified values and parameters from the structural data. In addition to deleting values and parameters, ERASE performs a consistency check in order to keep all data valid. Currently, if a member or joint is deleted the whole structure is not renumbered, the member location in the variable MT is merely filled with a dummy value to indicate item has been deleted. In other variables, such as MLOAD, the load to be deleted is simply overwritten. The load specifications in the MLTALLY (last position) are transferred to the location of the load to be deleted, then MLTALLY is reduced by one.

Main Variables Accessed: MT, NLOC, TALLY, NT, JLOAD, MLOAD, MLTALLY, JLTALLY, SREL, MREL

<u>OUT</u>:

calls: none
called by: PARSE

OUT is similar to LIS. This subroutine writes to an external file all the structural data required to describe the current frame. No commands are issued by the user. All program codes are converted to the standard user oriented mnemonics. Execution is automatically transferred back to PARSE after the FORTRAN write is completed.

Main Variables Accessed: NT, NLOC, TALLY, NT, MREL, SREL, MCASE, JCASE, NMCASE, NJCASE

<u>RESULT</u>:

calls: none
called by: PARSE

RESULT will write to an external file the structural results of the most recently solved problem. The subroutine is menu driven and thus allows the user to be selective when outputting the structural results. For the specific types of results able to be output see the user's manual, Appendix A. Exit from the routine is gained by issuing the command "EXIT".

Main Variables Accessed: MT, TALLY, NT

EMCASE	real	(10,40,0)	holds the 6 member end forces for each member for each load case and combination in K,K,F-K, K,K,F-K, respectively
SUPCASE	real	(10,40,3)	holds the 3 reactions for each joint that is a support for each load case and combination
SECTFORC	real	(12,40,21,3)	holds the 3 forces, axial, shear, and moment, at 21 equally spaced sections along each member for each load case and combination

SAVE:

calls: none
called by: PARSE

SAVE is identical in operation to the subroutine OUT. SAVE writes to an external file all of the data required to describe a structure. The difference between the subroutines OUT and SAVE is that the output from OUT is intended for the user to read (all program codes are converted) and the output from SAVE is intended to be read back into the program. When writing into an external file, SAVE keeps all data in coded form. After the FORTRAN write is completed, execution is automatically transferred back to PARSE.

Main Variables Accessed: MT, NLOC, TALLY, NT, MCASE, JCASE, NMCASE, NJCASE, SREL, MREL

<u>REDR</u>:

calls: PAGE
called by: PARSE

REDR will draw on the terminal screen the basic data that describes the current structure. A call to the subroutine PAGE is made to clear the screen. Then, REDR draws on the terminal screen the members, member numbers, and joint numbers. Execution is automatically transferred back to PARSE when the drawing is completed.

Main Variables Accessed: MT, TALLY, NLOC, NT

ZX, WX, ZY, WY	real	Holds the value, in virtual units, of the limits of the working screen
ROUND	real	Holds the rounding increment, in virtual units

<u>PAGE</u>:

calls: none
called by: PARSE, REDR, GRAPHIK

The subroutine PAGE will clear the screen then draw on the screen the global axes and the global virtual scale. PAGE displays tic marks that are 2.5% of the screen height at intervals of 5 times the rounding increment.

Main Variables Accessed: ZX, WX, ZY, WY, ROUND

<u>PLOT</u>:

calls: PAGE, DRWLOAD, DRWSUPP, DRWMREL
called by: PARSE

The subroutine PLOT is similar to the subroutine REDR. PLOT is able to display additional information on the terminal screen. The routine SETUP is used in conjunction with PLOT. The logical variable SET(7) is used to determine whether or not the structural data are to be labeled.

Execution is automatically passed back to PARSE when the drawing is completed.

Main Variables Accessed: MT, TALLY, NLOC. NT, MREL, SUPP, JLOAD, MLOAD, MLTALLY, JLTALLY

SET	LOGICAL	(7)	Holds the flags for the 7 pieces of structural data that are able to be turned on and off

ZOOM:

calls: none
called by: PARSE

The subroutine ZOOM allows the user to obtain a close look at the structure. A scale factor is entered by the user. This factor is multiplied by the present virtual distance across the screen to determine the virtual distance across the screen for the next display. The user then specifies the point on the current display that he/she wishes to be located at the center of the next display. With this information the virtual limits for the next display are calculated. Execution is passed back to PARSE which in turn calls REDR.

Main Variables Accessed: ZX, WX, ZY, WY

JCASEACT:

calls: none
called by: PARSE

The subroutine JCASEACT assembles the joint loads into a structural global action vector. This vector contains all the global joint loads that act on the structure, for each independent load case.

Main Variables Accessed: JCASE, NJCASE, CASES

ACT	real	(10,120)	Holds the load at each of 40 joints, each with 3 degrees of freedom, for each of 5 load cases - after the solution routine ACT holds the displacement of each degree of freedom for each load case and combination

MCASEACT:

calls: (each of these subroutines calculates the equivalent joint load for a particular load condition)

PX	concentrated load directed along the member X axis
PY	concentrated load directed along the member Y axis
MMZ	concentrated moment applied to a member
MX	uniform load applied to a member along the local X axis
MPY	uniform load applied to a member along the local Y axis over a portion of the member
GLOBCASE	see description below

called by: PARSE

This subroutine resolves the program member load code, in the variable MCASE, to determine which load case subroutine to call. The appropriate load case subroutine determines the equivalent joint loads due to this member load. After the load type subroutine has calculated the six equivalent joint loads in the members local coordinates, a call is made to the subroutine GBLDCASE to transform the joint loads into global. MCASEACT also sums the equivalent joint loads for each member due to all the loads on the member for each load case. The resultant equivalent joint load for each member is stored in the variable EMCASE.

Main Variables Accessed: MCASE, NMCASE, MT, TALLY, ACT, MREL, CASES, NLOC

EMCASE	real	(10,40,6)	Holds the summed equivalent joint loads for each member for each load case and combination

GBLDCASE:

calls: none
called by: MCASEACT

This subroutine calculates the sine and cosine for each member then transforms the member equivalent joint loads into global. Also, GBLDCASE assembles the equivalent joint loads for each member load into the global action vector according to the joint numbers at the member ends.

Main Variables Accessed: ACT

LOCASE:

calls: none
called by: PARSE, CASEFORC

The subroutine LOCASE will calculate the local stiffness matrix for a member. This subroutine takes into consideration the member end releases when calculating the stiffness matrix. LOCASE accesses the member properties, constants, and length in the variable MT.

Main Variables Accessed: MT

GLOBSTIF:

calls: none
called by: PARSE

This subroutine transforms the member local stiffness matrix calculated by the subroutine LOCASE into a member global stiffness matrix.

Main Variables Accessed: MT, NLOC

BNASMBL:

calls: none
called by: PARSE

BNASMBL completes the stiffness matrix phase by assembling each member global stiffness matrix into a single banded global stiffness matrix. This subroutine takes into consideration the joint numbers at the member ends and the overall band width.

Main Variables Accessed:

BASS	real	(120,120)	holds the overall stiffness matrix for the structure, in banded form

SOLVE:

calls: none
called by: PARSE

The subroutine SOLVE solves the matrix equations for the unknown displacements of the free degrees of freedom. SOLVE alters the stiffness matrix, BASS, and load vector, ACT, to account for the structural support conditions, SREL. The solution process is Gauss elimination with backward substitution.

Main Variables Accessed: ACT, BASS, NT, SREL

CASEFORC:

calls: MULT6X1, LOCASE
called by: PARSE

CASEFORC will determine the resultant joint forces acting on each member for each load case. When calculating the resultant actions at the ends of a member, this subroutine takes into consideration both the joint displacement and the applied member load. The joint displacements (in the variable ACT) are transformed into the member local coordinate system and then multiplied by the member stiffness matrix to determine the local end forces. The equivalent joint forces, (stored in the variable EMCASE) due to applied member loads have been previously determined by the subroutine MCASEACT.

Main Variables Accessed: EMCASE, MT, ACT, NLOC, MREL, TALLY, CASES

MULT6X1:

calls: none
called by: CASEFORC

This subroutine multiplies a 6 x 6 matrix to a 6 x 1 matrix.

RECASE:

calls: none
called by: PARse

RECASE will calculate the support reactions for the independent load cases. RECASE accesses the variable SREL, EMCASE and MT to determine which joints are support and the reactions at the supports.

Main Variables Accessed: SREL, EMCASE, MT, TALLY, NT, SUPCASE, CASES, NCOMB, NLOC

CASEMOSH:

calls:	PYMOMSHE	handles a concentrated load direct in the member local Y direction
	MYMOMSHE	handles a uniform load directed in the member local Y direction
	MPYMOMSHE	handles a uniform load, applied over only a portion of the member, directed in the member local Y direction
	MMZMOMSHE	handles a concentrated moment applied within a member

called by: none

CASEMOSH is very similar to the subroutine CASEFORC in that they both organize the calculation for member forces due to applied member loads. Again, the contents of the variable MCASE is used to determine which load type subroutine is needed. CASEMOSH calculates the forces at 21 equally spaced sections within a member. The shear and moment forces are calculated for all members for all load cases.

Main Variables Accessed: MCASE, NMCASE, TALLY, MREL, CASES

<u>FACTOR</u>:

calls: none
called by: PARSE

The subroutine FACTOR creates the results for the dependent load combinations from the independent load cases. The dependent load combinations are calculated using the load case factors in the variable COMB. Because the program assumes a linearly elastic structure, all load combinations are created by factoring and summing the independent load cases. First, the joint displacmenets are calculated for the load combinations using the variables COMB and ACT. Secondly, the forces acting at the member ends are computed using the variables COMB and EMCASE. Next, independent load case support reactions are factored and summed in order to obtain the load combination support reactions. And lastly, each member's sectional forces (axial, shear and moment) for the load combinations are computed.

Main Variables Accessed: ACTLIST, ACASES, SECFORC, SUPCASE, ACT, COMB

<u>ENVEL</u>:

calls: none
called by: PARSE

ENVEL will determine the maximum and minimum value for each force for each section of each member. All the active load cases and combinations are considered when determining the member performance envelopes. This subroutine will create a maximum and minimum shear and moment envelope for each member.

Main Variables Accessed: SECTFORC, ACTLIST

INDIV:

calls: none
called by: PARSE

INDIV is a post-processing graphic subroutine that will display on the terminal screen member shear and moment diagrams. The member performance plot for any active load case or combination or envelope can be displayed. Also, load diagrams can be displayed. A question and answer sequence is used by the program to determine which members and which diagrams the user wishes to be displayed.

Main Variables Accessed: SECTFORC, MCASE, NMCASE, TALLY

APPENDIX C

LOGIC AND FLOW CHARTS

This Appendix contains several diagrams that illustrate the flow of execution through the program. A list of the main menu commands is also given. When specifying a main menu command, only the first four letters of the command are required. The flow chart for the subroutine BUIL shows the program flow within the subroutine. The flow chart for SOLVE shows the macro-flow of the program as different subroutines are executed. The reader is referred to Appendices A and B of this thesis for a more complete description of the individual commands and subroutines.

Main Menu Commands

MAIN MENU

- BUILd
- DIGItize
- STORies
- BAYS

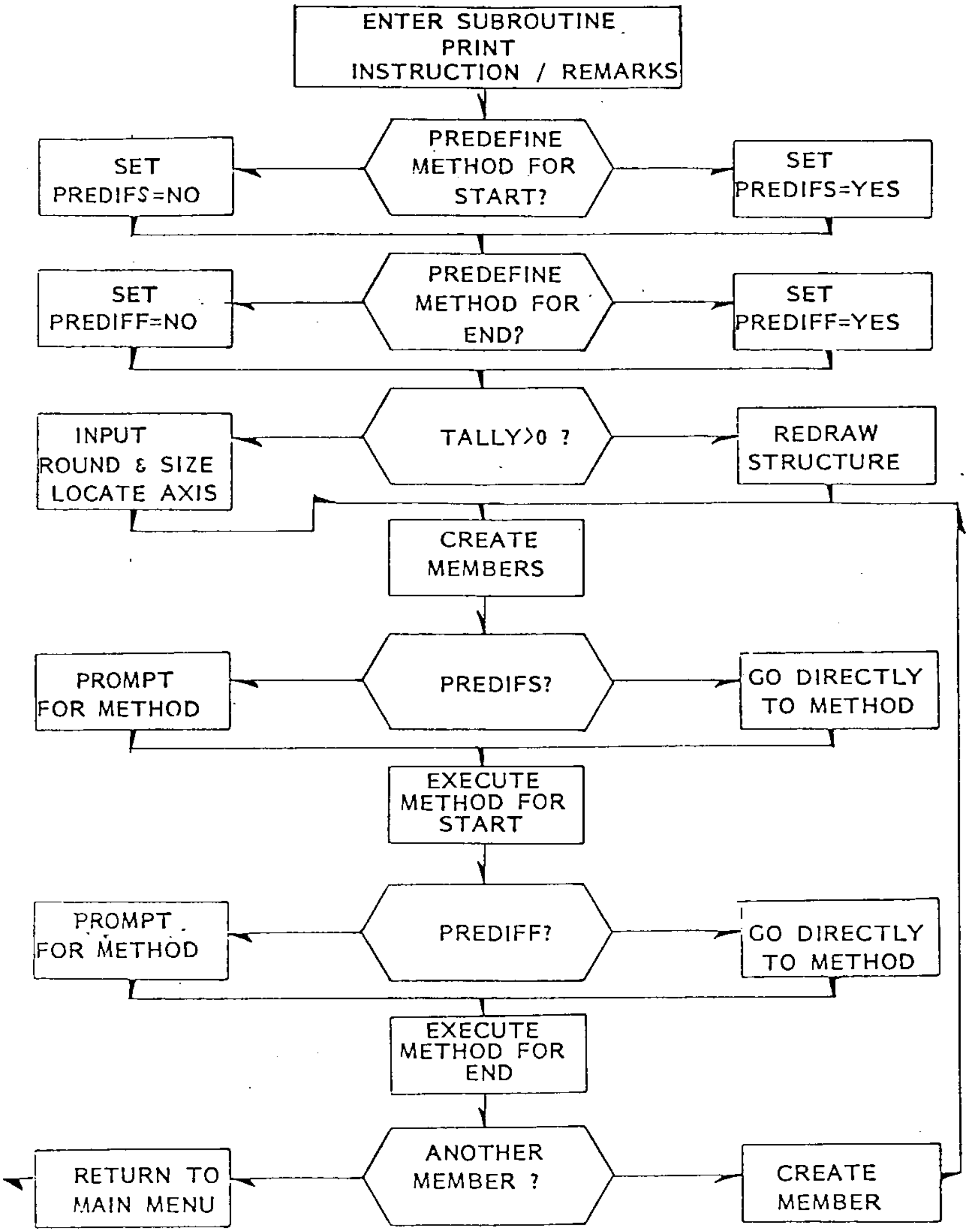
- PROPerties
- CONStants
- RELEase member
- SUPPort
- LOAD
- LCASe

- LIST
- ERASe
- CHANge
- DATA output
- RESUlt output
- ANSWers

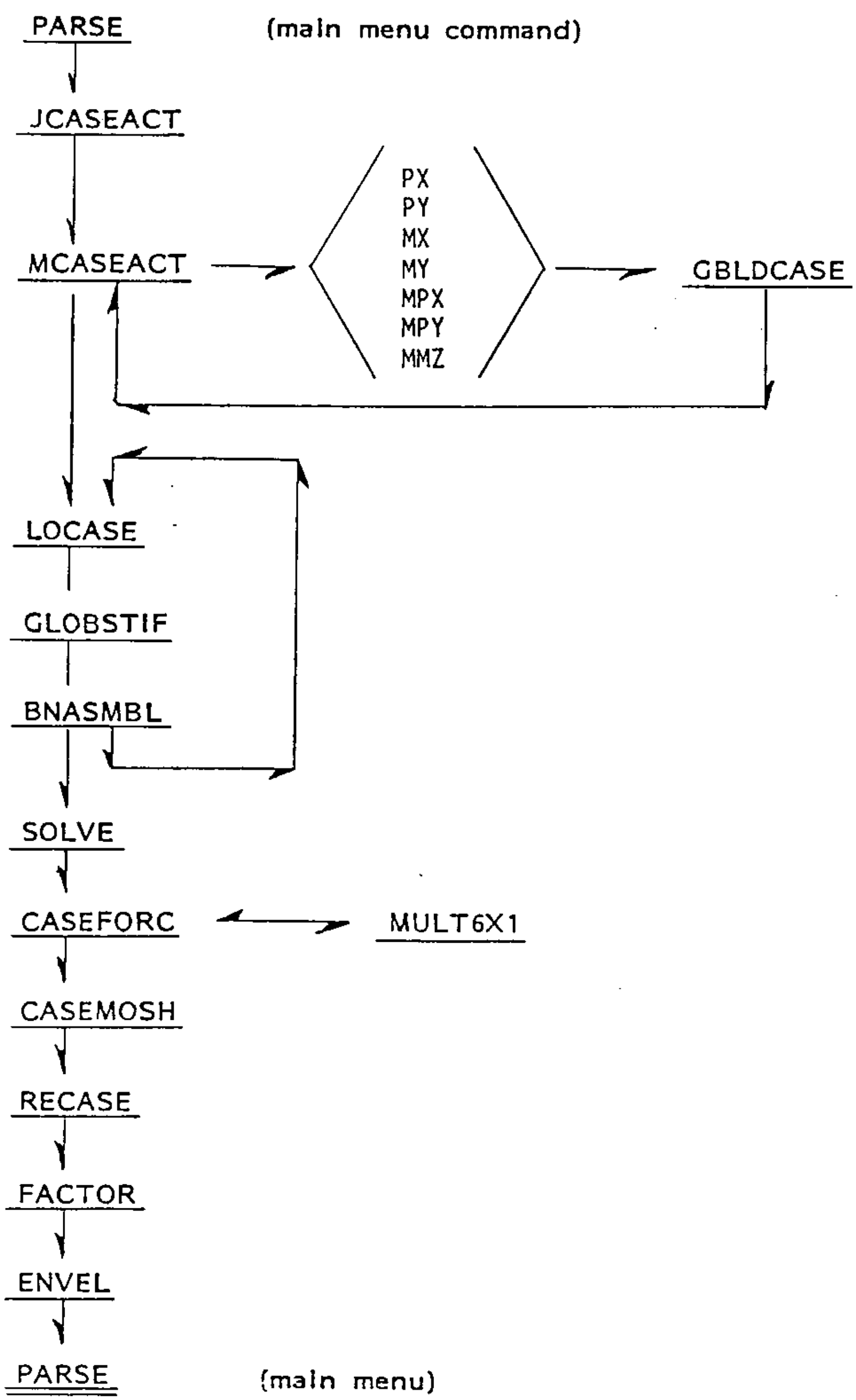
- REDRaw
- PLOT
- ZOOM
- SAVE
- RESTore
- SOLVe (see SOLVE flow chart)

- INDIvidual
- DEFOrmation
- HELP
- QUIT

BUIL Flow Chart



SOLVE Flow Chart



APPENDIX D

SAMPLE PROBLEMS

This Appendix illustrates the use of the FORTRAN program by solving 2 sample problems. Program generated graphics and printed output are also included.

The first problem is a simple hoist frame with a concentrated and a uniform load. The problem illustrates hinged supports and pinned member ends. Note that the start of member 3 is not released because the support is released from carrying moment.

Problem two is a typical two story one bay structural frame. This example illustrates the command STORIES and multiple load cases.

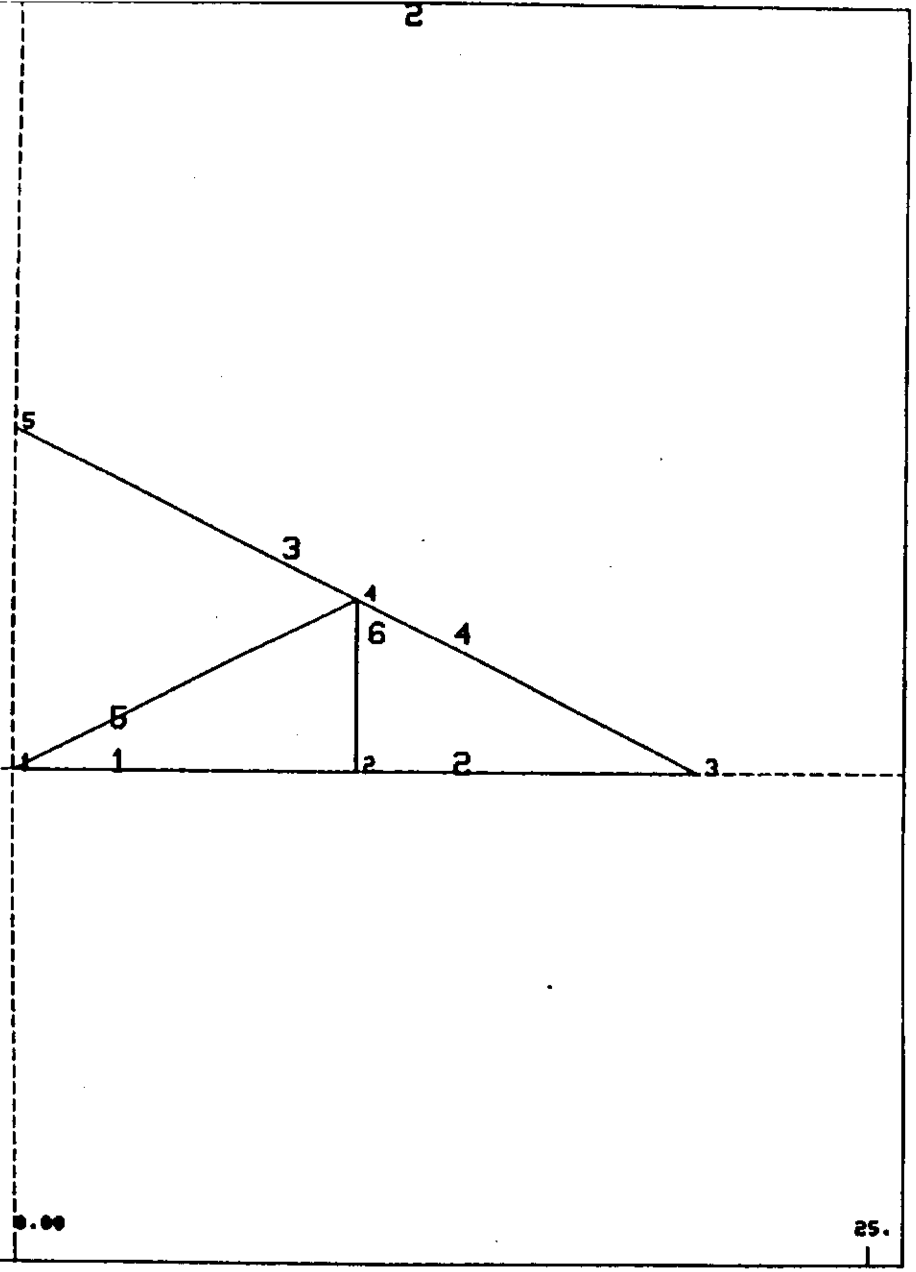
```
***********************************************************

*********  INTERACTIVE GRAPHIC STRUCTURAL ANALYSIS  ********

***********************************************************

>DO YOU NEED INSTRUCTIONS? Y/N
Y

This program will create and analyse a 2-dimensional
 plane frame structure in an interactive graphic mode

 -A TEKTRONIX 4014 or 4051 is needed to obtain graphics
    a digitizing tablet is optional for the 4014
 -Responses for YES and NO may be shortened to 1 letter
 -All commands must be at least 4 characters long
 -Remember to SWITCH or STORE your Load Case before you
    execute the SOLUTION phase
 -HELP sections are provided in all routines that ask
    for word commands
 -The user is referred to the USERS MANUAL for
    further documentation
ARE YOU ON A GRAPHICS TERMINAL? Y/N
Y
>ENTER YOUR TERMINAL TYPE AND OPTION -- one of the following:
  1) 4014 1      2) 4014 2      3) 4051 1
4014 2
> DO YOU HAVE A DIGITIZING TABLET? Y/N
Y
```

```
>>BUILD SECTION
  DO YOU NEED INSTRUCTIONS? Y/N
YES
**********************************************
READY TO BEGIN:
ALL LENGTHS WILL BE ROUNDED
 TO THE NEAREST INCRIMENT THAT YOU SPECIFY
INPUT THE ROUNDING INCRIMENT IN FEET
5.0
**********************************************
NODES WILL BE NUMBERED IN THE ORDER CREATED
MEMBERS ON EACH NODE WILL BE NUMBERED IN
 THE ORDER CREATED
**********************************************
INPUT THE LARGEST OVERALL DIMENSION
25.0
```

```
**********************************************
  THIS SECTION WILL ASSIST IN CREATING A
  2-D FRAME IN AN INTERACTIVE GRAPHIC MODE
  THE STRUCTURE CAN BE CREATED, IN PIECES,
  IN A COMBINATION OF METHODS
**********************************************
  WHEN SPECIFYING THE FIRST END F A MEMBER
  YOU CAN LOCATE IT BY:
   1. X,Y COORDINATE
   2. NODE NUMBER (THAT ALREADY HAS AN
       X,Y COORDINATE ASSOCIATED WITH IT)
   3.POINT TO IT WITH A LOCATE COMMAND
TO LOCATE THE ENDING POINT OF THE MEMBER:
   1. X,Y COORDINATE
   2. NODE NUMBER
   3. POINT TO IT WITH A LOCATE COMMAND
   4. SPECIFY AN ANGLE AND A LENGTH
   5. MOVE TO IT IN INCREMENTED STEPS
 IT WILL AUTOMATICALLY CALCULATE THE LENGTH
**********************************************
NOTE: DO YOU ALWAYS PLAN TO ENTER END
 ONE OF THE MEMBER BY THE SAME FORMAT?
NO
```

```
METHODS TO IDENTIFY END 2
 1=X,Y   2=NODE #   3=LOCATE IT
 4=ANGLE AND LENGTH 5=STEP TO IT
FOR METHOD 4, HORIZONTL TO THE RIGHT
 IS 0.0 DEGREES STRAIGHT UP IS +90.0
 DEGREES, NEGITIVE ANGLES ACCEPTED
FOR METHOD 5, USE THE KEYBOARD:
 U=UP
 D=DOWN
 R=RIGHT
 L=LEFT
SHIFT AND THE LETTER IS 5 TIMES THE AMOUNT
E=ENTER THIS POINT AS THE MEMBER END
**********************************************

 DO YOU PLAN TO ENTER THE SECOND END BY
 THE SAME METHOD??
NO
```

```
END1 1,2,3?
**XF,YF??
*END2
CHOOSE 1,2,3,4,5
**XS,YS??
ANOTHER MEMBER?
*END1
END1 1,2,3?
**NODEF #?
*END2
CHOOSE 1,2,3,4,5
**XS,YS??
ANOTHER MEMBER?
*END1
END1 1,2,3?
**XF,YF??
*END2
CHOOSE 1,2,3,4,5
**XS,YS??
ANOTHER MEMBER?
*END1
END1 1,2,3?
**NODEF #?
*END2
CHOOSE 1,2,3,4,5
**NODES #?
ANOTHER MEMBER?
*END1
END1 1,2,3?
**NODEF #?
*END2
CHOOSE 1,2,3,4,5
**NODES #?
ANOTHER MEMBER?
*END1
END1 1,2,3?
**NODEF #?
*END2
```

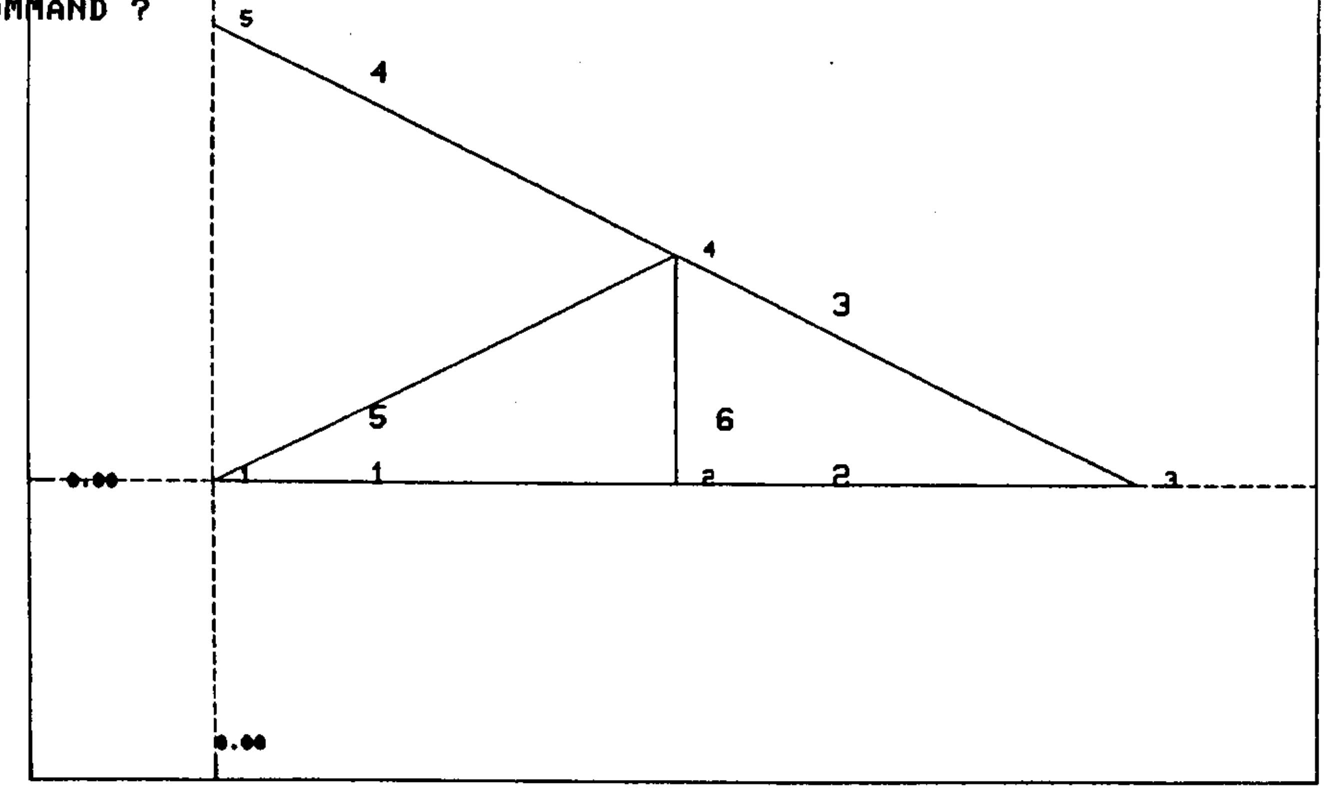


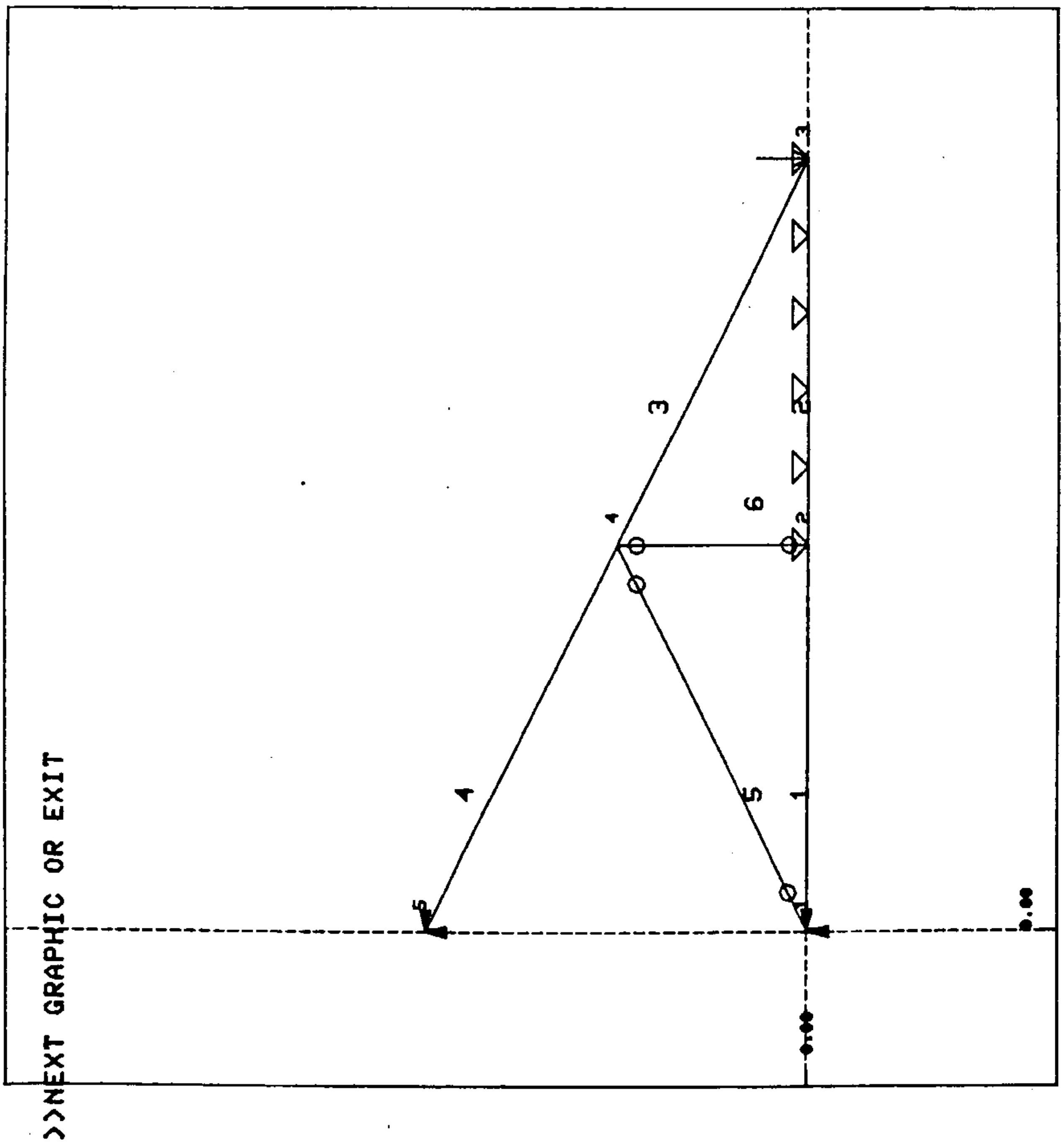
```
COMMAND ?
CONSTANTS
MEMBER CONSTANTS: E, ALPHA, DENS
MEMBER NUMBER>>
1
>>>CONSTANTS:
29000.
MEMBER NUMBER>>
0
COPY MEMBER PROPERTIES FROM # ?
1
  #        E          ALPHA          DENS
  1        29000.   .00000000         0.0000
COPY TO MEMBERS>>> START,END,INC
2 6 1
MEMBER NUMBER>>
-1
COMMAND ?
PROPERTIES
MEMBER PROPERTIES: Ax, Iz, Sx, Q
>>MEMBER NUMBER
1
PROPERTIES>>>
4.0  100.0
>>MEMBER NUMBER
0
COPY MEMBER PROPERTIES FROM # ?
1
  #       Ax       Iz        Sx      Q
  1    4.00    100.00      0.00       0.00
COPY TO MEMBERS>>>START,END,INC
2 6 1
>>MEMBER NUMBER
-1
COMMAND ?
```

```
SUPP
SUPPORT/SUPPORT RELEASE: TX,TY,RZ,TT,XR,YR,NO
JOINT NUMBER>>
1
RELEASE DIRECTION>>>
RZ
JOINT NUMBER>>
5
RELEASE DIRECTION>>>
RZ
JOINT NUMBER>>
-1
COMMAND ?
LOAD
LOAD SECTION
>> LOAD TYPE
MWFY
JOINT OR MEMBER NUMBER>>
2
LOAD MAGNITUDE,LOC,LOC,LOC
-2.0 0.0 1.0 0
>> LOAD TYPE
JPFY
JOINT OR MEMBER NUMBER>>
3
LOAD MAGNITUDE,LOC,LOC,LOC
-20. 0 0 0
>> LOAD TYPE
EXIT
COMMAND ?
```

```
LCASE
--- LOAD CASES SECTION ---
   OUT OF    1  LOAD CASES
       1       LOAD CASE IS THE WORKING CASE
LIST OF CURRENT LOAD CASES
     1   NONE GIVEN

LIST OF CURRENT LOAD COMBINATIONS

>>NEXT LOAD CASE OR EXIT
RENAME
>>> RENAME LOAD CASE # ?
1
 OLD NAME -- NONE GIVEN
ENTER NEW NAME -- 30 CHAR MAX
HOIST JOINT AND MEMBER LOADS
>>NEXT LOAD CASE OR EXIT
ACTIV
ACTIVATE LOAD CASES
 NOTE: ALL LOAD CASES ARE ACTIVE FOR THE SOLUTION
       ALL LOAD COMBINATIONS ARE ACTIVE FOR POST-PROCESS
>>THIS SECTION TO ACTIVATE ONLY CERTAIN INDEPENTANT LOAD
 CASES FOR THE POST-PROCESSING
LOAD CASE   1 ACTIVATE FOR POST-PROCESS ? Y/N
Y
THIS IS A PRINTOUT OF ACASES
  ACTIVE LOAD CASES   1  HOIST JOINT AND MEMBER LOADS
>>NEXT LOAD CASE OR EXIT
STORE
```

```
COMMAND ?
MREL
MEMBER END RELEASE: START OR END OR BOTH
MEMBER NUMBER ..
5
RELEASE :
BOTH
MEMBER NUMBER ..
6
RELEASE :
BOTH
MEMBER NUMBER ..
-1
COMMAND ?
```




>>NEXT GRAPHIC OR EXIT

```
DATA
STRUCTURE OUTPUT SECTION
> ENTER A   VAX  OUTPUT FILE NAME
OUTPUT1
> ENTER A TITLE FOR THIS STRUCTURE-- 30 CHARARACTERS MAX
HOIST FRAME
COMMAND ?
RESULTS
RESULT OUTPUT SECTION
ENTER A NAME FOR THE OUTPUT FILE --  8 CHARACTERS MAX
RESULT1
ENTER A JOB TITLE FOR THE OUTPUT -- 30 CHARACTERS MAX
HOIST FRAME
>>NEXT RESULT OR EXIT
DISPLACEMENTS
OUTPUTING JOINT DISPLACEMENTS
>>NEXT RESULT OR EXIT
SUPPORTS
OUTPUTING SUPPORT REACTIONS
>>NEXT RESULT OR EXIT
FORCES
OUTPUTING MEMBER END FORCES
>>NEXT RESULT OR EXIT
SECF
OUTPUTING MEMBER SECTION FORCES
>>NEXT RESULT OR EXIT
EXIT
COMMAND ?
```

>NEXT RESULT OR EXIT
DISPLACEMENTS
OUTPUTING JOINT DISPLACEMENTS

JOINT DISPLACEMENTS

JOINT #	LOAD CASE	TRANS X	TRANS Y	ROTATE Z
1				
	1	0.0000	0.0000	0.0004
2				
	1	-0.0562	-0.1299	-0.0041
3				
	1	-0.1123	-0.5852	-0.0028
4				
	1	0.0390	-0.1233	-0.0026
5				
	1	0.0000	0.0000	0.0003

>>NEXT RESULT OR EXIT
SUPPORTS
OUTPUTING SUPPORT REACTIONS

*** SUPPORT REACTIONS ***

JOINT #	LOAD CASE	FORCE X	FORCE Y	MOMENT Z
1				
	1	70.2574	6.1493	0.0000
5				
	1	-71.1476	34.5364	0.0000

>>NEXT RESULT OR EXIT
FORCES
OUTPUTING MEMBER END FORCES

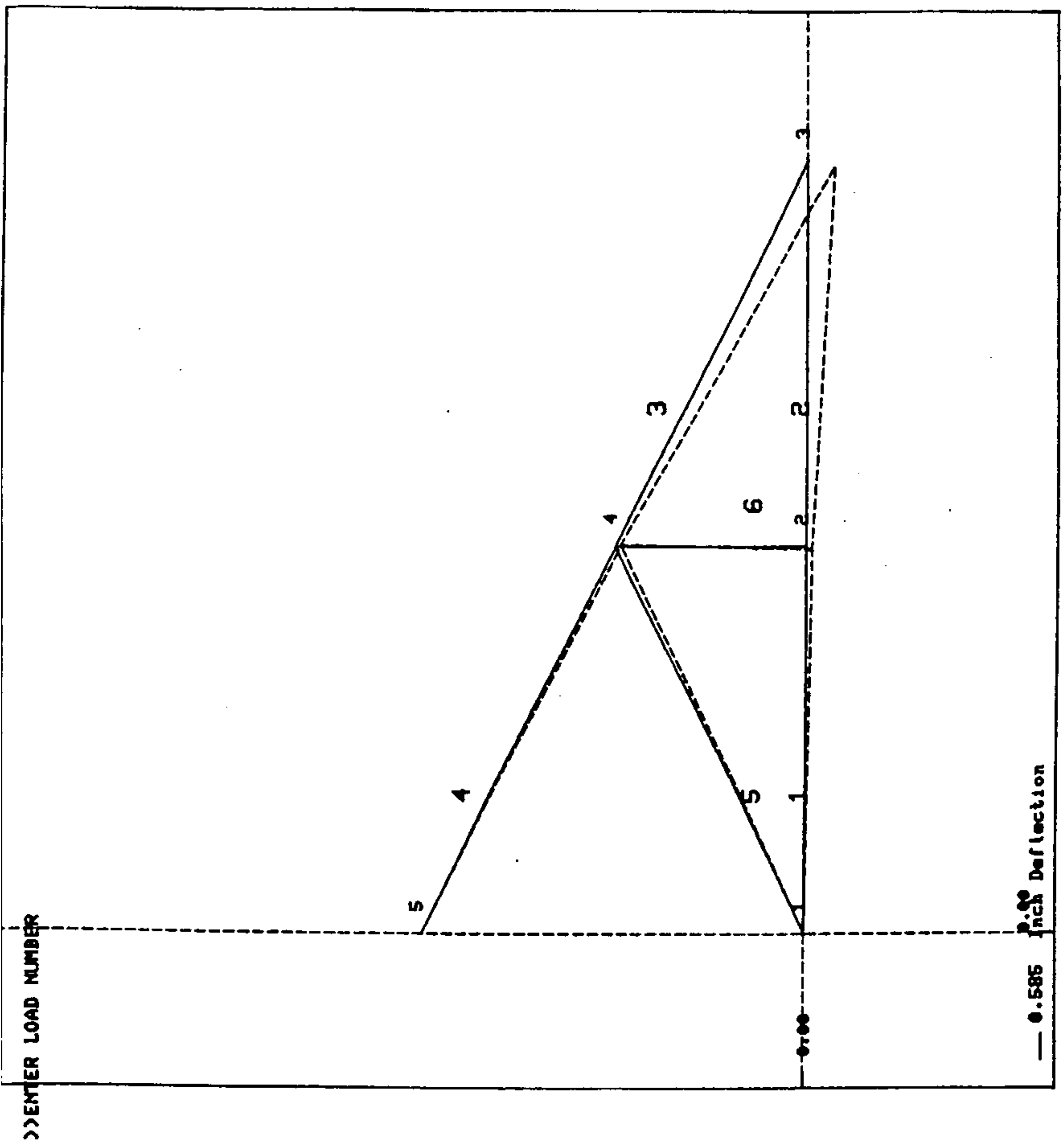
*** MEMBER END FORCES ***

MEMBER #	LOAD CASE	JOINT #	AXIAL	SHEAR	MOMENT
1					
	1	1.	54.30	-1.83	0.00
		2.	-54.30	1.83	-18.30
2					
	1	2.	54.30	10.87	18.30
		3.	-54.30	9.13	-9.58
3					
	1	4.	-61.59	1.77	10.21
		3.	61.59	-1.77	9.58
4					
	1	5.	-77.81	-0.91	0.00
		4.	77.81	0.91	-10.21
5					
	1	1.	17.55	0.00	0.00
		4.	-17.55	0.00	0.00
6					
	1	2.	-12.70	0.00	0.00
		4.	12.70	0.00	0.00

>>NEXT RESULT OR EXIT
SECF
OUTPUTING MEMBER SECTION FORCES

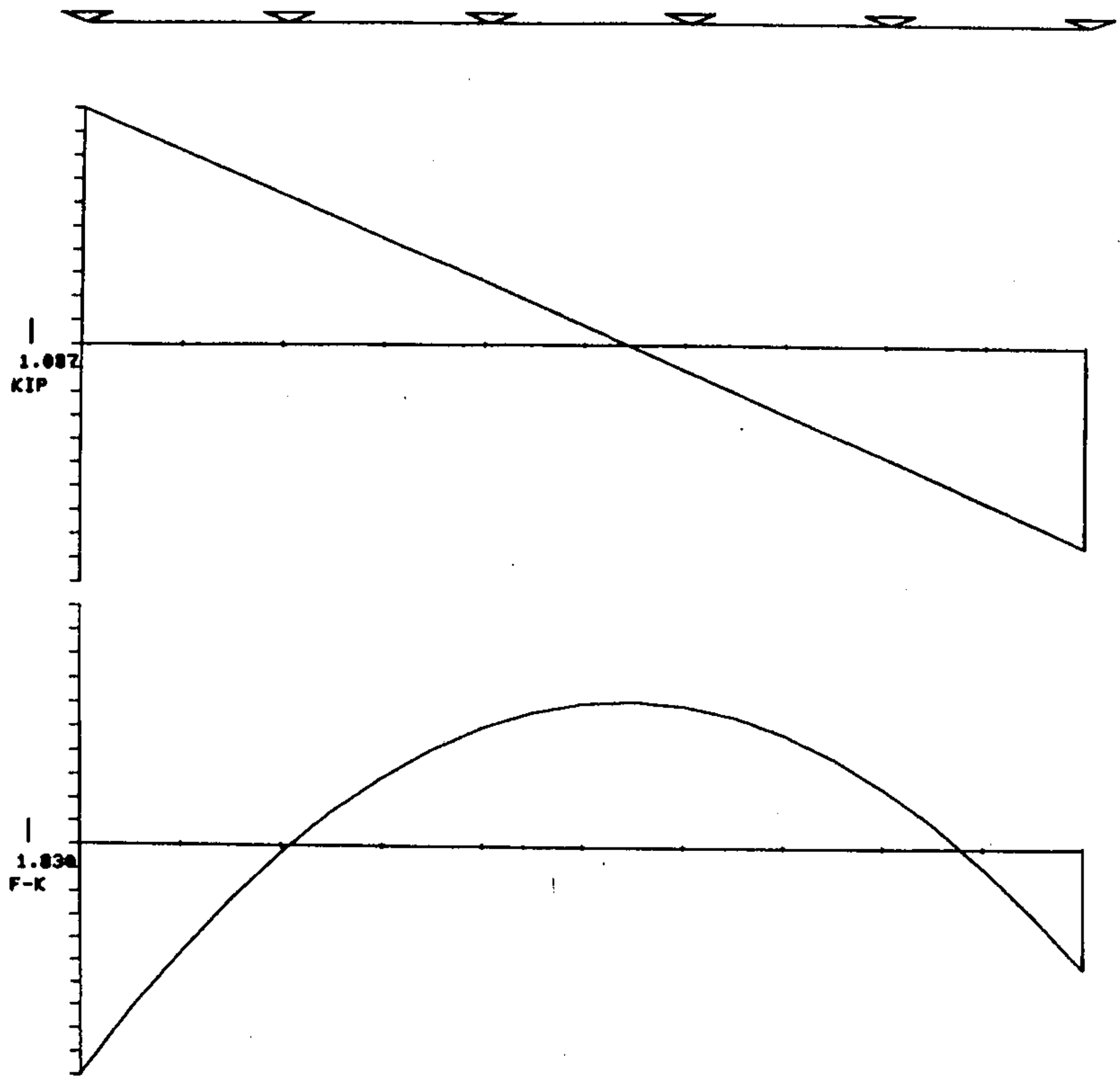
MEMBER SECTION FORCES

MEMBER #	LOAD CASE	SECTION	AXIAL	SHEAR	MOMENT
1	1				
		0.00		-1.83	0.00
		0.10		-1.83	-1.83
		0.20		-1.83	-3.66
		0.30		-1.83	-5.49
		0.40		-1.83	-7.32
		0.50		-1.83	-9.15
		0.60		-1.83	-10.98
		0.70		-1.83	-12.81
		0.80		-1.83	-14.64
		0.90		-1.83	-16.47
		1.00		-1.83	-18.30
2	1				
		0.00		10.87	-18.30
		0.10		8.87	-8.43
		0.20		6.87	-0.55
		0.30		4.87	5.32
		0.40		2.87	9.19
		0.50		0.87	11.06
		0.60		-1.13	10.93
		0.70		-3.13	8.81
		0.80		-5.13	4.68
		0.90		-7.13	-1.45
		1.00		-9.13	-9.58
3	1				
		0.00		1.77	-10.21
		0.10		1.77	-8.23
		0.20		1.77	-6.25
		0.30		1.77	-4.27
		0.40		1.77	-2.29
		0.50		1.77	-0.31
		0.60		1.77	1.66
		0.70		1.77	3.64
		0.80		1.77	5.62
		0.90		1.77	7.60
		1.00		1.77	9.58
4					


>>ENTER LOAD NUMBER
0.585
Inch Deflection
1
2
3
4
5
6

>>ANOTHER MEMBER, OVERWRITE
OR ANOTHER LOAD CASE (M/O/L/NO)

MEMBER 2



SHEAR	MAG	LOC
ABS MAX	10.87	0.0
MAXIMUM	10.87	0.0
MINIMUM	-9.13	1.0

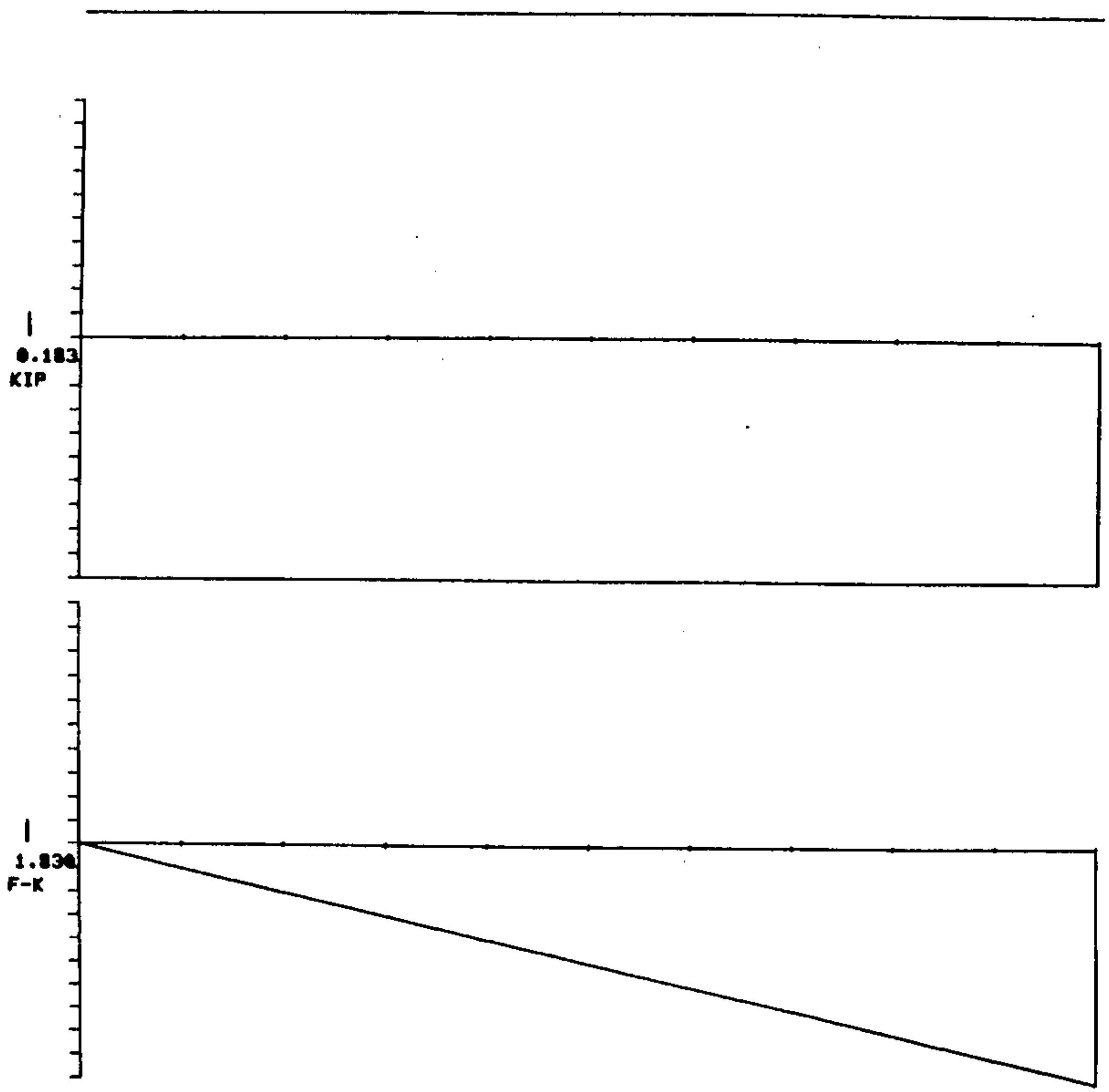
LOAD CASE 1

MOMENT	MAG	LOC
ABS MAX	-18.30	0.0
MAXIMUM	11.25	0.6
MINIMUM	-18.30	0.0

LOAD CASE 1

>>ANOTHER MEMBER, OVERWRITE
OR ANOTHER LOAD CASE (M/O/L/NO)

MEMBER 1



SHEAR	MAG	LOC
ABS MAX	-1.83	0.0
MAXIMUM	-1.83	0.0
MINIMUM	-1.83	0.0

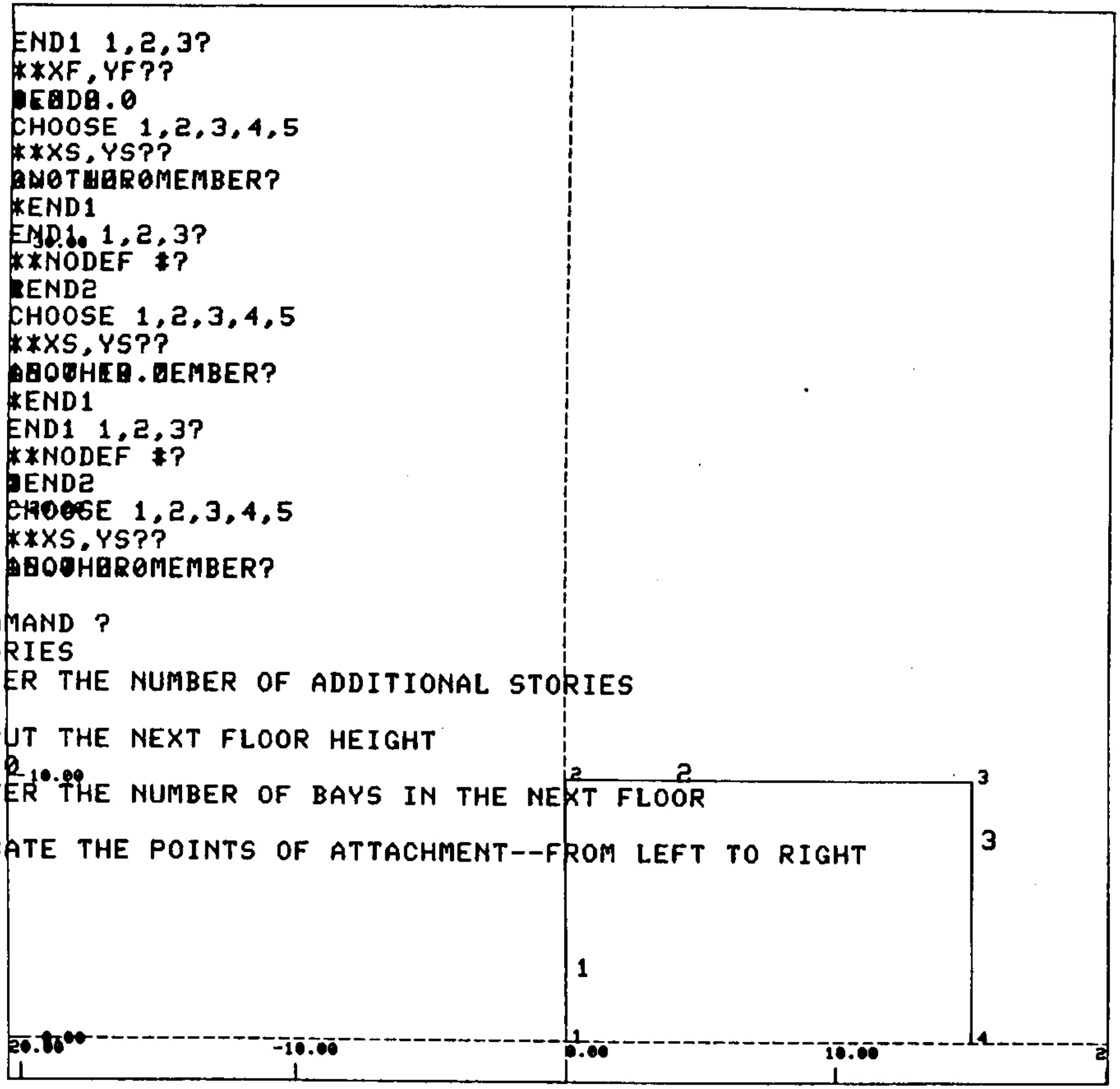
LOAD CASE 1

MOMENT	MAG	LOC
ABS MAX	-18.30	1.0
MAXIMUM	0.00	0.0
MINIMUM	-18.30	1.0

LOAD CASE 1

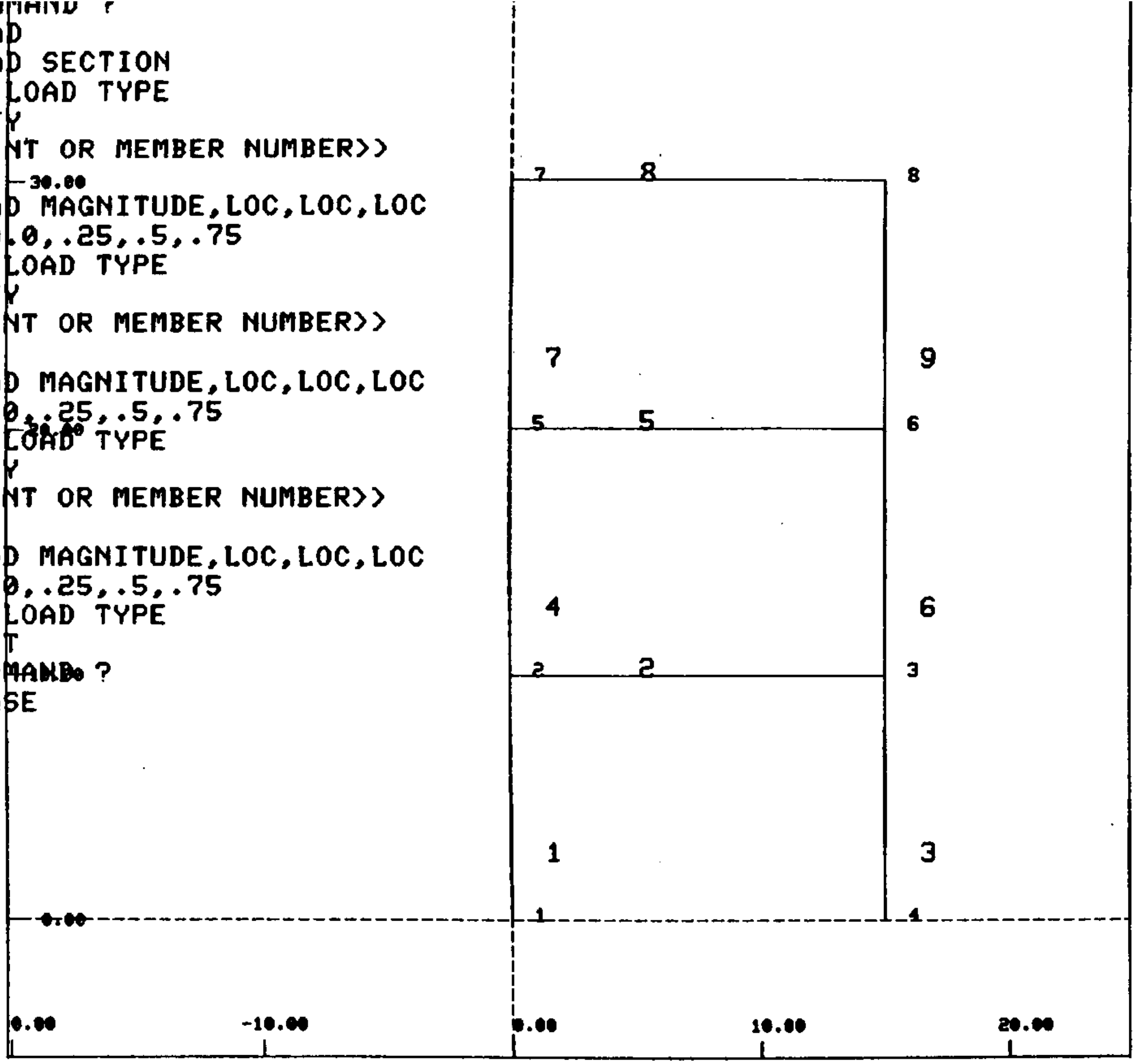
```
JOB TITILE:
*****************************************************************
*                 SAMPLE PROBLEM - HOIST FRAME                  *
*****************************************************************

NUMBER OF MEMBERS      ->   6
NUMBER OF JOINTS       ->   5
NUMBER OF LOAD CASES   ->   1
NUMBER OF LOAD COMB    ->   0

NODE #______X___________Y__
   1      0.000       0.000
   2     10.000       0.000
   3     20.000       0.000
   4      0.000      10.000
   5     10.000       5.000

MEMBER #___BEGIN____END____LENGTH
   1          1.      2.     120.00
   2          2.      3.     120.00
   3          4.      5.     134.16
   4          5.      3.     134.16
   5          1.      5.     134.16
   6          2.      5.      60.00

SUPPORT JOINT #_____FIXED
        1          TX TY
        4          TX TY

MEMBER #____RELEASES

MEMBER #___ E ksi ___
   1       29000.
   2       29000.
   3       29000.
   4       29000.
   5       29000.
   6       29000.

MEMBER #___ AREA ___ Iz ___
   1       4.00    100.00
   2       4.00    100.00
   3       4.00    100.00
   4       4.00    100.00
   5       4.00    100.00
   6       4.00    100.00

********************** LOAD CASE #  1 *********************

  LOAD CASE TITLE ===> JOINT LOAD AT JOINT 3
JOINT #___DIRECTION___MAGNITUDE
   3          FY        -20.000

MEMBER #___TYPE_____DIR____MAGNITUDE__BEG___END

LOAD COMBINATION DATA

************ ACTIVE POST-PROCESSING LOAD CASES ************
   1      JOINT LOAD AT JOINT 3
  11      MAXIMUM ENVELOPE
  12      MINIMUM ENVELOPE
#
```

```
*******************************************************************

*********  INTERACTIVE GRAPHIC STRUCTURAL ANALYSIS  ********

*******************************************************************

>DO YOU NEED INSTRUCTIONS? Y/N
Y

This program will create and analyse a 2-dimensional
 plane frame structure in an interactive graphic mode

 -A TEKTRONIX 4014 or 4051 is needed to obtain graphics
    a digitizing tablet is optional for the 4014
 -Responses for YES and NO may be shortened to 1 letter
 -All commands must be at least 4 characters long
 -Remember to SWITCH or STORE your Load Case before you
    execute the SOLUTION phase
 -HELP sections are provided in all routines that ask
    for word commands
 -The user is referred to the USERS MANUAL for
    further documentation
ARE YOU ON A GRAPHICS TERMINAL? Y/N
Y
>ENTER YOUR TERMINAL TYPE AND OPTION -- one of the following:
  1) 4014 1      2) 4014 2      3) 4051 1
4014 2
> DO YOU HAVE A DIGITIZING TABLET? Y/N
Y
```




```
END1 1,2,3?
**XF,YF??
▮E▮D▮.0
CHOOSE 1,2,3,4,5
**XS,YS??
▮N0T▮▮R0MEMBER?
*END1
END1 1,2,3?
**NODEF #?
▮END2
CHOOSE 1,2,3,4,5
**XS,YS??
▮▮O▮HE▮.▮EMBER?
*END1
END1 1,2,3?
**NODEF #?
▮END2
CH▮▮▮SE 1,2,3,4,5
**XS,YS??
▮▮O▮H▮R0MEMBER?

COMMAND ?
STORIES
ENTER THE NUMBER OF ADDITIONAL STORIES
2
INPUT THE NEXT FLOOR HEIGHT
10.0
ENTER THE NUMBER OF BAYS IN THE NEXT FLOOR
1
LOCATE THE POINTS OF ATTACHMENT--FROM LEFT TO RIGHT
```

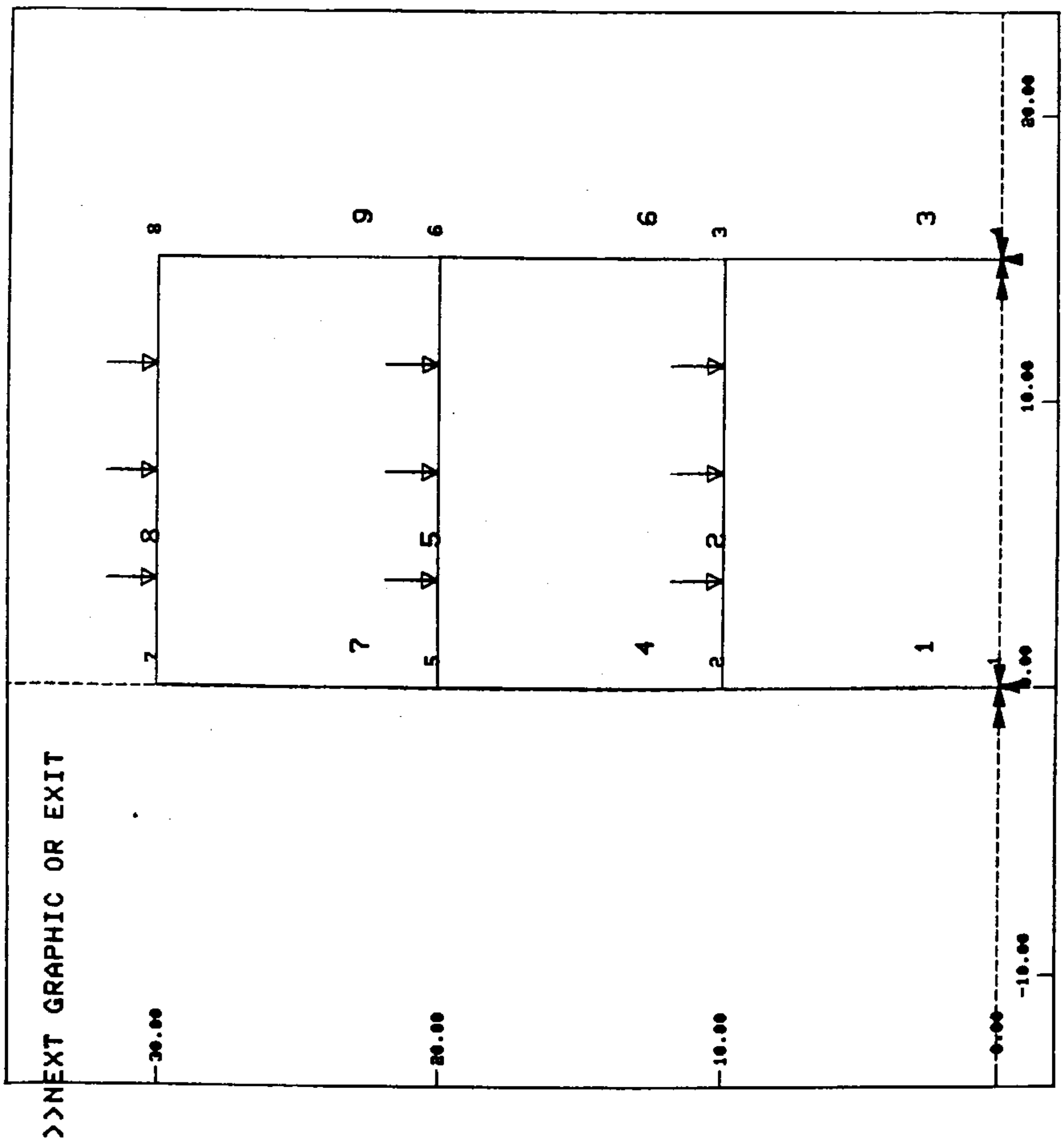
```
COMMAND ?
LOAD
LOAD SECTION
>> LOAD TYPE
MPFY
JOINT OR MEMBER NUMBER>>
2
LOAD MAGNITUDE,LOC,LOC,LOC
-10.0,.25,.5,.75
>> LOAD TYPE
MPFY
JOINT OR MEMBER NUMBER>>
5
LOAD MAGNITUDE,LOC,LOC,LOC
-8.0,.25,.5,.75
>> LOAD TYPE
MPFY
JOINT OR MEMBER NUMBER>>
8
LOAD MAGNITUDE,LOC,LOC,LOC
-8.0,.25,.5,.75
>> LOAD TYPE
EXIT
COMMAND ?
LCASE
```

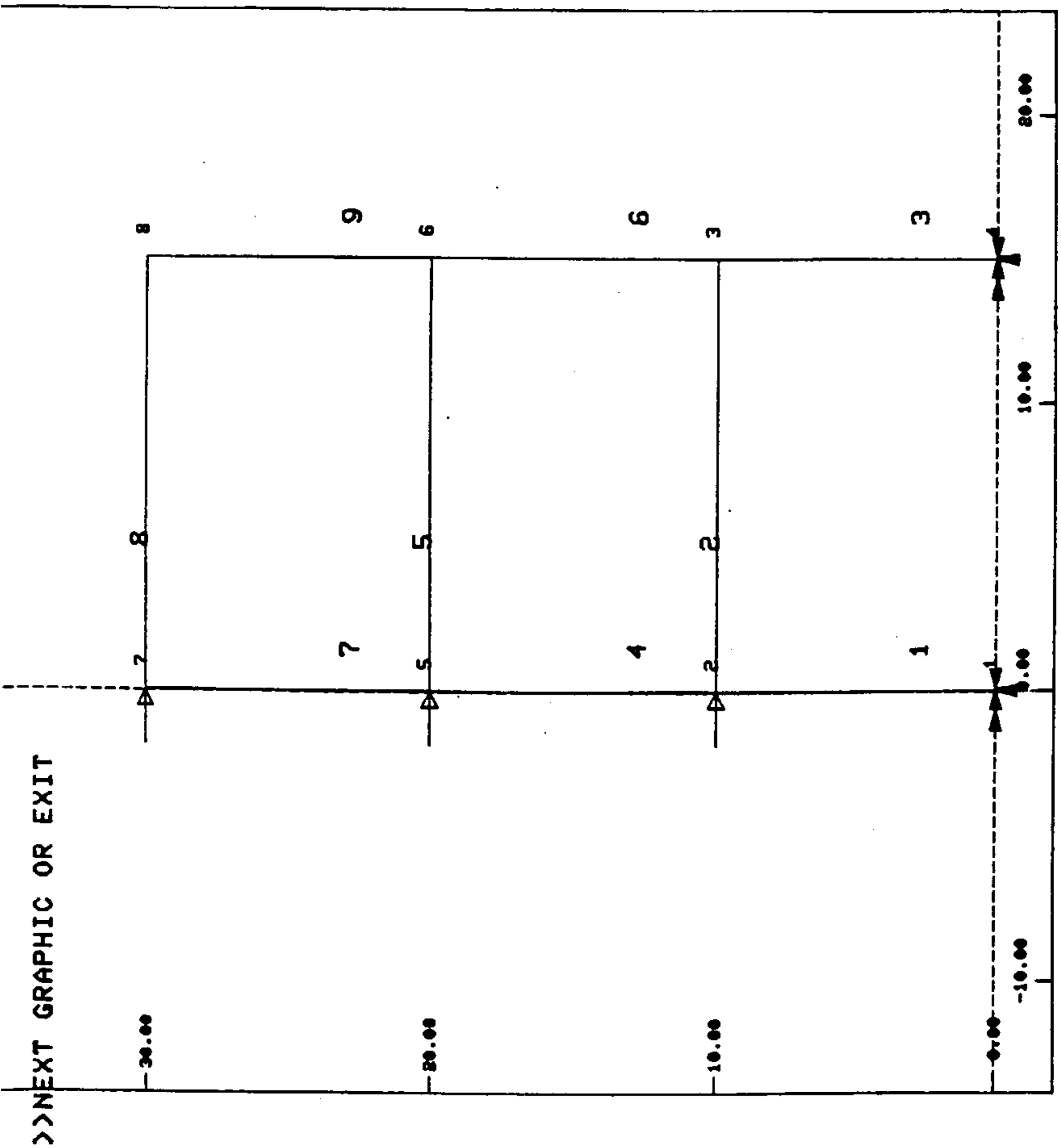


```
   OUT OF     1  LOAD CASES
       1         LOAD CASE IS THE WORKING CASE
LIST OF CURRENT LOAD CASES
    1   NONE GIVEN

LIST OF CURRENT LOAD COMBINATIONS

>>NEXT LOAD CASE OR EXIT
CREA
CREATE A NEW LOAD CASE
CASE NUMBER =>   2
ENTER A NAME FOR THIS CASE -- 30 CHAR MA
WIND LOAD
LOAD CASE   2 SUCCESSFULLY CREATED
NOTE -- OLD LOAD CASE STILL ACTIVATED
  LOAD CASE            1  STILL ACTIVE
>>NEXT LOAD CASE OR EXIT
SWIT
SWITCH TO LOAD CASE # ?
2
COMMAND ?
LOAD
LOAD SECTION
>> LOAD TYPE
JPFX
JOINT OR MEMBER NUMBER>>
2
LOAD MAGNITUDE,LOC,LOC,LOC
3.0,0,0,0
>> LOAD TYPE
JPFX
JOINT OR MEMBER NUMBER>>
5
LOAD MAGNITUDE,LOC,LOC,LOC
5.0,0,0,0
>> LOAD TYPE
7
LOAD MAGNITUDE,LOC,LOC,LOC
7.0,0,0,0
>> LOAD TYPE
EXIT
COMMAND ?
SUPPORTS
SUPPORT/SUPPORT RELEASE: TX,TY,RZ,TT,XR,Y
R,NO
JOINT NUMBER>>
1
RELEASE DIRECTION>>>
NO
JOINT NUMBER>>
4
RELEASE DIRECTION>>>
NO
JOINT NUMBER>>
-1
COMMAND ?
PLOT
```

```
LCASE
--- LOAD CASES SECTION ---
   OUT OF     2  LOAD CASES
        1        LOAD CASE IS THE WORKING CASE
LIST OF CURRENT LOAD CASES
     1    NONE GIVEN
     2    WIND LOAD

LIST OF CURRENT LOAD COMBINATIONS

>>NEXT LOAD CASE OR EXIT
COMBINE
COMBINE LOADING CASES
 CURRENTLY  2 LOAD CASES
 CURRENTLY  0 LOAD COMBINATION
                            1     2     3     4     5
LOAD COMBINATION => 1 TO 5
>>> ENTER LOAD COMBINATION #
1
CREATE A NEW LOAD COMBINATION
   ENTER A NAME FOR THIS LOAD COMBINATION
GRAVITY + WIND @ .75
LOAD COMBINATION   1 NOW ALL 0.0
 LOAD CASE   1 TIMES X.XX
.75
 LOAD CASE   2 TIMES X.XX
.75
>>NEXT LOAD CASE OR EXIT
ACTIVATE
ACTIVATE LOAD CASES
 NOTE: ALL LOAD CASES ARE ACTIVE FOR THE SOLUTION
       ALL LOAD COMBINATIONS ARE ACTIVE FOR POST-PROCESS
>>THIS SECTION TO ACTIVATE ONLY CERTAIN INDEPENTANT LOAD
 CASES FOR THE POST-PROCESSING
LOAD CASE   1 ACTIVATE FOR POST-PROCESS ? Y/N
Y
```

```
Y
THIS IS A PRINTOUT OF ACASES
  ACTIVE LOAD CASES    1  NONE GIVEN

  ACTIVE LOAD CASES    2  WIND LOAD

  ACTIVE LOAD CASES    6  GRAVITY + WIND @
 .75
>>NEXT LOAD CASE OR EXIT
EXIT
COMMAND ?
SOLVE
```


>>NEXT GRAPHIC OR EXIT
30.00
20.00
10.00
0.00
-10.00
10.00
20.00


>>NEXT GRAPHIC OR EXIT
30.00
20.00
10.00
0.00
-10.00
0.00
10.00
20.00

DISPLACEMENTS
OUTPUTING JOINT DISPLACEMENTS

JOINT DISPLACEMENTS

JOINT #	LOAD CASE	TRANS X	TRANS Y	ROTATE Z
1				
	1	0.0000	0.0000	0.0000
	2	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000
2				
	1	-0.0001	-0.0081	-0.0002
	2	0.0614	0.0034	-0.0004
	6	0.0460	-0.0035	-0.0005
3				
	1	0.0001	-0.0081	0.0002
	2	0.0610	-0.0034	-0.0004
	6	0.0459	-0.0086	-0.0001
4				
	1	0.0000	0.0000	0.0000
	2	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000
5				
	1	-0.0001	-0.0130	-0.0001
	2	0.1358	0.0053	-0.0003
	6	0.1018	-0.0058	-0.0003
6				
	1	0.0001	-0.0130	0.0001
	2	0.1352	-0.0053	-0.0003
	6	0.1015	-0.0137	-0.0002
7				
	1	0.0005	-0.0155	-0.0003
	2	0.1854	0.0058	-0.0002
	6	0.1394	-0.0073	-0.0003
8				
	1	-0.0005	-0.0155	0.0003
	2	0.1845	-0.0058	-0.0002
	6	0.1380	-0.0160	0.0000

>>NEXT RESULT OR EXIT
SUPPORTS
OUTPUTING SUPPORT REACTIONS

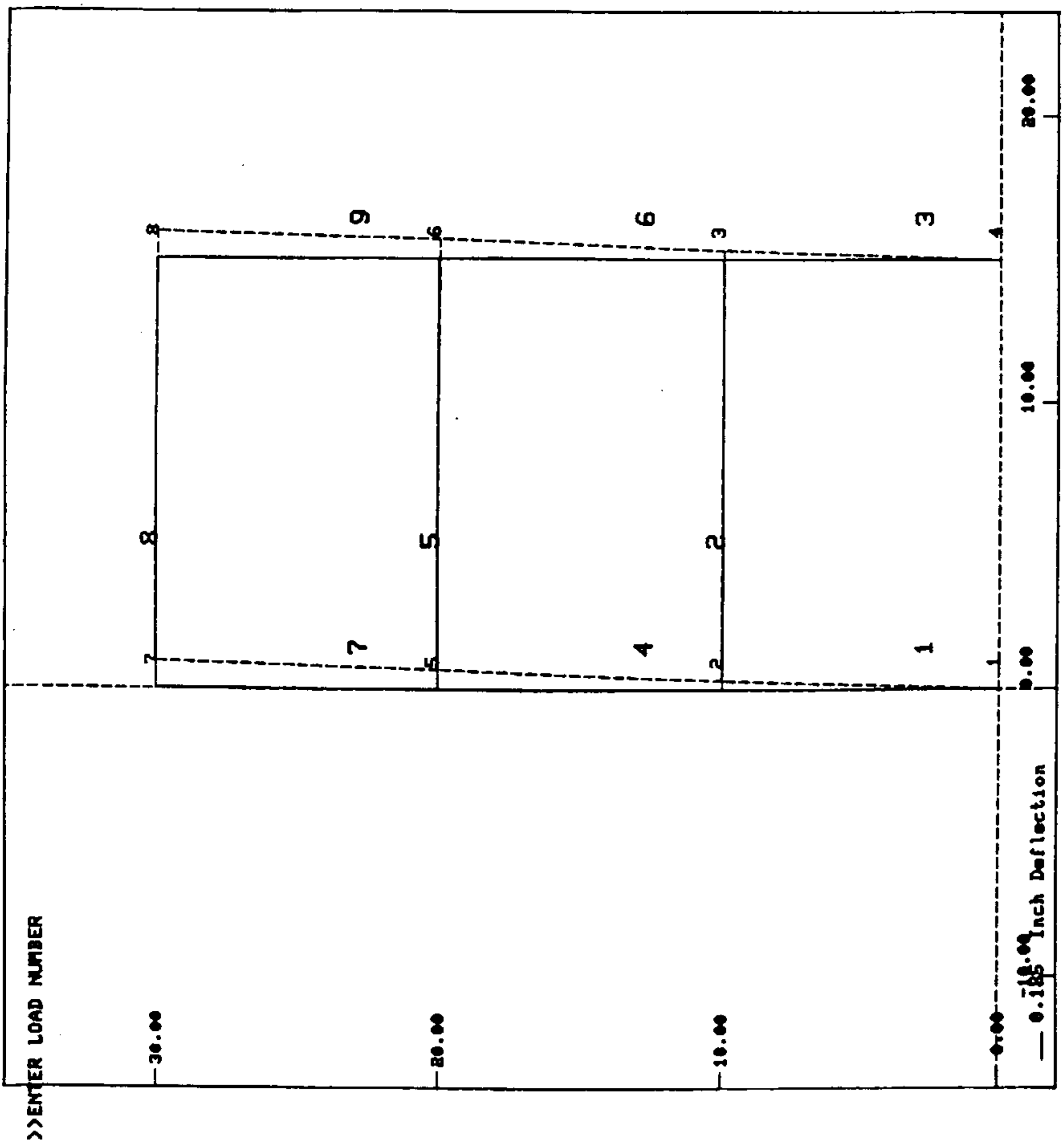
* SUPPORT REACTIONS *

JOINT #	LOAD CASE	FORCE X	FORCE Y	MOMENT Z
1				
	1	2.4512	39.0000	-8.2154
	2	-7.5211	-16.5928	45.6849
	6	-3.8024	16.8054	28.1021
4				
	1	-2.4512	39.0000	8.2154
	2	-7.4788	16.5928	45.4206
	6	-7.4475	41.6946	40.2270

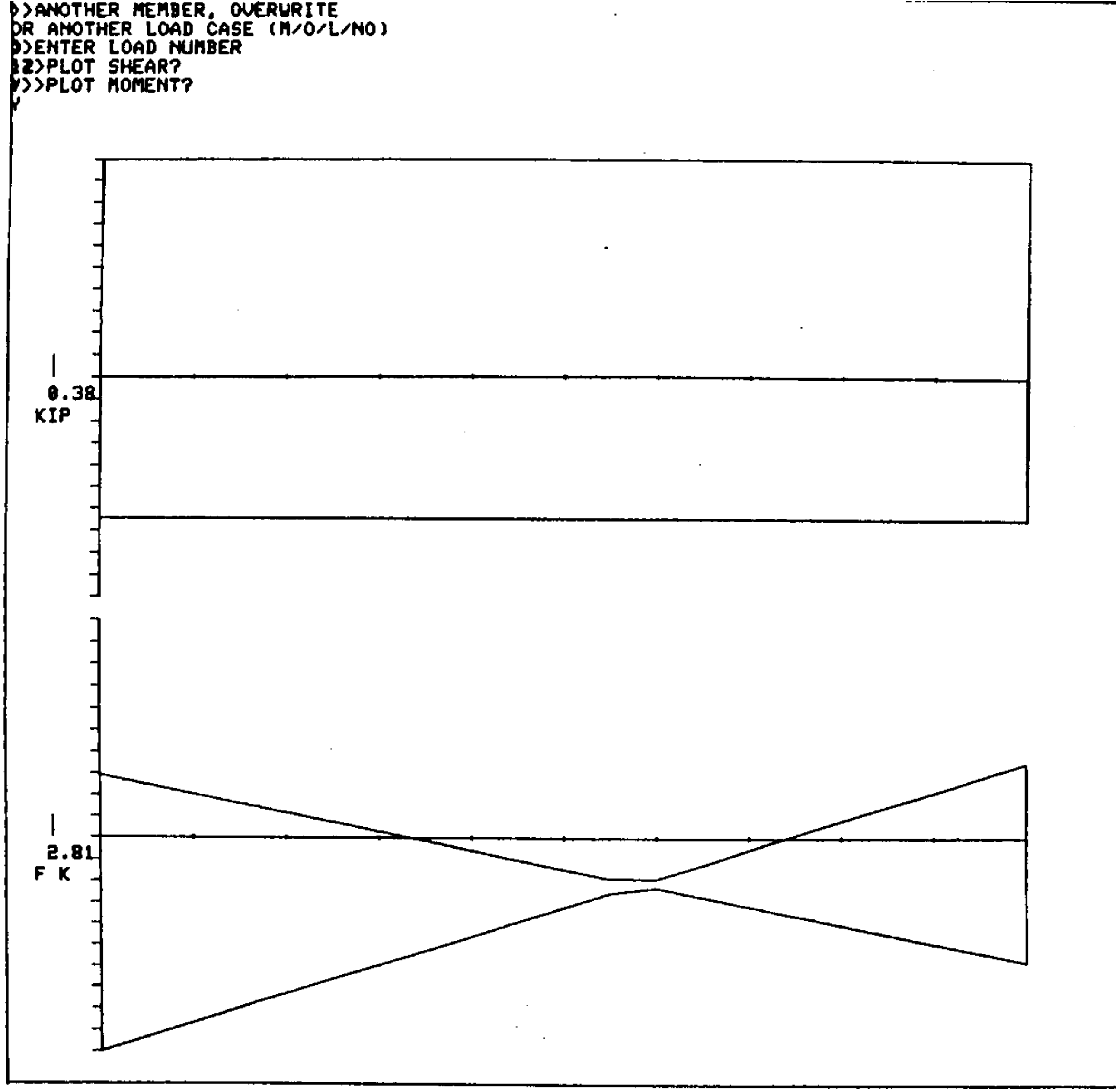
>>NEXT RESULT OR EXIT
FORCES
OUTPUTING MEMBER END FORCES

* MEMBER END FORCES *

MEMBER #	LOAD CASE	JOINT #	AXIAL	SHEAR	MOMENT
1					
	1	1.	39.00	-2.45	-8.22
		2.	-39.00	2.45	-16.30
	2	1.	-16.59	7.52	45.68
		2.	16.59	-7.52	29.53
	6	1.	16.81	3.80	28.10
		2.	-16.81	-3.80	9.92
2					
	1	2.	-1.07	15.00	36.10
		3.	1.07	15.00	-36.10
	2	2.	1.48	-7.77	-58.35
		3.	-1.48	7.77	-68.81
	6	2.	0.31	5.42	-16.69
		3.	-0.31	17.08	-70.73
3					
	1	4.	39.00	2.45	8.22
		3.	-39.00	-2.45	16.30
	2	4.	16.59	7.48	45.42
		3.	-16.59	-7.48	29.37
	6	4.	41.69	7.45	40.23
		3.	-41.69	-7.45	34.25
4					
	1	2.	24.00	-3.52	-19.80
		5.	-24.00	3.52	-15.42
	2	2.	-8.82	6.00	28.82
		5.	8.82	-6.00	31.18
	6	2.	11.38	1.86	6.76
		5.	-11.38	-1.86	11.82
5					
	1	5.	-0.57	12.00	32.56
		6.	0.57	12.00	-32.56
	2	5.	2.51	-6.09	-45.70
		6.	-2.51	6.09	-45.61
	6	5.	1.46	4.43	-9.85
		6.	-1.46	13.57	-58.63
6					
	1	3.	24.00	3.52	19.80
		6.	-24.00	-3.52	15.42
	2	3.	8.82	6.00	28.84
		6.	-8.82	-6.00	31.16
	6	3.	24.62	7.14	36.48
		6.	-24.62	-7.14	34.04
7					
	1	5.	12.00	-4.09	-17.14
		7.	-12.00	4.09	-83.76

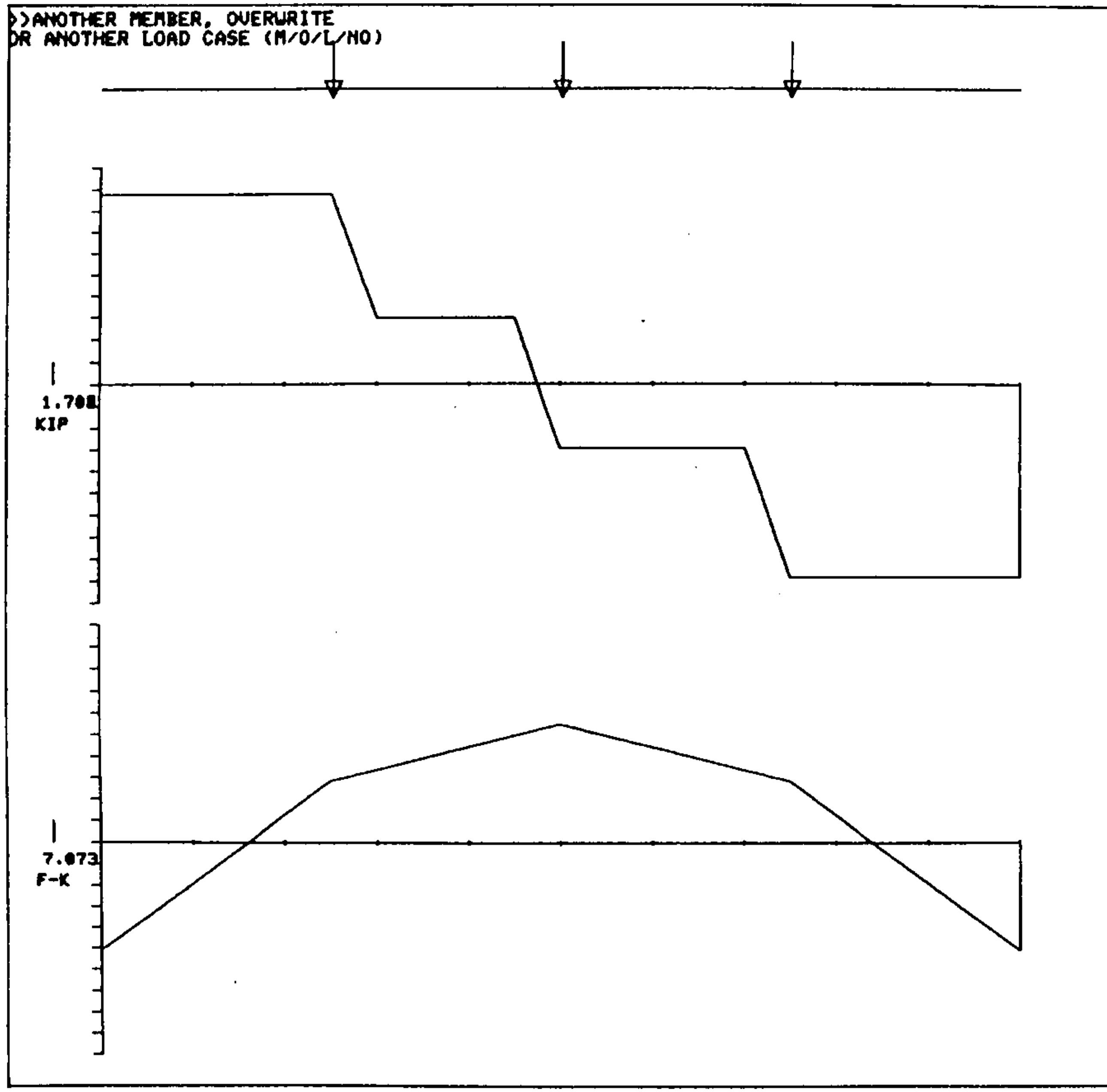

>>ENTER LOAD NUMBER
30.00
20.00
10.00
0.00
-10.00
0.00
10.00
20.00
0.185 Inch Deflection

MEMBER 1



SHEAR	MAG	LOC
ABS MAX	-3.80 / -2.45	0.0 / 0.0
MAXIMUM	-3.80 / -2.45	0.0 / 0.0
MINIMUM	-3.80 / -2.45	0.0 / 0.0
LOAD CASE	11 / 12	

MOMENT	MAG	LOC
ABS MAX	-8.92 / -28.10	1.0 / 0.0
MAXIMUM	-8.92 / -6.45	1.0 / 0.6
MINIMUM	-5.29 / -28.10	0.6 / 0.0
LOAD CASE	11 / 12	



MEMBER 2

SHEAR	MAG	LOC
ABS MAX	15.00	0.0
MAXIMUM	15.00	0.0
MINIMUM	-15.00	0.8

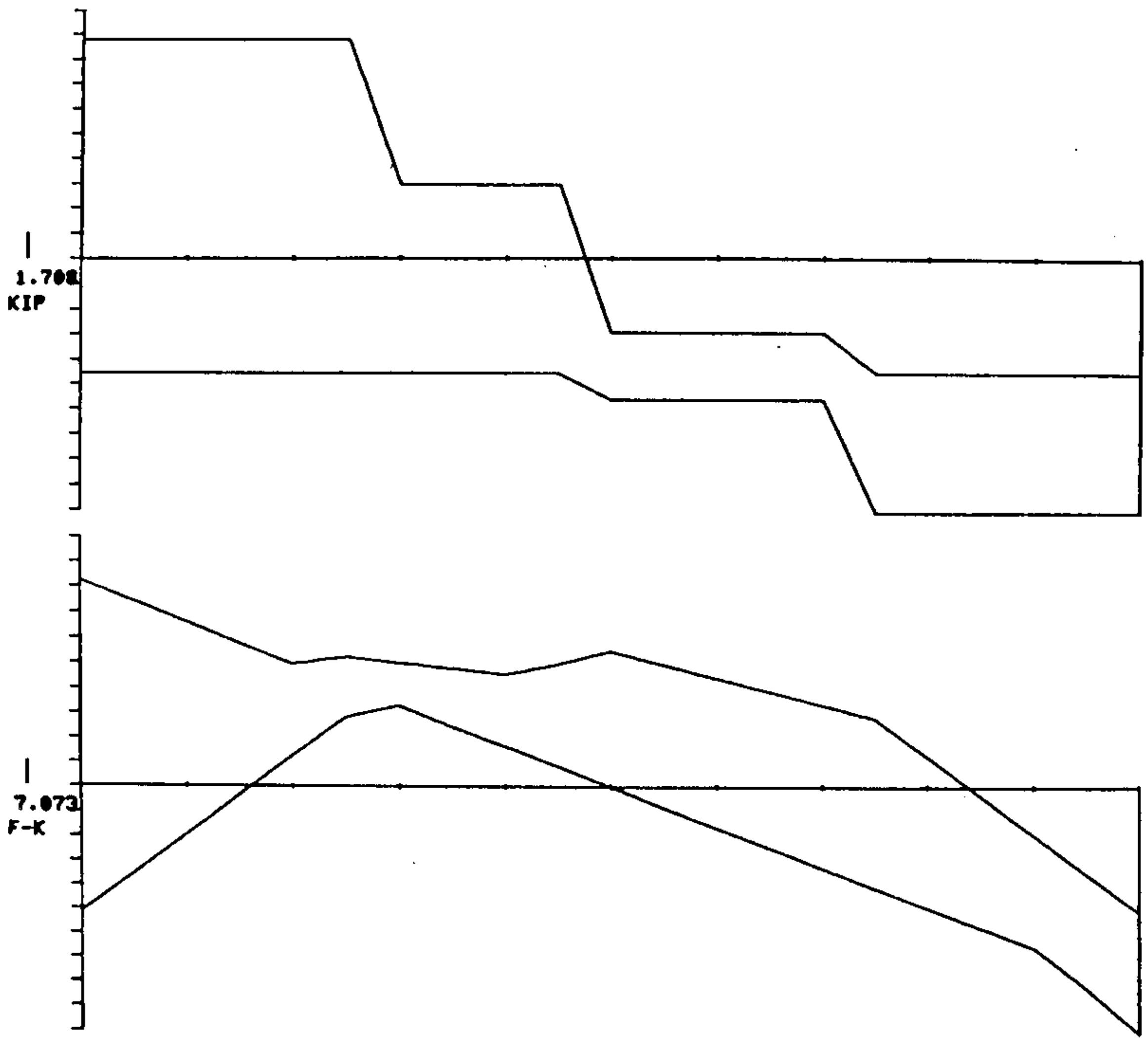
LOAD CASE 1

MOMENT	MAG	LOC
ABS MAX	38.90	0.5
MAXIMUM	38.90	0.5
MINIMUM	-36.10	1.0

LOAD CASE 1

MEMBER 2

```
>>ANOTHER MEMBER, OVERWRITE
OR ANOTHER LOAD CASE (M/O/L/NO)
>>ENTER LOAD NUMBER
12>PLOT SHEAR?
Y>>PLOT MOMENT?
Y
```



SHEAR	MAG	LOC
ABS MAX	15.00 / -17.08	0.0 / 0.0
MAXIMUM	15.00 / 7.77	0.0 / 0.0
MINIMUM	-7.77 / -17.08	0.0 / 0.0
LOAD CASE	11 / 12	

MOMENT	MAG	LOC
ABS MAX	58.35 / -70.73	0.0 / 1.0
MAXIMUM	58.35 / 23.38	0.0 / 0.3
MINIMUM	-36.19 / -70.73	1.0 / 1.0
LOAD CASE	11 / 12	

```
JOB TITILE:
******************************************************************
*               SAMPLE PROBLEM -3 STORY FRAME                    *
******************************************************************

NUMBER OF MEMBERS       => 9
NUMBER OF JOINTS        => 8
NUMBER OF LOAD CASES    => 2
NUMBER OF LOAD COMB     => 1

NODE #______X___________Y__
   1       0.000      0.000
   2       0.000     10.000
   3      15.000     10.000
   4      15.000      0.000
   5       0.000     20.000
   6      15.000     20.000
   7       0.000     30.000
   8      15.000     30.000

MEMBER #___BEGIN____END____LENGTH
   1          1.     2.     120.00
   2          2.     3.     180.00
   3          4.     3.     120.00
   4          2.     5.     120.00
   5          5.     6.     180.00
   6          3.     6.     120.00
   7          5.     7.     120.00
   8          7.     8.     180.00
   9          6.     8.     120.00

SUPPORT JOINT #_____FIXED
          1        TX TY MZ
          4        TX TY MZ

MEMBER #____RELEASES

MEMBER #___ E ksi ___
   1       29000.
   2       29000.
   3       29000.
   4       29000.
   5       29000.
   6       29000.
   7       29000.
   8       29000.
   9       29000.

MEMBER #___ AREA ___ Ix ___
   1      20.00   1000.00
   2      25.00   2000.00
   3      20.00   1000.00
   4      20.00   1000.00
   5      25.00   2000.00
   6      20.00   1000.00
   7      20.00   1000.00
   8      25.00   2000.00
   9      20.00   1000.00
********************** LOAD CASE #  1 **********************
 LOAD CASE TITLE ===> NONE GIVEN
JOINT #___DIRECTION___MAGNITUDE

MEMBER #___TYPE_____DIR____MAGNITUDE__BEG___END
   2      CONCEN    FY     -10.000  0.2500  0.0000
   2      CONCEN    FY     -10.000  0.5000  0.0000
   2      CONCEN    FY     -10.000  0.7500  0.0000
   5      CONCEN    FY      -8.000  0.2500  0.0000
   5      CONCEN    FY      -8.000  0.5000  0.0000
   5      CONCEN    FY      -8.000  0.7500  0.0000
   8      CONCEN    FY      -8.000  0.2500  0.0000
   8      CONCEN    FY      -8.000  0.5000  0.0000
   8      CONCEN    FY      -8.000  0.7500  0.0000
********************** LOAD CASE #  2 **********************

 LOAD CASE TITLE ===> WIND LOAD
JOINT #___DIRECTION___MAGNITUDE
   2         FX          3.000
   5         FX          5.000
   7         FX          7.000

MEMBER #___TYPE_____DIR____MAGNITUDE__BEG___END

LOAD COMBINATION DATA

          CASE # =>  1      2      3      4      5
 COMB NAME
   6  GRAVITY + WIND @ .75
                   0.750  0.750

************ ACTIVE POST-PROCESSING LOAD CASES ************
   1     NONE GIVEN
   6     GRAVITY + WIND @ .75
  11     MAXIMUM ENVELOPE
  12     MINIMUM ENVELOPE
$
```

APPENDIX E

PROGRAM LISTING

This appendix is the listing of the actual FORTRAN source code. The program is organized into six subdirectories. These subdirectories cover the areas of: structural data setup, structural data printing, graphics, assembling the matrix equations, problem solution, and post-processing. The titles given to these six subdirectories are, SETUP, LOOK, PICTURE, ASSEMBLE, SOLUTION, and POST, respectively. Within this appendix the subdirectories are ordered alphabetically. Within each subdirectory the subroutines are also ordered alphabetically.

****

```
C     THIS IS THE SUBROUTINE THAT WILL ASSEMBLE
C      THE BANDED STIFFNESS MATRIX
        SUBROUTINE BNASMBL(NI,NJ,SM)
        COMMON /ASSEMB/ BMAX,BASS
        REAL BASS(120,120)
        INTEGER BMAX
        REAL SM(6,6)
        INTEGER KK(6),I,J,K,NI,NJ,IC,Q,IR
C    PLACE DOF NUMBER IN KK
         KK(3)=3*NI
         KK(2)=KK(3)-1
         KK(1)=KK(3)-2
         KK(6)=3*NJ
         KK(5)=KK(6)-1
         KK(4)=KK(6)-2
C    ASSEMBLE GLOBAL BANDED MATRIX
C    ACCORDING TO KK
C     GLOBAL ASSEMBLED BANDED SM
        DO 20 J=1,6
         IR=KK(J)
          DO 30 K=1,6
           IF(KK(K).LT.IR)GOTO 30
            IC=KK(K)-IR+1
            BASS(IR,IC)=BASS(IR,IC)+SM(J,K)
30        CONTINUE
20      CONTINUE
        RETURN
        END


        SUBROUTINE GBLDCASE(EJL,NI,NJ,ACT,L,XI,XJ,YI,YJ,I,J)
C       THIS WILL TAKE THE EQUAVELENT JOINT LOADS FORM MCASEACT
C       AND TURN THEM INTO GLOBAL LOADS AT THE JOINTS
C       THEN ADD THESE LOADS TO THE APPROPRIATE ACTION VECTOR LOCATION
        REAL XI,XJ,YI,YJ,EJL(6),ACT(10,120),F(6),L
        INTEGER I,J,K,KK(6)
C       GET THE SIN AND COSINE
        C=(XJ-XI)/L
        S=(YJ-YI)/L
C      TURN THE LOCAL EJL INTO GLOBAL JOINT LOADS

        F(1)=C*EJL(1)-S*EJL(2)
        F(2)=S*EJL(1)+C*EJL(2)
        F(3)=EJL(3)
        F(4)=C*EJL(4)-S*EJL(5)
        F(5)=S*EJL(4)+C*EJL(5)
        F(6)=EJL(6)
C     ASSEMBLE INTO ACT ACCORDING TO 3*NODE
        KK(1)=3*NI-2
        KK(2)=3*NI-1
        KK(3)=3*NI
        KK(4)=3*NJ-2
        KK(5)=3*NJ-1
        KK(6)=3*NJ
        DO 20 K=1,6
         ACT(J,KK(K))=ACT(J,KK(K))+F(K)
20      CONTINUE
        RETURN
        END
```

```
C     THIS ROUTINE TRANSFORMS ALL THE STIFFNESSES TO GLOBAL
         SUBROUTINE GLOBSTIF(SM,S,C)
         REAL SM(6,6),S,C
         REAL T(6,6),TT(6,6),STOR(6,6),TEM(6,6),TEM2(6,6)
         INTEGER I,J,K,M
C     ZERO T AND TT
         DO 20 I=1,6
          DO 30 J=1,6
           T(I,J)=0
           TT(I,J)=0
30       CONTINUE
20       CONTINUE
C     FILL T AND TT
         T(1,1)=C
         T(1,2)=S
         T(2,1)=-S
         T(2,2)=C
         T(3,3)=1
         T(4,4)=C
         T(4,5)=S
         T(5,4)=-S
         T(5,5)=C
         T(6,6)=1
C
         TT(1,1)=C
         TT(1,2)=-S
         TT(2,1)=S
         TT(2,2)=C
         TT(3,3)=1
         TT(4,4)=C
         TT(4,5)=-S
         TT(5,4)=S
         TT(5,5)=C
         TT(6,6)=1
C
C   DO THE MULTIPLY GLOBSTIF=Tt * SM *T
         DO 40 K=1,6
          DO 50 L=1,6
           STOR(K,L)=0
            DO 60 M=1,6
            STOR(K,L)= STOR(K,L)+SM(K,M)*T(M,L)
60          CONTINUE
         TEM(K,L)=STOR(K,L)
50       CONTINUE
40       CONTINUE
         DO 70 K=1,6
          DO 80 L=1,6
           TEM2(K,L)=0
            DO 90 M=1,6
             TEM2(K,L)=TEM2(K,L)+TT(K,M)*TEM(M,L)
90       CONTINUE
              SM(K,L)=TEM2(K,L)
80       CONTINUE
70       CONTINUE
C        ALL SM NOW IN GLOBAL STIFFNESS
C      GOTO ASSEMBLE TOTAL STIFFNESS
         RETURN
         END
```

```
C       THIS WILL ADD THE JOINT LOADS TO THE ACTION VECTOR 'ACT'
        SUBROUTINE JCASEACT(ACT)
        COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
        REAL MCASE(5,40,6),JCASE(5,40,3)
        REAL ACT(10,120)
        INTEGER CASES,NMCASE(5),NJCASE(5)
        INTEGER NI,J
        DO 1 J=1,CASES
          DO 10 I=1,NJCASE(J)
C        THIS IS FUNNY BUT TAKE THE JOINT NUMBER SUBTRACT 1, MULTIPLY BY 3
C        (FOR THE DOF) THEN ADD TO THIS TH DIR  # (1 OR 2 OR 3)
           NI=(JCASE(J,I,1)-1)*3 +JCASE(J,I,2)
           ACT(J,NI)=JCASE(J,I,3)+ACT(J,NI)
10      CONTINUE
1       CONTINUE
        RETURN
        END
```

```
C    THIS SUBROUTINE WILL CALCULATE THE LOCAL STIFFNESS OF A
C        MEMBER ACCORDING TO ITS MEMBER END RELEASES
C        CASE 1  BOTH ENDS FIXED
C        CASE 2  BOTH ENDS RELEASE MOMENT Z
C        CASE 3 END I RELEASED MOMENT Z
C        CASE 4 END J RELEASED MOMENT Z
         SUBROUTINE LOCASE(L,E,I,A,CASE,SM)
         REAL SM(6,6),L,E,I,A
         INTEGER CASE
         DO 2 K=1,6     ! THIS WILL ZERO 000 ALL 36 OF THE MATRIX LOCATIONS
                 DO 2 M=1,6
2                 SM(K,M)=0
         GOTO (10,20,30,40),CASE
10       SM(1,1)=A*E/L  !  THIS WILL FILL ONLY THE NEEDED LOCATIONS
         SM(1,4)=A*E/L*(-1)
         SM(2,2)=12*E*I/L**3
         SM(2,3)=6*E*I/L**2
         SM(2,5)=-1*SM(2,2)
         SM(2,6)=6*E*I/L**2
         SM(3,2)=SM(2,3)
         SM(3,3)=4*E*I/L
         SM(3,5)=-1*6*E*I/L**2
         SM(3,6)=2*E*I/L
         SM(4,1)=SM(1,4)
         SM(4,4)=SM(1,1)
         SM(5,2)=SM(2,5)
         SM(5,3)=SM(3,5)
         SM(5,5)=SM(2,2)
         SM(5,6)=SM(3,5)
         SM(6,2)=SM(2,6)
         SM(6,3)=SM(3,6)
         SM(6,5)=SM(5,6)
         SM(6,6)=SM(3,3)
         GOTO 200
20       SM(1,1)=A*E/L     !  ROWS AND COLUMNS 3 AND 6 ARE LEFT 000
         SM(1,4)=-SM(1,1)
         SM(4,1)=-SM(1,1)
         SM(4,4)=SM(1,1)
         GOTO 200
30       SM(1,1)=A*E/L  ! ROW AND COLUMN  3  IS LEFT 0000
         SM(1,4)=-SM(1,1)
         SM(2,2)=3*E*I/L**3
         SM(2,5)=-SM(2,2)
         SM(2,6)=3*E*I/L**2
         SM(4,1)=-SM(1,1)
         SM(4,4)=SM(1,1)
         SM(5,2)=SM(2,5)
         SM(5,5)=SM(2,2)
         SM(5,6)=-SM(2,6)
         SM(6,2)=SM(2,6)
         SM(6,5)=SM(5,6)
         SM(6,6)=3*E*I/L
         GOTO 200
40       SM(1,1)=A*E/L  !  ROW AND COLUMN 6 IS LEFT 000
         SM(1,4)=-SM(1,1)
         SM(2,2)=3*E*I/L**3
         SM(2,3)=3*E*I/L**2
         SM(2,5)=-SM(2,2)
         SM(3,2)=SM(2,3)
         SM(3,3)=3*E*I/L
```

```
      SM(3,5)=-SM(2,3)
      SM(4,1)=-SM(1,1)
      SM(4,4)=SM(1,1)
      SM(5,2)=SM(2,5)
      SM(5,3)=SM(3,5)
      SM(5,5)=SM(2,2)
C    THE STIFFNESS MATRIX IS NOW IN LOCAL
C     THIS SECTION LEFT IN FOR FUTURE DEBUGGING
200   I=I
C     PRINT*,'HERE IS THE MEMBER STIFFNESS IN LOCAL'
C     PRINT*,' '
C     DO 2002 I=1,6
C      PRINT 2100,(SM(I,J),J=1,6)
C2002 CONTINUE
C2100 FORMAT(6(F12.2,1X))
      RETURN
      END
```

```
      SUBROUTINE MCASEACT
      COMMON /GEOM/   MT,TALLY,NLOC,NT
      COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
      COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
      COMMON /RELEASE/MBREL,SREL,STALLY
      REAL MT(40,12),NLOC(40,2)
      INTEGER TALLY,NT
      REAL MCASE(5,40,6),JCASE(5,40,3)
      INTEGER CASES,NMCASE(5),NJCASE(5)
      REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
      REAL  FEMDIS(5,40,6)
      INTEGER MBREL(40),SREL(40),STALLY
      REAL MAG,LA,LB,L,EJL(6),XI,XJ,YI,YJ
      INTEGER K,NI,J,NJ,NK,TYPE,DIR,MEMNUM,CASE
C     THIS IS FOR MEMBER LOADS ONLY!!! JOINT LOADS ADD DIRECTLY
      DO 1 J=1,CASES
       DO 10 I=1,NMCASE(J)
              MEMNUM=MCASE(J,I,1)
              TYPE=MCASE(J,I,2)
              DIR =MCASE(J,I,3)
              MAG= MCASE(J,I,4)
              LA= MCASE(J,I,5)
              LB= MCASE(J,I,6)
              L=MT(MEMNUM,5)/12    ! NOW IN FEET
        NI=MT(MEMNUM,1)
        NJ=MT(MEMNUM,2)
         XI=NLOC(NI,1)
         YI=NLOC(NI,2)
         XJ=NLOC(NJ,1)
         YJ=NLOC(NJ,2)
      CASE=MBREL(MEMNUM)+1
      IF(TYPE.EQ.1)THEN
              IF (DIR.EQ.2) THEN
                CALL PY(MAG,L,LA,LB,EJL,CASE)
              ELSE
                CALL PX(MAG,EJL)
              END IF
      ELSE IF(TYPE.EQ.2)THEN
              IF(DIR.EQ.2) THEN
                IF(LA.EQ.0.AND.LB.EQ.0.OR.LA.EQ.0.AND.LB.EQ.1) THEN
                CALL MY(MAG,L,EJL,CASE)
              ELSE
                CALL MPY(MAG,L,LA,LB,EJL,CASE)
              END IF
              ELSE
              IF(LA.EQ.0.AND.LB.EQ.0.OR.LA.EQ.0.AND.LB.EQ.1) THEN
                CALL MX(MAG,L,EJL)
              ELSE
                CALL MPX     ! NOT YET AVAILABLE
              END IF
        END IF
      ELSE
        CALL MMZ       ! NOT YET AVAILABLE
      END IF
      DO 11 K=1,6
11     EMCASE(J,MEMNUM,K)=EMCASE(J,MEMNUM,K)+EJL(K)
      CALL GBLDCASE(EJL,NI,NJ,ACT,L,XI,XJ,YI,YJ,MEMNUM,J)
10    CONTINUE
1     CONTINUE
C    PUT EMCASE, FORCES FOR MOMENTS, IN F-K
```

```
      DO 12 J=1,CASES
       DO 13 M=1,TALLY
        EMCASE(J,M,3)=EMCASE(J,M,3)/12.0
        EMCASE(J,M,6)=EMCASE(J,M,6)/12.0
13     CONTINUE
12    CONTINUE
      RETURN
      END


C     THIS IS THE SUBROUTINE THAT WILL CALCULATE THE
C     EQU JOINT LOADS OF  A MEMBER THAT
C     HAS A UNIFORM LOAD OVER PART OF THE MEMBER
      SUBROUTINE MPY(MAG,L,A,B,EJL,CASE)
      REAL MAG,A,B,EJL(6),J,C,D,E,RL1,RL2,ML1,ML2
      REAL RR1,RR2,MR1,MR2,L
      INTEGER I,CASE
      A  =A*L
      B  =B*L
      C  =B-A
      D  =L-A
      E  =L-B
      EJL(1)=0
      EJL(2)=0
      GOTO (10,20,30,40),CASE
10    RL1=MAG*B/2.0 *  (2*(1-(B/L)**2)+(B/L)**3)
      RL2=MAG*A/2.0 *  (2*(1-(A/L)**2)+(A/L)**3)
      EJL(2)=RL1-RL2
      EJL(5)= MAG*C-EJL(2)
      ML1=MAG*B**2/12*(1+2*E/L+3*(E/L)**2)
      ML2=MAG*A**2/12*(1+2*D/L+3*(D/L)**2)
      EJL(3)=(ML1-ML2)*12    ! NOW IN IN-KIPS
      MR1=-MAG*B**2/12*B/L*(1+3*E/L)
      MR2=-MAG*A**2/12*A/L*(1+3*D/L)
      EJL(6)=(MR1-MR2)*12
      RETURN
20    EJL(2)=MAG*C/(2*L)*(2*E+D)
      EJL(3)=0
      EJL(5)=MAG*C-EJL(2)
      EJL(6)=0
      RETURN
30    RL1=MAG*B/8*(8-6*B/L+(B/L)**3)
      RL2=MAG*A/8*(8-6*A/L+(A/L)**3)
      EJL(2)=RL1-RL2
      EJL(5)= MAG*C-EJL(2)
      EJL(3)=0
      MR1=-MAG*B**2/8*(2-(B/L)**2)
      MR2=-MAG*A**2/8*(2-(A/L)**2)
      EJL(6)=(MR1-MR2)*12
      RETURN
40    RR1=MAG*D/8*(8-6*D/L+(D/L)**3)
      RR2=MAG*E/8*(8-6*E/L+(E/L)**3)
      EJL(5)=RR1-RR2
      EJL(2)=MAG*C-EJL(5)
      EJL(6)=0
      ML1=MAG*B**2/2*((1-B/(L*2))**2)
      ML2=MAG*A**2/2*((1-A/(L*2))**2)
      EJL(3)=(ML1-ML2)*12
      RETURN
      END
```

```
C      THIS IS A SUBROUTINE THAT WILL CALCULATE THE EJL FOR A MEMBER
C      WITH A UNIFORM LOAD ALONG THE X AXIS
        SUBROUTINE MX(MAG,L,EJL)
        REAL MAG,LA,LB,EJL(6),L
        INTEGER NI,NJ,CASE
        EJL(1)=MAG*L/2
        EJL(2)=0
        EJL(3)=0
        EJL(4)=MAG*L/2
        EJL(5)=0
        EJL(6)=0
        RETURN
        END


C    THIS IS A SUBROUTINE THAT WILL CALCULATE THE EJL FOR A MEMBER
C    WITH A UNIFORM LOAD ALONG THE Y AXIS--OVER ALL OF THE MEMBER
        SUBROUTINE MY(MAG,L,EJL,CASE)
        REAL MAG,LA,LB,L,EJL(6)
        INTEGER CASE
        EJL(1)=0
        EJL(4)=0
         GOTO (10,20,30,40),CASE
10      EJL(2)=MAG*L/2
        EJL(3)=MAG*L**2/12*12
        EJL(5)=EJL(2)
        EJL(6)=-EJL(3)
        RETURN
20      EJL(2)=MAG*L/2
        EJL(3)=0
        EJL(5)=EJL(2)
        EJL(6)=0
        RETURN
30      EJL(2)=3*MAG*L/8
        EJL(3)=0
        EJL(5)=5*MAG*L/8
        EJL(6)=-MAG*L**2/8*12
        RETURN
40      EJL(2)=5*MAG*L/8
        EJL(3)=MAG*L**2/8*12
        EJL(5)=3*MAG*L/8
        EJL(6)=0
        RETURN
        END


C    THIS IS A SUBROUTINE THAT WILL CALCULATE THE EJL OF A MEMBER
C    WITH A CONCENTRATED LOAD ALONG THE X AXIS
        SUBROUTINE PX(MAG,EJL)
        REAL MAG,EJL(6),L
        EJL(1)=MAG/2
        EJL(2)=0
        EJL(3)=0
        EJL(4)=MAG/2
        EJL(5)=0
        EJL(6)=0
        RETURN
        END
```

```
C    THIS SUBROUTINE WILL CALCULATE THE EJL OF A MEMBER WITH
C    A CONCENTRATED LOAD ALONG THE Y AXIS
        SUBROUTINE PY(MAG,L,LA,LB,EJL,CASE)
        REAL MAG,L,EJL(6),LA,LB,A,B
        INTEGER CASE
        EJL(1)=0
        EJL(4)=0
        A=L*LA
        B=L-A
        GOTO(10,20,30,40),CASE
10      EJL(2)=(MAG*B**2/L**3)*(3*A+B)
        EJI(3)=MAG*A*B**2/L**2*12
        EJL(5)=MAG-EJL(2)
        EJL(6)=-MAG*B*A**2/L**2*12
        RETURN
20      EJL(2)=MAG*B/L
        EJL(3)=0
        EJL(5)=MAG-EJL(2)
        EJL(6)=0
        RETURN
30      EJL(2)=MAG*B**2*(A+2*L)/(2*L**3)
        EJL(3)=0
        EJL(5)=MAG-EJL(2)
        EJL(6)=-MAG*A*B*(A+B/2)/L**2*12
        RETURN
40      EJL(6)=0
        EJL(5)=MAG*B**2*(A+2*L)/(2*L**3)
        EJL(3)=MAG*A*B*(A+B/2)/L**2*12
        EJL(2)=MAG-EJL(5)
        RETURN
        END
```

```
*******************************************************************************************

C       THIS SUBROUTINE IS TO OUTPUT THE RESULTS OF THE PROBLEM
        SUBROUTINE ANSWERS(CASES,NCOMB)
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
        COMMON /RELEASE/ MREL,SREL,STALLY
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT,MREL(40),SREL(40),STALLY
        REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
        REAL  FEMDIS(5,40,6)
        REAL SK
        CHARACTER*4 PAR
        INTEGER I,J,K,L,CASES,JJ,JJJ
        PRINT*,'ANSWERS OUTPUT SECTION'
        PRINT 104,
104     FORMAT(' ',' ')
1       PRINT*,'>>NEXT ANSWER OR EXIT'
        READ 100,PAR
100     FORMAT(A4)
        K=INDEX('FORC SUPP DISP SECF SECD HELP EXIT',PAR)
        IF(K.EQ.0) THEN
         PRINT*,'*** OUTPUT WHAT ?? ***'
         GOTO 1
        END IF
        K=(K+4)/5
        GOTO (10,20,30,40,50,60,70),K
C    MEMBER END FORCES
10      PRINT*,'OUTPUTING MEMBER END FORCES'
        PRINT 106,
106     FORMAT(' ',20X,21('-'))
        PRINT 110,'* MEMBER END FORCES *'
        PRINT 106,
110     FORMAT(' ',20X,A21)
        PRINT 105,
105     FORMAT(' ',60('-'))
        PRINT 111,'MEMBER #','LOAD','JOINT #','AXIAL','SHEAR','MOMENT'
111     FORMAT(' ',A8,3X,A4,3X,A7,2X,A5,7X,A5,7X,A6)
        PRINT 112,
112     FORMAT(' ',11X,'CASE')
        PRINT 105,
        PRINT 104,
        DO 11 I=1,TALLY
         PRINT 113,I
113      FORMAT(' ',3X,I3)
         JJJ=0
          DO 12 JJ=1,CASES+NCOMB
           J=JJ
           IF(JJ.GT.CASES) THEN
                JJJ=JJJ+1
                J=JJJ+5
           END IF
           PRINT 114,J,MT(I,1),EMCASE(J,I,1),EMCASE(J,I,2),EMCASE(J,I,3)
114        FORMAT(' ',12X,I2,6X,F2.0,3X,3(F10.2,2X))
           PRINT 115,MT(I,2),EMCASE(J,I,4),EMCASE(J,I,5),EMCASE(J,I,6)
115        FORMAT(' ',20X,F2.0,3X,3(F10.2,2X))
12        CONTINUE
11       CONTINUE
        GOTO 1
C      SUPPORT OUTPUT
20      PRINT*,'OUTPUTING SUPPORT REACTIONS'
```

```
      PRINT 104,
      PRINT 106,
      PRINT 110,'* SUPPORT REACTIONS *'
      PRINT 106,
      PRINT 104,
      PRINT 120,' JOINT #','LOAD','FORCE X','FORCE Y','MOMENT Z'
120   FORMAT(' ',A8,3X,A4,3X,A7,5X,A7,4X,A8)
      PRINT 112,
      PRINT 105,
      DO 21 I=1,NT
       IF(SREL(I).EQ.0) GOTO 21
       PRINT 121,I
121    FORMAT(' ',3X,I2)
       JJJ=0
       DO 22 JJ=1,CASES+NCOMB
        J=JJ
        IF(JJ.GT.CASES) THEN
              JJJ=JJJ+1
              J=JJJ+5
        END IF
        PRINT 122,J,SUPCASE(J,I,1),SUPCASE(J,I,2),SUPCASE(J,I,3)
122     FORMAT(' ',7X,I2,2X,3(F10.4,2X))
22     CONTINUE
21    CONTINUE
      GOTO 1
C    JOINT DISPLACEMENT
30    PRINT*,'OUTPUTING JOINT DISPLACEMENTS'
      PRINT 104,
      PRINT 106,
      PRINT 110,' JOINT DISPLACEMENTS '
      PRINT 106,
      PRINT 120,' JOINT #','LOAD','TRANS X','TRANS Y','ROTATE Z'
      PRINT 112,
      PRINT 105,
      DO 31 I=1,NT
       PRINT 121,I
       JJJ=0
       DO 32 JJ=1,CASES+NCOMB
        J=JJ
        IF(JJ.GT.CASES) THEN
              JJJ=JJJ+1
              J=JJJ+5
        END IF
        K=(I-1)*3+1
        PRINT 122,J,ACT(J,K),ACT(J,K+1),ACT(J,K+2)
32     CONTINUE
31    CONTINUE
      GOTO 1
C   SECTIONAL MEMBER FORCES
40    PRINT*,'OUTPUTING MEMBER SECTION FORCES'
      PRINT 104,
      PRINT 106,
      PRINT 110,'MEMBER SECTION FORCES'
      PRINT 106,
      PRINT 111,'MEMBER #','LOAD','SECTION','AXIAL','SHEAR','MOMENT'
      PRINT 112,
      PRINT 105,
      DO 41 I=1,TALLY
       JJJ=0
       DO 42 JJ=1,CASES+NCOMB
```

```
          J=JJ
          IF(JJ.GT.CASES) THEN
               JJJ=JJJ+1
               J=JJJ+5
          END IF
         PRINT 141,I,J
141      FORMAT(' ',3X,I2,7X,I2)
        DO 43 K=1,21,2
         SK=(K-1)*.05
         PRINT 143,SK,SECTFORC(J,I,2,K),SECTFORC(J,I,3,K)
143      FORMAT(' ',19X,F4.2,12X,2(F10.2,2X))
C       PRINT 142,SK,SECTFORC(J,I,1,K),SECTFORC(J,I,2,K),SECTFORC(J,I,3,K)
C142    FORMAT(' ',19X,F4.2,3(F10.2,2X))
43      CONTINUE
42      CONTINUE
41     CONTINUE
        GOTO 1
50     GOTO 1
60     PRINT*,'ANSWER OUTPUT HELP SECTION -- COMMANDS AVAILABLE'
       PRINT*,' FORCE  SUPPORT  DISPLACEMENTS  '
       PRINT*,'  SECF  SECD  HELP  EXIT'
        GOTO 1
70     RETURN
       END
```

```
C         SUBROUTINE TO CHANGE THE DATA BASE
        SUBROUTINE CHAN
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        COMMON /RELEASE/MBREL,SREL,STALLY
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        INTEGER MBREL(40),SREL(40),STALLY
        CHARACTER*4 PAR2
        CHARACTER*1 PAR3
        CHARACTER*2 PAR4
        CHARACTER*10 WRI
        CHARACTER*8 FIXED(7)
        INTEGER FIXITY(7)
        CHARACTER*6 STR
        REAL X1,X2,Y1,Y2,MIDX,MIDY
        INTEGER N,NODE,K,J,IFIX
        FIXED(1)='TX TY MZ'
        FIXED(2)='TX TY   '
        FIXED(3)='TX    MZ'
        FIXED(4)='TX      '
        FIXED(5)='   TY MZ'
        FIXED(6)='   TY   '
        FIXED(7)='      MZ'
        FIXITY(1)=111
        FIXITY(2)=110
        FIXITY(3)=101
        FIXITY(4)=100
        FIXITY(5)= 11
        FIXITY(6)= 10
        FIXITY(7)=  1
1       PRINT*,'>>NEXT CHANGE OR EXIT'
        READ 100,PAR2
100     FORMAT(A4)
99      FORMAT(A1)
        K=INDEX('NODE MEMB SUPP MREL CONS PROP EXIT HELP',PAR2)
        IF(K.EQ.0) THEN
         PRINT*,'*** CHANGE WHAT?? ***'
         GOTO 1
        END IF
        K=(K+4)/5
        GOTO (10,20,30,40,50,50,60,70),K
10      PRINT*,'CHANGE LOCATION OF NODE #'
        READ*,N
        IF(N.EQ.0)GOTO 1
        IF(N.LT.0.OR.N.GT.NT) THEN
         PRINT*,'*** INVALID NODE ***'
         CALL BELL
         GOTO 10
        END IF
11      PRINT*,'COORDINATE  X  Y '
         READ*,X,Y
        NLOC(N,1)=X
        NLOC(N,2)=Y
C    SEARCH FOR ALL MEMBERS ATTACHED TO THIS NODE
C       AND CHECK FOR END 1 AND 2
        DO 13 I=1,TALLY
         IF(MT(I,1).EQ.N.OR.MT(I,2).EQ.N) THEN
          X1=NLOC(MT(I,1),1)
          Y1=NLOC(MT(I,1),2)
          X2=NLOC(MT(I,2),1)
```

```
          Y2=NLOC(MT(I,2),2)
          IF(X1.LT.X2)GOTO 12
          IF(X1.EQ.X2) THEN
           IF(Y1.LT.Y2) GOTO 12
          END IF
          TNODE=MT(I,2)
          MT(I,2)=MT(I,1)
          MT(I,1)=TNODE
12       MT(I,5)=(SQRT((X2-X1)**2+(Y2-Y1)**2))*12
         END IF
13       CONTINUE
         PRINT 101,'NODE ',N,' MOVED TO X=',X,' Y=',Y
101      FORMAT(' ',A5,I3,A12,F7.3,A3,F7.3)
         GOTO 1
20       PRINT*,'CHANGE CONNECTIVITY OF MEMBER #'
         READ*,N
         IF(N.EQ.0)GOTO 1
         IF(N.LT.0.OR.N.GT.TALLY) THEN
          PRINT*,'*** NOT A MEMBER ***'
          CALL BELL
          GOTO 20
         END IF
         CALL MOVE(NLOC(MT(N,1),1),NLOC(MT(N,1),2))
         CALL DRAW(NLOC(MT(N,2),1),NLOC(MT(N,2),2))
         CALL CMCLOS
         CALL CMOPEN
         PRINT*,'GIVE NODE NUMBERS (2)'
23        PRINT*,'NODE1  NODE2'
          READ*,N1,N2
           IF(N1.EQ.0.OR.N2.EQ.0) GOTO 1
          IF(N1.LT.0.OR.N1.GT.NT.OR.N2.LT.0.OR.N2.GT.NT) THEN
           PRINT*,'*** INVALID NODE ***'
           GOTO 23
         END IF
         MT(N,1)=N1
         MT(N,2)=N2
         X1=NLOC(N1,1)
         Y1=NLOC(N1,2)
         X2=NLOC(N2,1)
         Y2=NLOC(N2,2)
C    FIND END 1 AND 2
         IF(X1.LT.X2)GOTO 24
         IF(X1.EQ.X2) THEN
          IF(Y1.LT.Y2)GOTO 24
         END IF
         TNODE=MT(N,2)
         MT(N,2)=MT(N,1)
         MT(N,1)=TNODE
24       MT(N,5)=(SQRT((X2-X1)**2+(Y2-Y1)**2))*12

         CALL MOVE(X1,Y1)
         CALL DRAW(X2,Y2)
C    FIND MIDPOINT
         MIDX=X1+(X2-X1)/2
         MIDY=Y1+(Y2-Y1)/2
         CALL MOVE(MIDX,MIDY)
         CALL INUMBR(N,3)
         CALL CMCLOS
         CALL CMOPEN
         GOTO 1
```

```
30      PRINT*,'CHANGE CONDITION OF SUPPORT #'
        READ*,N
        IF(N.EQ.0)GOTO 1
        IF(N.LT.0.OR.N.GT.NT)THEN
         PRINT*,'*** INVALID NODE ***'
         CALL BELL
         GOTO 30
        END IF
        IF(SREL(N).EQ.0) THEN
         PRINT 102, 'SORRY JOINT ',N,' IS NOT A SUPPORT'
102     FORMAT(' ',A12,I3,A17)
         CALL MOVE(NLOC(N,1),NLOC(N,2))
         CALL INUMBR(N,3)
         CALL CMCLOS
         CALL CMOPEN
         GOTO 30
        END IF
C   OBTAIN DEGREES OF FIXITY
        IFIX=SREL(N)
        DO 33 I=1,7
33       IF(IFIX.EQ.FIXITY(I)) GOTO 34
34        PRINT 132,N,FIXED(I)
132     FORMAT(' ',I3,5X,A8)
        SREL(N)=111
        PRINT*,'SUPPORT IS NOW COMPLETELY FIXED'
31      PRINT*,'ENTER RELEASE DIRECTIONS :'
        PRINT*,'TX TY RZ TT XR YR NONE'
        READ 98,PAR4
98      FORMAT(A2)
        K=INDEX('NO RZ TY YR TX XR TT NO',PAR4)
        IF(K.EQ.0) GOTO 31
        K=(K+2)/3
        SREL(N)=FIXITY(K)
        PRINT 105,'SUPPORT ',N,' NOW FIXED IN ',FIXED(K),' DIRECTIONS '
105     FORMAT(' ',A8,I3,A13,A8,A12)
        GOTO 1

40      PRINT*,'CHANGE END RELEASE OF MEMBER #'
        READ*,N
        IF(N.EQ.0)GOTO 1
        IF(N.LT.0.OR.N.GT.TALLY) THEN
         PRINT*,'*** NOT A VALID MEMBER ***'
         CALL BELL
         GOTO 40
        END IF
        CALL MOVE(NLOC(MT(N,1),1),NLOC(MT(N,1),2))
        CALL DRAW(NLOC(MT(N,2),1),NLOC(MT(N,2),2))
        CALL CMCLOS
        CALL CMOPEN
        IF(MBREL(N).EQ.0) THEN
         PRINT*,'** PERHAPS THE WRONG MEMBER--BOTH ENDS FIXED **'
         GOTO 40
        END IF
C   DISPLAY RELEASES
        IF(MBREL(N).EQ.1) THEN
         STR='  BOTH '
         ELSE IF(MBREL(N).EQ.2) THEN
          STR=' START '
          ELSE
           STR='  END   '
```

```
      END IF
      PRINT103,'MEMBER ',N,STR,' END RELEASED'
103   FORMAT(' ',A7,I3,A7,A13)
      PRINT*,'NOW BOTH ENDS FIXED'
41    PRINT*,'RELEASE WHICH ENDS: START END BOTH NEITHER'
      READ*,PAR3
      K=INDEX('BSEN',PAR3)
      IF(K.EQ.0) GOTO 41
      MBREL(N)=K+1
      IF(K.EQ.4) MBREL(N)=0
      PRINT*,'MEMBER END RELEASES ADJUSTED'
      GOTO 1
50    PRINT*,'CHANGE CONSTANTS OR PROPERTIES MEMBER #'
      READ*,N
      IF(N.EQ.0) GOTO 1
      IF(N.LT.0.OR.N.GT.TALLY) THEN
       PRINT*,'*** NOT A MEMBER ***'
       GOTO 50
       END IF
      PRINT*,'  L      E       A      I'
      PRINT*,'  in.  ksi    in¬4  in¬2'
      PRINT 104,MT(N,5),MT(N,6),MT(N,7),MT(N,8)
104   FORMAT(' ',F8.2,2X,F8.0,2X,F6.1,2X,F4.1)
51    PRINT*,'CHANGE WHICH:  E  I  A'
      READ*,PAR3
      K=INDEX('EIA',PAR2)
      IF(K.EQ.0) GOTO 51
52    PRINT*,'INPUT NEW VALUE'
      READ*,X
      IF(X.LE.0) GOTO 52
      MT(N,K+5)=X
      PRINT 104,MT(N,5),MT(N,6),MT(N,7),MT(N,8)
      GOTO 1
60    PRINT*,'EXIT CHANGE SECTION'
      RETURN
70    PRINT*,'CHANGE HELP SECTION--COMMANDS AVAILABLE:'
      PRINT*,'NODE MEMB SUPP MREL CONS PROP EXIT HELP'
      GOTO 1
      END
```

```
C    THIS IS A SUB TO DELETE DATA FROM THE DATABASE
        SUBROUTINE ERASE(NERASE,MERASE)
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
        COMMON /LOADONE/MLTALLY,JLTALLY,MLOAD,JLOAD
        COMMON /RELEASE/MBREL,SREL,STALLY
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        REAL MCASE(5,40,6),JCASE(5,40,3)
        INTEGER CASES,NMCASE(5),NJCASE(5)
        REAL MLOAD(40,6),JLOAD(40,3)
        INTEGER MLTALLY,JLTALLY
        INTEGER MBREL(40),SREL(40),STALLY
        CHARACTER*7 TYPE
        CHARACTER*4 PAR2
        CHARACTER*8 DIR
        INTEGER TJL,TML,LNUM,NERASE,MERASE
        INTEGER N,I,J,K,TLT,TLD(10)
        LOGICAL LTEST
1       PRINT*,'>>NEXT DELETE OR EXIT'
        READ 100,PAR2
100     FORMAT(A4)
        K=INDEX('NODE MEMB SUPP MREL JLOA MLOA EXIT HELP',PAR2)
        IF(K.EQ.0) THEN
         PRINT*,'*** DELETE WHAT?? ***'
         CALL BELL
         GOTO 1
        END IF
        K=(K+4)/5
        GOTO (10,20,30,40,80,90,60,70),K
10      PRINT*,'DELETE NODE #'
        READ*,N
        IF(N.EQ.0)GOTO 1
        IF(N.LT.0.OR.N.GT.NT) THEN
         PRINT*,'*** NOT A NODE ***'
         CALL BELL
         GOTO 10
        END IF
C    CHECK IF NODE IS ATTACHED TO A MEMBER
        DO 11 I=1,TALLY
         IF(MT(I,1).EQ.N.OR.MT(I,2).EQ.N) THEN
          PRINT*,'*** CANNOT DELETE NODE ***'
           PRINT 101,'MEMBER ',I,' ATTACHED '
101        FORMAT(' ',A6,I3,A10)
           GOTO 1
         END IF
11      CONTINUE
C    SEE IF ANY JOINT LOADS ARE ON THIS NODE
        TJL=JLTALLY
        DO 12 I=1,TJL
         IF(JLOAD(I,1).EQ.N) THEN
13        IF(JLOAD(JLTALLY,1).EQ.N) THEN
           JLOAD(JLTALLY,1)=0
           JLOAD(JLTALLY,2)=0
           JLOAD(JLTALLY,3)=0
           JLTALLY=JLTALLY-1
           GOTO 13
          END IF
           JLOAD(I,1)=JLOAD(JLTALLY,1)
           JLOAD(JLTALLY,1)=0
```

```
          JLOAD(I,2)=JLOAD(JLTALLY,2)
          JLOAD(JLTALLY,2)=0
          JLOAD(I,3)=JLOAD(JLTALLY,3)
          JLOAD(JLTALLY,3)=0
          JLTALLY=JLTALLY-1
         END IF
12      CONTINUE
C     SEE IF NODE IS A SUPPORT
        IF(SREL(N).NE.0) THEN
          PRINT 103 ,'NODE ',N,' WAS A SUPPORT; BUT IS NOW DELETED'
          STALLY=STALLY-1
103     FORMAT(' ',A5,I3,A32)
        END IF
        SREL(N)=0
        NLOC(N,1)=-1000
        NLOC(N,2)=-1000
        PRINT 104,'NODE ',N,' IS NOW GONE!!'
104     FORMAT(' ',A5,I3,A14)
        GOTO 1
20      PRINT*,'DELETE MEMBER #'
        READ*,N
        IF(N.EQ.0) GOTO 1
        IF(N.LT.0.OR.N.GT.TALLY) THEN
         PRINT*,'*** INVALID MEMBER ***'
         CALL BELL
         GOTO 20
         END IF
        DO 21 I=1,12
21       MT(N,I)=0
C     CHECK AND DELETE MEMBER LOADS
        TML=MLTALLY
        DO 22 I=1,TML
         IF(MLOAD(I,1).EQ.N) THEN
          IF(MLOAD(MLTALLY,1).EQ.N) THEN
           DO 23 J=1,6
            MLOAD(MLTALLY,J)=0
23         CONTINUE
            MLTALLY=MLTALLY-1
           END IF
          DO 24 J=1,6
         MLOAD(I,J)=MLOAD(MLTALLY,J)
         MLOAD(MLTALLY,J)=0
24      CONTINUE
         MLTALLY=MLTALLY-1
        END IF
22      CONTINUE
C     CHANGE ANY MEMBER END RELEASES TOO
        MBREL(N)=0
        PRINT 109,'MEMBER ',N,' NOW GONE...FOREVER'
109     FORMAT(' ',A7,I3,A18)
         GOTO 1
30      PRINT*,'ERASE SUPPORT #'
        READ*,N
        IF(N.EQ.0)GOTO 1
        IF(N.LT.0.OR.N.GT.NT)THEN
         PRINT*,'*** NOT A VALID NODE ***'
         CALL BELL
         GOTO 30
        END IF
```

```
      IF(SREL(N).EQ.0) THEN
       PRINT105 ,'YOU GOOFED---',N,' IS NOT A SUPPORT'
105    FORMAT(' ',A13,I3,A17)
       GOTO 30
      END IF
      SREL(N)=0
      PRINT*,'NODE IS NO LONGER A SUPPORT'
      STALLY=STALLY-1
      GOTO 1
40    PRINT*,'DELETE MEMBER END RELEASE #'
      READ*,N
      IF(N.EQ.0)GOTO 1
      IF(N.LT.0.OR.N.GT.TALLY) THEN
       PRINT*,'*** NOT A MEMBER ***'
       CALL BELL
       GOTO 40
      END IF
      IF(MBREL(N).EQ.0)THEN
       PRINT 108,'MEMBER ',N,' HAS NO ENDS RELEASED'
108    FORMAT(' ',A7,I3,A21)
       GOTO 40
      END IF
       MBREL(N)=0
       PRINT*,'BOTH MEMBER ENDS ARE NOW FIXED'
       GOTO 1
C   ERASE THE JOINT LOAD
80    PRINT*,'DELETE LOAD ON JOINT #'
      READ*,N
      IF(N.EQ.0) GOTO 1
      IF(N.LT.0.OR.N.GT.NT) THEN
       PRINT*,'*** INVALID JOINT NUMBER ***'
       GOTO 80
      END IF
      LTEST=.TRUE.
      TLT=0
      PRINT*,'LIST # -- JOINT # -- DIRECTION -- MAGNITUDE'
      DO 81 I=1,JLTALLY
       IF(JLOAD(I,1).EQ.N) THEN
        LTEST = .FALSE.
        IF(JLOAD(I,2).EQ.1) THEN
         DIR='FORCE  X'
        ELSE IF(JLOAD(I,2).EQ.2) THEN
         DIR='FORCE  Y'
        ELSE
         DIR='MOMENT Z'
        END IF
        TLT=TLT+1
        TLD(TLT)=I
        PRINT 110,TLT,JLOAD(I,1),DIR,JLOAD(I,3)
       END IF
81    CONTINUE
110   FORMAT(' ',I3,8X,F4.0,8X,A8,4X,F9.2)
      IF(LTEST) THEN
        PRINT 111,'*** SORRY JOINT ',N,' HAS NO LOADS ***'
        GOTO 1
      END IF
111   FORMAT(' ',A16,I3,A17)
82    PRINT*,'ENTER LOAD LIST #'
      READ*,M
      IF(M.EQ.0) GOTO 1
```

```
      IF(M.LT.0.OR.M.GT.TLT) THEN
       PRINT*,'*** BAD LIST NUMBER ***'
       GOTO 82
      END IF
      LNUM=TLD(M)
C   FIND THE DIRECTIN OF THIS LOAD - AGAIN
        IF(JLOAD(LNUM,2).EQ.1) THEN
         DIR='FORCE  X'
        ELSE IF(JLOAD(LNUM,2).EQ.2) THEN
         DIR='FORCE  Y'
        ELSE
         DIR='MOMENT Z'
        END IF
      PRINT*,'THE FOLLOWING LOAD IS NOW GONE'
        PRINT 110,M,JLOAD(LNUM,1),DIR,JLOAD(LNUM,3)
      IF(LNUM.EQ.JLTALLY) THEN
       JLOAD(JLTALLY,1)=0
       JLOAD(JLTALLY,2)=0
       JLOAD(JLTALLY,3)=0
      ELSE
       DO 83 I=1,3
        JLOAD(LNUM,I)=JLOAD(JLTALLY,I)
        JLOAD(JLTALLY,I)=0
83     CONTINUE
      END IF
      JLTALLY=JLTALLY-1
      GOTO 1
C  MEMBER LOAD DELETE SECTION
90    PRINT*,'DELETE LOAD ON MEMBER #'
      READ*,N
      IF(N.EQ.0) GOTO 1
      IF(N.LT.0.OR.N.GT.TALLY) THEN
       PRINT*,'*** INVALID MEMBER NUMBER ***'
       GOTO 90
      END IF
      TLT=0
      LTEST=.TRUE.
      PRINT*,'LIST # -- MEMBER # -- TYPE -- DIRECTION -- MAGNITUDE --
     * START -- END'
      DO 91 I=1,MLTALLY
       IF(MLOAD(I,1).EQ.N) THEN
        LTEST=.FALSE.
        IF(MLOAD(I,2).EQ.1) THEN
         TYPE='CONCEN '
        ELSE IF(MLOAD(I,2).EQ.2) THEN
         TYPE='UNIFORM'
        ELSE
         TYPE='MOMENT '
        END IF
        IF(MLOAD(I,3).EQ.1) THEN
         DIR='FORCE  X'
        ELSE IF(MLOAD(I,3).EQ.2) THEN
         DIR='FORCE  Y'
        ELSE
         DIR='MOMENT Z'
        END IF
        TLT=TLT+1
        TLD(TLT)=I
        PRINT 112,I,MLOAD(I,1),TYPE,DIR,MLOAD(I,4),MLOAD(I,5),MLOAD(I,6)
       END IF
```

```
91      CONTINUE
112     FORMAT(' ',2X,I3,8X,F3.0,6X,A7,2X,A8,5X,F9.2,4X,F4.3,5X,F4.3)
        IF(LTEST) THEN
         PRINT 113,'*** SORRY MEMBER ',N,' HAS NO LOADS ***'
         GOTO 1
        END IF
113     FORMAT(' ',A17,I3,A17)
92      PRINT*,'ENTER LIST NUMBER LOAD'
        READ*,M
        IF(M.EQ.0) GOTO 1
        IF(M.LT.0.OR.M.GT.TLT) THEN
         PRINT*,'*** BAD LIST NUMBER ***'
         GOTO 92
        END IF
        LNUM=TLD(M)
C    FIND LOAD TYPE AND DIRECTION AGAIN
          IF(MLOAD(LNUM,2).EQ.1) THEN
           TYPE='CONCEN '
          ELSE IF(MLOAD(LNUM,2).EQ.2) THEN
           TYPE='UNIFORM'
          ELSE
           TYPE='MOMENT '
          END IF
          IF(MLOAD(LNUM,3).EQ.1) THEN
           DIR='FORCE  X'
          ELSE IF(MLOAD(LNUM,3).EQ.2) THEN
           DIR='FORCE  Y'
          ELSE
           DIR='MOMENT Z'
          END IF
          PRINT 112,LNUM,MLOAD(LNUM,1),TYPE,DIR,MLOAD(LNUM,4)
     *          ,MLOAD(LNUM,5),MLOAD(LNUM,6)
        IF(LNUM.EQ.MLTALLY)THEN
        DO 93 I=1,6
         MLOAD(MLTALLY,I)=0
93      CONTINUE
         ELSE
          DO 94 I=1,6
           MLOAD(LNUM,I)=MLOAD(MLTALLY,I)
           MLOAD(MLTALLY,I)=0
94        CONTINUE
        END IF
        MLTALLY=MLTALLY-1
        GOTO 1
70      PRINT*,'DELETE HELP SECTION--COMMANDS AVAILABLE:'
        PRINT*,'NODE MEMBER SUPPORT MREL EXIT HELP'
        GOTO 1
60      PRINT*,'EXIT DELETE SECTION'
        RETURN
        END
```

```
      SUBROUTINE LIS
      COMMON /GEOM/   MT,TALLY,NLOC,NT
      COMMON /LOADONE/MLTALLY,JLTALLY,MLOAD,JLOAD
      COMMON /RELEASE/MBREL,SREL,STALLY
      REAL MT(40,12),NLOC(40,2)
      INTEGER TALLY,NT
      REAL MLOAD(40,6),JLOAD(40,3)
      INTEGER MLTALLY,JLTALLY
      INTEGER MBREL(40),SREL(40),STALLY
      CHARACTER*4 PAR2
      CHARACTER*6 TYP
      CHARACTER*10 WRI
      INTEGER BEG,EN,INC
1     PRINT*,'>>NEXT LIST OR EXIT'
      READ 100,PAR2
100   FORMAT(A4)
      K=INDEX('NODE MEMB SUPP MREL JLOA MLOA CONS PROP EXIT
     * HELP',PAR2)
      IF(K.EQ.0) THEN
       PRINT*,'www LIST WHAT ?? www>>'
        GOTO 1
      END IF
      K=(K+4)/5
      GOTO(110,120,130,150,160,170,180,190,200,210),K

110   PRINT*,'LIST NODES:  BEGIN,END>>>'
      READ*,BEG,EN
      IF(BEG.LT.1.OR.BEG.GT.EN.OR.BEG.GT.NT)THEN
        PRINT*,'www INVALID NODE # www'
        GOTO 110
      END IF
      IF(EN.GT.NT)EN=NT
      PRINT*,'NODE #______X______Y  '
      PRINT*,' '
      DO 112 I=BEG,EN
112   PRINT 111,I,NLOC(I,1),NLOC(I,2)
      GOTO 1
111   FORMAT(' ',2X,I3,5X,F9.3,3X,F9.3)
120   PRINT*,'LIST MEMBER CONNECTIVITY:  BEGIN,END>>>'
      READ*,BEG,EN
      IF(BEG.LT.1.OR.BEG.GT.EN.OR.BEG.GT.TALLY)THEN
        PRINT*,'www INVALID MEMBER # www'
        GOTO 120
       END IF
      IF(EN.GT.TALLY)EN=TALLY
      PRINT *,'MEMBER #___BEGIN____END____LENGTH'
      PRINT*,' '
      DO 122 I=BEG,EN
122    PRINT 121,I,MT(I,1),MT(I,2),MT(I,5)
      GOTO 1
121   FORMAT(' ',2X,I3,8X,F3.0,4X,F3.0,5X,F6.2)
130   PRINT*,'LIST SUPPORTS JOINTS:  BEGIN,END'
      READ*,BEG,EN
      IF(BEG.LT.1.OR.BEG.GT.EN.OR.BEG.GT.NT)THEN
        PRINT*,'www INVALID JOINT # www'
        GOTO 130
       END IF
      IF(EN.GT.NT) EN=NT
      PRINT*,'JOINT #_____FIXED '
      PRINT*,' '
```

```
      DO 131 I=BEG,EN
      IF(SREL(I).EQ.0) THEN
       GOTO 131
      ELSE IF(SREL(I).EQ.111) THEN
      WRI='TX TY MX'
       ELSE IF(SREL(I).EQ.110) THEN
       WRI='TX TY   '
        ELSE IF(SREL(I).EQ.101) THEN
         WRI='TX    MZ'
        ELSE IF(SREL(I).EQ.11) THEN
         WRI='  TY MZ'
             ELSE IF(SREL(I).EQ.10) THEN
              WRI='   TY   '
       ELSE
              WRI='       MZ'
      END IF
       PRINT 132,I,WRI
131   CONTINUE
132   FORMAT(' ',I3,5X,A10)
      GOTO 1
150   PRINT*,'MEMBER END RELEASES:  BEGIN,END'
      READ*,BEG,EN
      IF(EN.GT.TALLY)  EN=TALLY
      IF(BEG.LT.1.OR.BEG.GT.EN.OR.BEG.GT.TALLY)THEN
       PRINT*,'www INVALID MEMBER # www'
        GOTO 150
       END IF
       PRINT*,'MEMBER #____RELEASES'
       PRINT*,' '
      DO 151 I=BEG,EN
       IF(MBREL(I).EQ.0) GOTO 151
       IF(MBREL(I).EQ.2) THEN
        WRI='BEG       '
        ELSE IF(MBREL(I).EQ.3) THEN
         WRI='       END'
        ELSE
         WRI='BEG    END'
        END IF
       PRINT 152,I,WRI
151   CONTINUE
152   FORMAT(' ',I3,2X,A10)
      GOTO 1
160            PRINT*,'JOINT LOADS:  BEGIN,END'
      READ*,BEG,EN
      IF(BEG.LT.1.OR.BEG.GT.EN.OR.BEG.GT.NT)THEN
       PRINT*,'www INVALID NODE# www'
       GOTO 160
      END IF
      IF(EN.GT.NT)EN=NT
      PRINT*,'JOINT #___DIRECTION___MAGNITUDE'
      PRINT*,' '
      DO 163 I=BEG,EN
       DO 162 J=1,JLTALLY
        IF(JLOAD(J,1).NE.I)GOTO 162
         IF(JLOAD(J,2).EQ.1) THEN
              WRI='FX'
          ELSE IF(JLOAD(J,2).EQ.2) THEN
               WRI='FY'
           ELSE
              WRI='MZ'
```

```
   END IF
                        PRINT 169,I,WRI,JLOAD(J,3)
162     CONTINUE
163     CONTINUE
169     FORMAT(' ',I3,10X,A2,5X,F10.3)
        GOTO 1
170     PRINT*,'MEMBER LOADS:  BEGIN,END'
        READ*,BEG,EN
        IF(BEG.LT.1.OR.BEG.GT.EN.OR.BEG.GT.TALLY)THEN
         PRINT*,'www INVALID MEMBER # www'
         GOTO 170
        END IF
        IF(EN.GT.TALLY) EN=TALLY
         PRINT*,'MEMBER #___TYPE___DIR___MAGNITUDE___BEG___END'
        PRINT*,' '
        DO 173 I=BEG,EN
         DO 172 J=1,MLTALLY
          IF(MLOAD(J,1).NE.I) GOTO 172
                IF(MLOAD(J,2).EQ.1) THEN
                 TYP='CONCEN'
                  ELSE IF(MLOAD(J,2).EQ.2)THEN
                        TYP='UNIFOR'
                  ELSE
                 TYP='MOMENT'
         END IF
        IF(MLOAD(J,3).EQ.1)THEN
         WRI='FX'
         ELSE IF(MLOAD(J,3).EQ.2) THEN
          WRI='FY'
         ELSE
          WRI='MZ'
         END IF
        PRINT 171,I,TYP,WRI,MLOAD(J,4),MLOAD(J,5),MLOAD(J,6)
172     CONTINUE
173     CONTINUE
        GOTO 1
171     FORMAT(' ',2X,I3,5X,A6,4X,A3,2X,F9.3,2X,F6.4,2X,F6.4)
180     PRINT*,'LIST CONSTANTS:  BEGIN,END'
        READ*,BEG,EN
        IF(BEG.LT.1.OR.BEG.GT.EN.OR.BEG.GT.TALLY)THEN
          PRINT*,'www  INVALID MEMBER #  www'
          GOTO 180
        END IF
         IF(EN.GT.TALLY) EN=TALLY
         PRINT*,'MEMBER #___ E ksi ___ '
        DO 181 I=BEG,EN
         PRINT 182,I,MT(I,6)
181     CONTINUE
182     FORMAT(' ',2X,I2,4X,F9.0)
        GOTO 1

190     PRINT*,'LIST PROPERTIES:  BEGIN,END'
        READ*,BEG,EN
        IF(BEG.LT.1.OR.BEG.GT.EN.OR.BEG.GT.TALLY)THEN
         PRINT*,'www INVALID MEMBER # www'
         goto 190
         END IF
        IF(EN.GT.TALLY) EN=TALLY
         PRINT*,'MEMBER #___ AREA ___ Iz ___'
        DO 191 I=BEG,EN
```

```
          PRINT 192,I,MT(I,7),MT(I,8)
191       CONTINUE
192       FORMAT(' ',2X,I2,4X,F6.2,2X,F8.2)
          GOTO 1
200       PRINT*,'EXIT THIS SECTION'
          RETURN
210       PRINT*,'LIST HELP SECTION  COMMANDS AVAILABLE ARE:'
          PRINT*,' NODE MEMB SUPP MREL JLOA MLOA CONS PROP EXIT HELP'
          GOTO 1
          END
```

```
      SUBROUTINE OUT(NAME)
      COMMON /GEOM/   MT,TALLY,NLOC,NT
      COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
      COMMON /LOADONE/MLTALLY,JLTALLY,MLOAD,JLOAD
      COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
      COMMON /RELEASE/MBREL,SREL,STALLY
      REAL MT(40,12),NLOC(40,2)
      INTEGER TALLY,NT
      REAL MCASE(5,40,6),JCASE(5,40,3)
      INTEGER CASES,NMCASE(5),NJCASE(5)
      REAL MLOAD(40,6),JLOAD(40,3)
      INTEGER MLTALLY,JLTALLY
      REAL COMB(5,5)
      INTEGER NCOMB,ACTLIST(10),ACASES
      INTEGER MBREL(40),SREL(40),STALLY
      CHARACTER*4 PAR2
      CHARACTER*6 TYP
      CHARACTER*10 WRI,VAXNAME
      CHARACTER*30 TITLE,NAME(10)
      INTEGER BEG,EN,INC
C     THIS OUTPUTS A SUMMARY OF THE STRUCTURAL DATA
      PRINT*,'STRUCTURE OUTPUT SECTION'
      PRINT*,'> ENTER A   VAX  OUTPUT FILE NAME'
      READ 200,VAXNAME
      OPEN(UNIT=8,FILE=VAXNAME,STATUS='NEW')
      PRINT*,'> ENTER A TITLE FOR THIS STRUCTURE-- 30 CHARARACTERS MAX'
      READ 201,TITLE
      WRITE(8,198)
198   FORMAT(' ','JOB TITILE:')
      WRITE (8,199)
199   FORMAT(' ',60('*'))
      WRITE (8,197),TITLE
197   FORMAT(' ','*',14X,A30,14X,'*')
      WRITE (8,199)
      WRITE (8,196)
196   FORMAT(' ',' ')
      WRITE(8,210),'NUMBER OF MEMBERS       => ',TALLY
      WRITE(8,210),'NUMBER OF JOINTS        => ',NT
      WRITE(8,210),'NUMBER OF LOAD CASES    => ',CASES
      WRITE(8,210),'NUMBER OF LOAD COMB     => ',NCOMB
210   FORMAT(' ',A26,I3)
200   FORMAT(A10)
201   FORMAT(A30)
      IF(NT.LE.0) THEN
       PRINT*,'SORRY THERE ARE NO NODES--WHY STORE IT !!'
       RETURN
      END IF
      WRITE(8,196)
      WRITE(8,*),'NODE #______X___________Y__   '
      DO 112 I=1,NT
112   WRITE(8,111),I,NLOC(I,1),NLOC(I,2)
111   FORMAT(' ',2X,I3,2X,F9.3,2X,F9.3)
      WRITE(8,196)
      WRITE(8,*),'MEMBER #___BEGIN____END____LENGTH'
      DO 122 I=1,TALLY
122    WRITE(8,121),I,MT(I,1),MT(I,2),MT(I,5)
121   FORMAT(' ',2X,I3,8X,F3.0,4X,F3.0,5X,F6.2)
      WRITE(8,196)
      WRITE(8,*),'SUPPORT JOINT #_____FIXED '
       DO 131 I=1,NT
```

```
          WRI='              '
         IF(SREL(I).EQ.0) THEN
          GOTO 131
         ELSE IF(SREL(I).EQ.111) THEN
         WRI='TX TY MZ'
          ELSE IF(SREL(I).EQ.110) THEN
          WRI='TX TY    '
           ELSE IF(SREL(I).EQ.101) THEN
            WRI='TX    MZ'
           ELSE IF(SREL(I).EQ.11) THEN
            WRI='  TY MZ'
                ELSE IF(SREL(I).EQ.10) THEN
                 WRI='   TY    '
          ELSE
                  WRI='         MZ'
         END IF
          WRITE(8,132),I,WRI
131     CONTINUE
        WRITE(8,196)
132     FORMAT(' ',8X,I3,8X,A10)
         WRITE(8,*),'MEMBER #____RELEASES'
        DO 151 I=1,TALLY
         IF(MBREL(I).EQ.0) GOTO 151
         IF(MBREL(I).EQ.2) THEN
          WRI='BEG       '
          ELSE IF(MBREL(I).EQ.3) THEN
           WRI='       END'
          ELSE
           WRI='BEG    END'
          END IF
         WRITE(8,152),I,WRI
151     CONTINUE
152     FORMAT(' ',2XI3,2X,A10)
        WRITE(8,196)
         WRITE(8,*),'MEMBER #___ E ksi ___ '
        DO 181 I=1,TALLY
         WRITE(8,182),I,MT(I,6)
181     CONTINUE
182     FORMAT(' ',2X,I3,3X,F9.0)
        WRITE(8,196)
         WRITE(8,*),'MEMBER #___ AREA ___ Iz ___'
        DO 191 I=1,TALLY
         WRITE(8,192),I,MT(I,7),MT(I,8)
191     CONTINUE
192     FORMAT(' ',2X,I3,4X,F6.2,2X,F8.2)
        WRITE(8,196)
        DO 179 LC=1,CASES
         WRITE(8,215),' LOAD CASE # ',LC
         WRITE(8,196)
215      FORMAT(' ',22('*'),A13,I2,' ',21('*'))
         WRITE(8,2162),'  LOAD CASE TITLE ===> ',NAME(LC)
2162     FORMAT(' ',A23,A30)
        WRITE(8,*),'JOINT #___DIRECTION___MAGNITUDE'
        DO 163 I=1,NT
         DO 162 J=1,NJCASE(LC)
          IF(JCASE(LC,J,1).NE.I)GOTO 162
           IF(JCASE(LC,J,2).EQ.1) THEN
                WRI='FX'
            ELSE IF(JCASE(LC,J,2).EQ.2) THEN
                WRI='FY'
```

```
          ELSE
              WRI='MZ'
        END IF
         WRITE(8,169),I,WRI,JCASE(LC,J,3)
162   CONTINUE
163   CONTINUE
169   FORMAT(' ',2X,I3,9X,A2,5X,F10.3)
      WRITE(8,196)
       WRITE(8,*),'MEMBER #___TYPE_____DIR____MAGNITUDE__BEG___END'
      DO 173 I=1,TALLY
       DO 172 J=1,NMCASE(LC)
        IF(MCASE(LC,J,1).NE.I) GOTO 172
              IF(MCASE(LC,J,2).EQ.1) THEN
               TYP='CONCEN'
                ELSE IF(MCASE(LC,J,2).EQ.2)THEN
                      TYP='UNIFOR'
                ELSE
               TYP='MOMENT'
       END IF
      IF(MCASE(LC,J,3).EQ.1)THEN
       WRI='FX'
       ELSE IF(MCASE(LC,J,3).EQ.2) THEN
        WRI='FY'
       ELSE
        WRI='MZ'
       END IF
      WRITE(8,171),I,TYP,WRI,MCASE(LC,J,4),MCASE(LC,J,5),MCASE(LC,J,6)
172   CONTINUE
173   CONTINUE
171   FORMAT(' ',2X,I3,5X,A6,4X,A3,2X,F9.3,2X,F6.4,2X,F6.4)
179   CONTINUE
      WRITE(8,196)
      WRITE(8,*),'LOAD COMBINATION DATA'
      WRITE(8,196)
      IF(NCOMB.GE.1) THEN
       WRITE(8,213)
213   FORMAT(' ',14X,'CASE # =>  1       2       3       4       5')
      WRITE(8,218)
218   FORMAT(' ',2X,'COMB NAME')
      END IF
       DO 211 N=1,NCOMB
        M=N+5
        WRITE(8,212),M,NAME(M)
212     FORMAT(' ',2X,I3,2X,A30)
        WRITE(8,223),(COMB(N,J),J=1,CASES)
223     FORMAT(' ',22X,5(F6.3,1X))
211   CONTINUE
      WRITE(8,196)
      WRITE(8,219)
219   FORMAT(' ',12('*'),' ACTIVE POST-PROCESSING LOAD CASES ',12('*'))
      DO 180 I=1,ACASES
       WRITE(8,220),ACTLIST(I),NAME(ACTLIST(I))
180   CONTINUE
220   FORMAT(' ',2X,I3,5X,A30)
      WRITE(8,221)
221   FORMAT(' ',3X,'11',5X,'MAXIMUM ENVELOPE')
      WRITE(8,222)
222   FORMAT(' ',3X,'12',5X,'MINIMUM ENVELOPE')
      CLOSE(UNIT=8)
      RETURN
```

```
C    THIS SUBROUTINE WILL RESTORE THE STRUCTURAL DATA
C    INTO THE PROGRAM.. THE INFORMATION WAS STORED VIA
C    THE SUBROUTINE "SAVE"
         SUBROUTINE RESTORE(NAME,LCASE)
         COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
         COMMON /GEOM/   MT,TALLY,NLOC,NT
         COMMON /LOADONE/ MLTALLY,JLTALLY,MLOAD,JLOAD
         COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
         COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
         COMMON /RELEASE/MBREL,SREL,STALLY
         REAL ZX,WX,ZY,WY,ROUND
         REAL MT(40,12),NLOC(40,2)
         INTEGER TALLY,NT
         REAL MLOAD(40,6),JLOAD(40,3)
         INTEGER MLTALLY,JLTALLY
         REAL MCASE(5,40,6),JCASE(5,40,3)
         INTEGER CASES,NMCASE(5),NJCASE(5)
         REAL COMB(5,5)
         INTEGER NCOMB,ACTLIST(10),ACASES
         INTEGER MBREL(40),SREL(40),STALLY
         INTEGER I,J,K,L,II
         REAL A,B,C,D,E,F
         CHARACTER*8 VAXNAME
         CHARACTER*30 TITLE,NAME(10)
         PRINT*,'>>RESTORE ROUTINE:'
         IF(TALLY.GE.1) RETURN
         PRINT*,'>>>ENTER THE VAX FILE NAME -- 8 CHAR MAX'
         READ 100,VAXNAME
100      FORMAT(A8)
         OPEN(UNIT=1,FILE=VAXNAME,STATUS='OLD')
         READ(1,201),TITLE
201      FORMAT(1X,A30)
202      FORMAT(1X,I3)
C     READ IN THE STRUCTURAL PARAMETER COUNTERS
         READ(1,202),NT
         READ(1,202),TALLY
         READ(1,202),CASES
         READ(1,202),NCOMB
         READ(1,202),ACASES
C     READ IN THE NODE LOCATIONS
         DO 10 J=1,NT
          READ(1,203)X,Y
          NLOC(J,1)=X
10        NLOC(J,2)=Y
203      FORMAT(1X,F10.4,2X,F10.4)
C     READ IN THE MEMBER PARAMETERS
         DO 11 M=1,TALLY
          READ(1,204),A,B,C,D,E,F
          MT(M,1)=A
          MT(M,2)=B
          MT(M,5)=C
          MT(M,6)=D
          MT(M,7)=E
11        MT(M,8)=F
204      FORMAT(1X,F3.0,2X,F3.0,2X,F8.3,2X,F8.0,2X,F8.3,2X,F8.3)
C     READ IN THE MEMBER RELEASES
         DO 12 M=1,TALLY
          READ(1,205),I
12        MBREL(M)=I
205      FORMAT(1X,I3)
```

```
C     READ IN SUPPORTS
         DO 13 J=1,NT
          READ(1,205),I
13        SREL(J)=I
C     READ IN THE LOAD CASE INFORMATION
         DO 20 J=1,CASES
          READ(1,209),TITLE
          NAME(J)=TITLE
          READ(1,210),I,II
          NJCASE(J)=I
          NMCASE(J)=II
209      FORMAT(1X,A30)
210      FORMAT(1X,I3,2X,I3)
          DO 21 I=1,NJCASE(J)
           READ(1,211),A,B,C
           JCASE(J,I,1)=A
           JCASE(J,I,2)=B
21         JCASE(J,I,3)=C
211      FORMAT(1X,3(F8.3,2X))
          DO 22 I=1,NMCASE(J)
           READ(1,212),A,B,C,D,E,F
           MCASE(J,I,1)=A
           MCASE(J,I,2)=B
           MCASE(J,I,3)=C
           MCASE(J,I,4)=D
           MCASE(J,I,5)=E
22         MCASE(J,I,6)=F
212      FORMAT(1X,6(F8.3,2X))
20       CONTINUE
C
C     READ IN THE LOAD COMBINATION DATA
         DO 23 J=1,NCOMB
          L=J+5
          READ(1,209),TITLE
          NAME(L)=TITLE
          READ(1,213),A,B,C,D,E
          COMB(J,1)=A
          COMB(J,2)=B
          COMB(J,3)=C
          COMB(J,4)=D
23        COMB(J,5)=E
213      FORMAT(1X,5(F8.4,2X))
         DO 24 J=1,ACASES
          READ(1,214),I
          ACTLIST(J)=I
24       CONTINUE
214      FORMAT(1X,I3)
         CLOSE(UNIT=1)
C
         PRINT*,'RESTORE COMPLETED'
         PRINT*,'>ENTER THE DISTANCE ACROSS THE SCREEN'
         READ*,DIST
         DIST=DIST*1.25
         PRINT*,'>ENTER THE ROUNDING INCREMENT'
         READ*,ROUND
         PRINT*,'LOCATE THE ORIGIN OF THE GLOBAL AXIS'
         CALL WINDOW(0.,DIST,0.,DIST)
         CALL VWPORT(5.,105.,0.,100.)
         CALL MOVE(0.,0.)
         CALL DRAW(0.,DIST)
```

```
      CALL DRAW(DIST,DIST)
      CALL DRAW(DIST,0.)
      CALL DRAW(0.,0.)
      CALL LOCATE(1,X,Y,IGOT,IDAT)
      CALL CMCLOS
      CALL CMOPEN
      ZX=-X
      WX=DIST+ZX
      ZY=-Y
      WY=DIST+ZY
C     LOAD LOAD CASE 1 AS THE WORKING LOAD CASE
      JLTALLY=NJCASE(1)
      MLTALLY=NMCASE(1)
      DO 30 I=1,JLTALLY
       JLOAD(I,1)=JCASE(1,I,1)
       JLOAD(I,2)=JCASE(1,I,2)
       JLOAD(I,3)=JCASE(1,I,3)
30    CONTINUE
      DO 31 I=1,MLTALLY
       DO 32 K=1,6
32       MLOAD(I,K)=MCASE(1,I,K)
31    CONTINUE
      LCASE=1
      RETURN
      END
```

```
C       THIS SUBROUTINE IS TO OUTPUT THE RESULTS OF THE PROBLEM
        SUBROUTINE RESULT(CASES,NCOMB)
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
        COMMON /RELEASE/ MREL,SREL,STALLY
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT,MREL(40),SREL(40),STALLY
        REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
        REAL  FEMDIS(5,40,6)
        REAL SK
        CHARACTER*30 TITLE
        CHARACTER*4 PAR
        CHARACTER*8 FIL
        INTEGER I,J,K,L,CASES,JJ,JJJ
        PRINT*,'RESULT OUTPUT SECTION'
        PRINT*,'ENTER A NAME FOR THE OUTPUT FILE --  8 CHARACTERS MAX'
        READ 102,FIL
102     FORMAT(A8)
        PRINT*,'ENTER A JOB TITLE FOR THE OUTPUT -- 30 CHARACTERS MAX'
        READ 101,TITLE
101     FORMAT(A30)
        OPEN(UNIT=8,FILE=FIL,STATUS='NEW')
        WRITE(8,198)
198     FORMAT(1X,'JOB TITLE:')
        WRITE(8,199)
199     FORMAT(1X,60('*'))
        WRITE(8,197),TITLE
197     FORMAT(1X,'*',14X,A30,14X,'*')
        WRITE(8,199)

        WRITE(8,104)
104     FORMAT(' ',' ')
1       PRINT*,'>>NEXT RESULT OR EXIT'
        READ 100,PAR
100     FORMAT(A4)
        K=INDEX('FORC SUPP DISP SECF SECD HELP EXIT',PAR)
        IF(K.EQ.0) THEN
         PRINT*,'*** OUTPUT WHAT ?? ***'
         GOTO 1
        END IF
        K=(K+4)/5
        GOTO (10,20,30,40,50,60,70),K
C    MEMBER END FORCES
10      PRINT*,'OUTPUTING MEMBER END FORCES'
        WRITE(8,106)
106     FORMAT(' ',20X,21('-'))
        WRITE(8,110),'* MEMBER END FORCES *'
        WRITE(8,106)
110     FORMAT(' ',20X,A21)
        WRITE(8,105)
105     FORMAT(' ',60('-'))
        WRITE(8,111),'MEMBER #','LOAD','JOINT #','AXIAL','SHEAR','MOMENT'
111     FORMAT(' ',A8,3X,A4,3X,A7,2X,A5,7X,A5,7X,A6)
        WRITE(8,112)
112     FORMAT(' ',11X,'CASE')
        WRITE(8,105)
        WRITE(8,104)
        DO 11 I=1,TALLY
         WRITE(8,113),I
113      FORMAT(' ',3X,I3)
```

```
        JJJ=0
         DO 12 JJ=1,CASES+NCOMB
          J=JJ
          IF(JJ.GT.CASES) THEN
               JJJ=JJJ+1
               J=JJJ+5
          END IF
          WRITE(8,114),J,MT(I,1),EMCASE(J,I,1),EMCASE(J,I,2),EMCASE(J,I,3)
114       FORMAT(' ',12X,I2,6X,F2.0,3X,3(F10.2,2X))
          WRITE(8,115),MT(I,2),EMCASE(J,I,4),EMCASE(J,I,5),EMCASE(J,I,6)
115       FORMAT(' ',20X,F2.0,3X,3(F10.2,2X))
12       CONTINUE
11      CONTINUE
       GOTO 1
C     SUPPORT OUTPUT
20     PRINT*,'OUTPUTING SUPPORT REACTIONS'
       WRITE(8,104)
       WRITE(8,106)
       WRITE(8,110),'* SUPPORT REACTIONS *'
       WRITE(8,106)
       WRITE(8,104)
       WRITE(8,120),' JOINT #','LOAD','FORCE X','FORCE Y','MOMENT Z'
120    FORMAT(' ',A8,3X,A4,3X,A7,5X,A7,4X,A8)
       WRITE(8,112)
       WRITE(8,105)
       DO 21 I=1,NT
        IF(SREL(I).EQ.0) GOTO 21
        WRITE(8,121),I
121     FORMAT(' ',3X,I2)
        JJJ=0
        DO 22 JJ=1,CASES+NCOMB
         J=JJ
         IF(JJ.GT.CASES) THEN
               JJJ=JJJ+1
               J=JJJ+5
         END IF
         WRITE(8,122),J,SUPCASE(J,I,1),SUPCASE(J,I,2),SUPCASE(J,I,3)
122      FORMAT(' ',7X,I2,2X,3(F10.4,2X))
22      CONTINUE
21     CONTINUE
       GOTO 1
C     JOINT DISPLACEMENT
30     PRINT*,'OUTPUTING JOINT DISPLACEMENTS'
       WRITE(8,104)
       WRITE(8,106)
       WRITE(8,110),' JOINT DISPLACEMENTS '
       WRITE(8,106)
       WRITE(8,120),' JOINT #','LOAD','TRANS X','TRANS Y','ROTATE Z'
       WRITE(8,112)
       WRITE(8,105)
       DO 31 I=1,NT
        WRITE(8,121),I
        JJJ=0
        DO 32 JJ=1,CASES+NCOMB
         J=JJ
         IF(JJ.GT.CASES) THEN
               JJJ=JJJ+1
               J=JJJ+5
         END IF
         K=(I-1)*3+1
```

```
         WRITE(8,122),J,ACT(J,K),ACT(J,K+1),ACT(J,K+2)
32      CONTINUE
31     CONTINUE
       GOTO 1
C    SECTIONAL MEMBER FORCES
40     PRINT*,'OUTPUTING MEMBER SECTION FORCES'
       WRITE(8,104)
       WRITE(8,106)
       WRITE(8,110),'MEMBER SECTION FORCES'
       WRITE(8,106)
       WRITE(8,111),'MEMBER #','LOAD','SECTION','AXIAL','SHEAR','MOMENT'
       WRITE(8,112)
       WRITE(8,105)
       DO 41 I=1,TALLY
        JJJ=0
        DO 42 JJ=1,CASES+NCOMB
          J=JJ
          IF(JJ.GT.CASES) THEN
               JJJ=JJJ+1
               J=JJJ+5
          END IF
         WRITE(8,141),I,J
141      FORMAT(' ',3X,I2,7X,I2)
        DO 43 K=1,21,2
         SK=(K-1)*.05
         WRITE(8,143),SK,SECTFORC(J,I,2,K),SECTFORC(J,I,3,K)
143      FORMAT(' ',19X,F4.2,12X,2(F10.2,2X))
C       WRITE(8,142),SK,SECTFORC(J,I,1,K),SECTFORC(J,I,2,K),SECTFORC(J,I,3,K)
C142    FORMAT(' ',19X,F4.2,3(F10.2,2X))
43      CONTINUE
42      CONTINUE
41     CONTINUE
        GOTO 1
50     GOTO 1
60     PRINT*,'RESULT OUTPUT HELP SECTION -- COMMANDS AVAILABLE'
       PRINT*,' FORCE  SUPPORT  DISPLACEMENTS  '
       PRINT*,'  SECF  SECD  HELP  EXIT'
        GOTO 1
70     CLOSE(UNIT=8)
       RETURN
       END
```

```
C     THIS SUBROUTINE WILL STORE THE STRUCTURAL DATA
C     FOR FUTURE RECALL INTO THE PROGRAM.. VIA
C     THE SUBROUTINE "RESTORE"
C
          SUBROUTINE SAVE(NAME)
          COMMON /GEOM/   MT,TALLY,NLOC,NT
          COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
          COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
          COMMON /RELEASE/MBREL,SREL,STALLY
          REAL MT(40,12),NLOC(40,2)
          INTEGER TALLY,NT
          REAL MCASE(5,40,6),JCASE(5,40,3)
          INTEGER CASES,NMCASE(5),NJCASE(5)
          REAL COMB(5,5)
          INTEGER NCOMB,ACTLIST(10),ACASES
          INTEGER MBREL(40),SREL(40),STALLY
          INTEGER I,J,K,L
          CHARACTER*8 VAXNAME
          CHARACTER*30 TITLE,NAME(10)
          PRINT*,'>>SAVE ROUTINE:'
          PRINT*,'>>>ENTER A VAX FILE NAME -- 8 CHAR MAX'
          READ 100,VAXNAME
100       FORMAT(A8)
          PRINT*,'>>>ENTER A TITLE FOR THIS PROBLEM -- 30 CHAR MAX'
          READ 101,TITLE
101       FORMAT(A30)
          OPEN(UNIT=1,FILE=VAXNAME,STATUS='NEW')
          WRITE(1,201),TITLE
201       FORMAT(' ',A30)
202       FORMAT(' ',I3,A30)
C      WRITE OUT THE STRUCTURAL PARAMETER COUNTERS
          WRITE(1,202),NT,    ' NUMBER OF JOINTS             '
          WRITE(1,202),TALLY,' NUMBER OF MEMBERS            '
          WRITE(1,202),CASES,' NUMBER OF LOAD CASES         '
          WRITE(1,202),NCOMB,' NUMBER OF LOAD COMBINATIONS  '
          WRITE(1,202),ACASES,' NUMBER OF ACTIVE CASES       '
C
          DO 10 J=1,NT
10         WRITE(1,203),NLOC(J,1),NLOC(J,2)
203       FORMAT(' ',F10.4,2X,F10.4)
          DO 11 M=1,TALLY
11         WRITE(1,204),MT(M,1),MT(M,2),MT(M,5),MT(M,6),MT(M,7),MT(M,8)
204       FORMAT(' ',F3.0,2X,F3.0,2X,F8.3,2X,F8.0,2X,F8.3,2X,F8.3)
          DO 12 M=1,TALLY
12         WRITE(1,205),MBREL(M)
205       FORMAT(' ',I3)
          DO 13 J=1,NT
13         WRITE(1,205),SREL(J)
C
          DO 20 J=1,CASES
           WRITE(1,209),NAME(J)
           WRITE(1,210),NJCASE(J),NMCASE(J)
209       FORMAT(' ',A30)
210       FORMAT(' ',I3,2X,I3,'   NJCASE,NMCASE')
           DO 21 I=1,NJCASE(J)
21          WRITE(1,211),JCASE(J,I,1),JCASE(J,I,2),JCASE(J,I,3)
211       FORMAT(' ',3(F8.3,2X))
           DO 22 I=1,NMCASE(J)
22          WRITE(1,212),(MCASE(J,I,K),K=1,6)
212       FORMAT(' ',6(F8.3,2X))
```

```
20      CONTINUE
C
        DO 23 J=1,NCOMB
         L=J+5
         WRITE(1,209),NAME(L)
23       WRITE(1,213),(COMB(J,I),I=1,5)
213     FORMAT(' ',5(F8.4,2X))
        DO 24 J=1,ACASES
24       WRITE(1,214),ACTLIST(J)
214     FORMAT(' ',I3)
        CLOSE(UNIT=1)
C
        PRINT*,'SAVE COMPLETED'
        RETURN
        END
```

```
*******************************************************************************************


C  THIS SUBROUTINE WORKS IN CONJUNCTION WITH DRWLOADS TO
C   DRAW THE ARROWS IN THE CORRECT DIRECTION AT THE
C   GIVEN LOCATION
      SUBROUTINE DRWARROW(X,Y,DIR,MAG,DEG,ZX,WX,ZY,WY)
      REAL X,Y,MAG,DEG
      INTEGER DIR
      REAL ZX,WX,ZY,WY
      REAL AEX(4),AWX(4),ANX(4),ASX(4),AEY(4),AWY(4),ANY(4),ASY(4)
      DATA AEX/0,-1.5,0,1.5/
      DATA AEY/0,.75,-1.5,.75/
      DATA AWX/0,1.5,0,-1.5/
      DATA AWY/0,.75,-1.5,.75/
      DATA ANX/0,-.75,1.5,-.75/
      DATA ANY/0,-1.5,0.,1.5/
      DATA ASX/0,-.75,1.5,-.75/
      DATA ASY/0,1.5,0.,-1.5/
      CALL WINDOW(ZX,WX,ZY,WY)
      CALL VWPORT(5.,105.,0.,100.)
      CALL MOVE(X,Y)
      CALL PIVOT(X,Y)
      CALL WINDOW(0.,100.,0.,100.)
      CALL VWPORT(5.,105.,0.,100.)
      IF(DEG.NE.0) CALL ROTATE(DEG,DEG)
      CALL VECREL
      IF(DIR.EQ.1) THEN        ! FX
       IF(MAG.GT.0) THEN
        CALL DRAW(-5.,0.)
        CALL MOVE(5.,0.)
        CALL POLY(4,AEX,AEY)
       ELSE
         CALL DRAW(5.,0.)
         CALL MOVE(-5.,0.)
        CALL POLY(4,AWX,AWY)
       END IF
      ELSE IF(DIR.EQ.2) THEN   !FY
       IF(MAG.GT.0) THEN
        CALL DRAW(0.,-5.)
        CALL MOVE(0., 5.)
        CALL POLY(4,ANX,ANY)
       ELSE
        CALL DRAW(0., 5.)
        CALL MOVE(0.,-5.)
        CALL POLY(4,ASX,ASY)
       END IF
      ELSE             ! MZ
       CALL ARC(3.,0.,180.)
       IF(MAG.LT.0) THEN
        CALL MOVE(-3.,0.)
        CALL POLY(4,ASX,ASY)
       ELSE
        CALL MOVE(3.,0.)
        CALL POLY(4,ASX,ASY)
       END IF
      END IF
      CALL VECABS
      IF(DEG.NE.0) CALL ROTATE(-DEG,-DEG)
      CALL CMCLOS
      CALL CMOPEN
      CALL WINDOW(ZX,WX,ZY,WY)
      CALL VWPORT(5.,105.,0.,100.)
      CALL PIVOT(0.,0.)
      RETURN
      END
```

```
C          THIS SUBROUTINE WILL DRAW THE LOADS ON THE STRUCTURE
        SUBROUTINE DRWLOAD
        COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        COMMON /LOADONE/MLTALLY,JLTALLY,MLOAD,JLOAD
        REAL ZX,WX,ZY,WY,ROUND
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        REAL MLOAD(40,6),JLOAD(40,3)
        INTEGER MLTALLY,JLTALLY
        REAL AUX(4),AUY(4),ADX(4),ADY(4),MAG,LA,LB,L,LP,K,DEG
        REAL XP(6),YP(6),X,Y,XS,YS,XL,YL,XP1,XP2,YP1,YP2
        INTEGER I,J,TYPE,DIR
        DATA AUX/0,-1.5,3.,-1.5/
        DATA AUY/0,-1.5,0.,1.5/
        DATA ADX/0,-1.5,3,-1.5/
        DATA ADY/0,1.5,0,-1.5/
        DO 10 I=1,JLTALLY
        DEG=0
         JNUM=JLOAD(I,1)
         DIR =JLOAD(I,2)
         MAG =JLOAD(I,3)
        X=NLOC(JNUM,1)
        Y=NLOC(JNUM,2)
        CALL DRWARROW(X,Y,DIR,MAG,DEG,ZX,WX,ZY,WY)
10      CONTINUE
        DO 100 I=1,MLTALLY
        DEG=0
         MNUM=MLOAD(I,1)
         TYPE=MLOAD(I,2)
         DIR =MLOAD(I,3)
         MAG =MLOAD(I,4)
         LA  =MLOAD(I,5)
         LB  =MLOAD(I,6)
        XS=NLOC(MT(MNUM,1),1)
        YS=NLOC(MT(MNUM,1),2)
        XL=NLOC(MT(MNUM,2),1)
        YL=NLOC(MT(MNUM,2),2)
        L =MT(MNUM,5)/12
        C=(XL-XS)/L
        S=(YL-YS)/L
        IF(C.LT.0.001) C=0.001
        IF(S.LT.0.001.AND.S.GT.-0.001) S=0.0
        XP1=XS+(LA*C*L)
        YP1=YS+(LA*S*L)
        IF(LB.NE.0) THEN
         XP2=XS+(LB*C*L)
         YP2=YS+(LB*S*L)
        END IF
         IF(C.LT.0.001.AND.C.GT.-0.001) THEN
          IF(S.LT.0) THEN
           DEG=-90.
          ELSE
          DEG=90.
         END IF
          GOTO 11
         END IF
         DEG= 57.2957795131*(ATAN(S/C))
11      IF(TYPE.EQ.1.OR.TYPE.EQ.3) THEN    ! GO DIRECTLY TO ARROW
         CALL DRWARROW(XP1,YP1,DIR,MAG,DEG,ZX,WX,ZY,WY)
```

```
        ELSE    ! IT IS A UNIFORM LOAD
        LP=SQRT((XP1-XP2)**2+(YP1-YP2)**2)
         M=1
              DO 120 K=.2,.8,.2
                 M=M+1
                 XP(M)=XP1+(LP*C*K)
                 YP(M)=YP1+(LP*S*K)
120     CONTINUE
                 XP(1)=XP1
                 YP(1)=YP1
                 XP(6)=XP2
                 YP(6)=YP2
C        DRAW THE 6 ARROWS ACCORDING TO DEGEES AND SIGN
        DO 121 K=1,6
                 CALL MOVE(XP(K),YP(K))
                 CALL PIVOT(XP(K),YP(K))
        CALL WINDOW(0.,100.,0.,100.)
        CALL VWPORT(5.,105.,0.,100.)
                 IF(DEG.NE.0) CALL ROTATE(DEG,DEG)
                 CALL VECREL
        IF(MAG.GT.0) THEN   !USE ARROW UP
          CALL POLY(4,AUX,AUY)
          ELSE
          CALL POLY(4,ADX,ADY)
         END IF
        CALL VECABS
         IF(DEG.NE.0) CALL ROTATE(-DEG,-DEG)
         CALL WINDOW(ZX,WX,ZY,WY)
         CALL VWPORT(5.,105.,0.,100.)
         CALL PIVOT(0.,0.)
        CALL CMCLOS
        CALL CMOPEN
121     CONTINUE
        END IF
100     CONTINUE
        RETURN
        END
```

```
      SUBROUTINE DRWMREL(X1,X2,Y1,Y2,IREL,L)
      COMMON /SCREEN/ZX,WX,ZY,WY,ROUND
      REAL X,Y,ZX,ZY,WX,WY,L,CX1,CX2,CY1,CY2,LL
      INTEGER IREL
C   DRAW THE CIRCLES 10% FROM EACH END
      LL=L*.10
      C=(X2-X1)/L
      S=(Y2-Y1)/L
      CX1=X1+(C*LL)
      CX2=X2-(C*LL)
      CY1=Y1+(S*LL)
      CY2=Y2-(S*LL)
      IF(IREL.EQ.4) GOTO 10
C   IREL HOLDS THE RELEASE CASE CODE
      CALL MOVE(CX1,CY1)
      CALL WINDOW(0.,100.,0.,100.)
      CALL VWPORT(5.,105.,0.,100.)
      CALL ARC(.75,0.,360.)
      CALL WINDOW(ZX,WX,ZY,WY)
      CALL VWPORT(5.,105.,0.,100.)
10    IF(IREL.EQ.3) GOTO 20
      CALL MOVE(CX2,CY2)
      CALL WINDOW(0.,100.,0.,100.)
      CALL VWPORT(5.,105.,0.,100.)
      CALL ARC(.75,0.,360.)
      CALL WINDOW(ZX,WX,ZY,WY)
      CALL VWPORT(5.,105.,0.,100.)
20    CALL CMCLOS
      CALL CMOPEN
      RETURN
      END
```

```
SUBROUTINE DRWSUPP(X,Y,IFIX)
COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
REAL ZX,WX,ZY,WY,ROUND
REAL SXX(7),SXY(7),X,Y,SMZX(15),SMZY(15)
INTEGER ISX,ISY,ISZ,IFIX
REAL SYX(7),SYY(7)
DATA SXY/0,-.5,1,-.5,-.15,.3,-.15/
DATA SXX/0,2,0,-2,2,0,-2/
DATA SYX/0,-.5,1,-.5,-.15,.3,.15/
DATA SYY/0,-2,0,2,-2,0,2/
DATA SMZX/0,-2,0,2,-2,0,2,-2,-2,0,2,-2,0,2,2/
DATA SMZY/0,.5,-1,.5,.15,-.3,.15,0,.5,-1,.5,.15,-.3,.15,0/
ISX=0
ISY=0
ISZ=0
IF(IFIX.GE.100) THEN
 ISX=1
 IFIX=IFIX-100
END IF
IF(IFIX.GE.10)  THEN
 ISY=1
 IFIX=IFIX-10
END IF
IF(IFIX.GE.1)   ISZ=1
CALL MOVE(X,Y)
CALL WINDOW(0.,100.,0.,100.)
CALL VWPORT(5.,105.,0.,100.)
CALL VECREL
IF(ISX.EQ.1)THEN
 CALL POLY(7,SXX,SXY)
END IF
IF(ISY.EQ.1)THEN
  CALL POLY(7,SYX,SYY)
 END IF
 IF(ISZ.EQ.1)THEN
       CALL POLY(15,SMZX,SMZY)
END IF
CALL WINDOW(ZX,WX,ZY,WY)
CALL VWPORT(5.,105.,0.,100.)
CALL VECABS
CALL CHCLOS
CALL CHOPEN
RETURN
END
```

```
C    THIS IS THE SUBROUTINE TO DRIVE THE GRAPHIC ROUTINES OF THIS
C         PROGRAMS
          SUBROUTINE GRAPHIK(SET)
          COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
          COMMON /GEOM/   MT,TALLY,NLOC,NT
          COMMON /LOADONE/MLTALLY,JLTALLY,MLOAD,JLOAD
          COMMON /RELEASE/MBREL,SREL,STALLY
          REAL ZX,WX,ZY,WY,ROUND
          REAL MT(40,12),NLOC(40,2)
          INTEGER TALLY,NT
          REAL MLOAD(40,6),JLOAD(40,3)
          INTEGER MLTALLY,JLTALLY
          INTEGER MBREL(40),SREL(40),STALLY
          REAL X,Y,X1,X2,Y1,Y2,L,MIDX,MIDY
          REAL XW(2),YW(2),DISTX,DISTY
          INTEGER I,J,K,M,IFIX,IDAT(2),IGOT(2)
          CHARACTER*4 PAR2
          LOGICAL RED,SET(7)
          CHARACTER*15 REZ(7)
          REZ(1)='JOINTS NUMBERS '
          REZ(2)='MEMBER NUMBERS '
          REZ(3)='        MEMBERS '
          REZ(4)='M & J  LOADS    '
          REZ(5)='MEMBER RELEASE '
          REZ(6)='    SUPPORTS    '
99        FORMAT(A4)
          CALL NEWPAG
1         CALL HOME
          CALL CMCLOS
          CALL CMOPEN
          PRINT*,'>>NEXT GRAPHIC OR EXIT'
          READ 99,PAR2
2         RED=.FALSE.
          K=INDEX('SETU STRU REDR ZOOM HELP EXIT',PAR2)
          IF(K.EQ.0) GOTO 1
          K=(K+4)/5
          GOTO(10,20,30,40,50,60),K
C    10 IS SETUP
10        PRINT*,'>>>NEXT SETUP OR NEXT GRAPHIC'
          READ 99,PAR2
          J=INDEX('JNUM MNUM MEMB LOAD MREL SUPP',PAR2)
          IF(J.EQ.0)GOTO 2
          J=(J+4)/5
          IF(SET(J)) THEN
           SET(J)=.FALSE.
           PRINT 101,REZ(J)
          ELSE
           SET(J)=.TRUE.
           PRINT 100,REZ(J)
          END IF
100       FORMAT(' ',A15,'LABELED')
101       FORMAT(' ',A15,'NOT LABELED')
          GOTO 10
C    BEGIN THE STRUCTURE DRAWING
20        CALL PAGE(ZX,WX,ZY,WY,ROUND)
          CALL CMCLOS
          CALL CMOPEN
C       CHECK FOR JOINT NUMBER
          IF(RED) THEN
           GOTO 200
```

```
      ELSE
       IF(SET(1)) GOTO 200
      END IF
      GOTO 201
200    CALL TXSIZE(3,0,0)
       CALL TXICUR(1)
       DO 21 I=1,NT
        CALL MOVE(NLOC(I,1),NLOC(I,2))
        CALL INUMBR(I,3)
21     CONTINUE
C    CHECK FOR MEMBER NUMBER
201   CALL CMCLOS
      CALL CMOPEN
      IF(RED) THEN
       GOTO 210
      ELSE
       IF(SET(2)) GOTO 210
      END IF
      GOTO 211
210    CALL TXSIZE(2,0,0)
       CALL TXICUR(1)
       DO 22 I=1,TALLY
        IF(MT(I,1).EQ.0) GOTO 22
        MIDX=NLOC(MT(I,1),1)+(NLOC(MT(I,2),1)-NLOC(MT(I,1),1))/4
        MIDY=NLOC(MT(I,1),2)+(NLOC(MT(I,2),2)-NLOC(MT(I,1),2))/4
        CALL MOVE(MIDX,MIDY)
        CALL INUMBR(I,3)
22     CONTINUE
211   CALL CMCLOS
      CALL CMOPEN
      IF(RED)THEN
       GOTO 220
      ELSE
       IF(SET(3)) GOTO 220
      END IF
      GOTO 221
220    DO 23 I=1,TALLY
        IF(MT(I,1).EQ.0) GOTO 23
        CALL MOVE(NLOC(MT(I,1),1),NLOC(MT(I,1),2))
        CALL DRAW(NLOC(MT(I,2),1),NLOC(MT(I,2),2))
23     CONTINUE
C    CHECK FOR DRAW LOADS
221   CALL CMCLOS
      CALL CMOPEN
      IF(RED)THEN
       GOTO 230
      ELSE
       IF(SET(4)) GOTO 230
      END IF
      GOTO 231
230   CALL DRWLOAD
231   CALL CMCLOS
      CALL CMOPEN
      IF(RED) GOTO 110
C     CHECK FOR DRAW MEMBER RELEASES
      IF(SET(5))THEN
250    DO 25 I=1,TALLY
        IF(MBREL(I).NE.0) THEN
        IREL=MBREL(I)+1
         X1=NLOC(MT(I,1),1)
```

```
         Y1=NLOC(MT(I,1),2)
         X2=NLOC(MT(I,2),1)
         Y2=NLOC(MT(I,2),2)
         L=MT(I,5)/12
         CALL DRWMREL(X1,X2,Y1,Y2,IREL,L)
         END IF
25      CONTINUE
       END IF
       CALL CMCLOS
       CALL CMOPEN
       IF(RED)GOTO 110
C    CHECK FOR DRAW SUPPORTS
       IF(SET(6))THEN
260     DO 26 I=1,NT
         IF(SREL(I).NE.0)THEN
          X=NLOC(I,1)
          Y=NLOC(I,2)
          IFIX=SREL(I)
           CALL DRWSUPP(X,Y,IFIX)
         END IF
26      CONTINUE
       END IF
       CALL CMCLOS
       CALL CMOPEN
       IF(RED)GOTO 110
29     RED=.FALSE.
       GOTO 1
C   THIS IS THE REDRAW BY COMMAND PART
30     RED=.TRUE.
       GOTO 20
110    PRINT*,'>>>NEXT REDRAW OR NEXT GRAPHIC'
       READ 99,PAR2
       M=INDEX('LOAD MREL SUPP HELP',PAR2)
       IF(M.EQ.0)GOTO 2
       M=(M+4)/5
       GOTO(230,250,50),M
111    PRINT*,'REDRAW/GRAPHIC HELP SECTION: '
       PRINT*,' LOAD MREL SUPP HELP EXIT'
       PRINT*,' OR ANY GRAPHIC COMMAND'
       GOTO 110
C    TWO POINT ZOOM SECTION
40     PRINT*,'ZOOM : LOCATE 2 POINTS TO FRAME THE WINDOW'
       PRINT*,' THE LOWER LEFT AND THE UPPER RIGHT CORNER'
       CALL LOCATE(2,XW,YW,IGOT,IDAT)
       DISTX= ABS(XW(1)-XW(2))
       DISTY= ABS(YW(1)-YW(2))
       IF(DISTX.GT.DISTY) THEN
          ZX=XW(1)
          WX=XW(2)
          ZY=YW(1)
          WY=YW(1)+DISTX
        ELSE
          ZX=XW(1)
          WX=XW(1)+DISTY
          ZY=YW(1)
          WY=YW(2)
        END IF
        GOTO 1
50      PRINT*,'GRAPHIC HELP SECTION:'
        PRINT*,' SETup  WILL KEY on OR off--'
```

```
      PRINT*,'  JNUM MNUM MEMB LOAD MREL SUPP'
      PRINT*,' STRUcture  WILL DRAW THE STRUCTURE ACCORDING TO  SETUP'
      PRINT*,' REDRaw  WILL REDRAW THE BASICS AND ALLOW ADDING THE EXTRAS'
      PRINT*,'COMMANDS AVAILABLE: '
      PRINT*,'SETU STRU REDR ZOOM HELP EXIT'
      GOTO 1
60    PRINT*,'EXIT GRAPHIC SECTION'
      RETURN
      END


      SUBROUTINE PAGE(ZX,WX,ZY,WY,ROUND)
C     THIS SUBROUTINE WILL CALL A NEW PAGE AND INITIALIZE THE THINGS
C   WINDOW -- VWPORT -- BOX AROUND VWPORT -- TIC MARKS AND NUMBERS
CCC   COMMON SCREEN
      REAL WX,WY,ZX,ZY,TIC,ROUND,RTEN,BOXX,BOXY
      INTEGER ITICX,ITICY
      REAL TIC2,R
      CALL NEWPAG
      CALL WINDOW(ZX,WX,ZY,WY)
      CALL VWPORT(5.,105.,0.,100.)
C  DRAW AXIS
      CALL DASHPT(3)
      CALL MOVE(0.,ZY)
      CALL DRAW(0.,WY)
      CALL MOVE(ZX,0.)
      CALL DRAW(WX,0.)
      CALL DASHPT(0)
C  DRAW TIC MARKS...THEY ARE 2.5% OF THE WINDOW HEIGHT
      TIC=(WX-ZX)*.015
      TIC2=2*TIC
C     TIC AT DIST ROUND * 5
C     DRAW BOX
      CALL MOVE(ZX,ZY)
      CALL DRAW(ZX,WY)
      CALL DRAW(WX,WY)
      CALL DRAW(WX,ZY)
      CALL DRAW(ZX,ZY)
      ITICX=(ZX/(ROUND*5))
      RTEN=ROUND*5
      ITICX=(ZX/RTEN)
      ITICX=ITICX*RTEN
      ITICY=(ZY/RTEN)
      ITICY=ITICY*RTEN
      CALL TXICUR(5)
      CALL TXFCUR(1)
      CALL TXSIZE(4,0.,0.)
      DO 10 R=ITICX,WX,RTEN
       CALL MOVE(R,ZY)
       CALL DRAW(R,ZY+TIC)
       CALL MOVE(R,ZY+TIC2)
       CALL RNUMBR(R,2,8)
10    CONTINUE
      DO 20 R=ITICY,WY,RTEN
       CALL MOVE(ZX,R)
       CALL DRAW(ZX+TIC,R)
       CALL MOVE(ZX+TIC2,R)
       CALL RNUMBR(R,2,8)
20    CONTINUE
      CALL TXSIZE(3,0,0)
      RETURN
      END
```

```
C    SUBROUTINE TO ZOOM IN ON THE SCREEN AND PAN TOO
        SUBROUTINE ZOOMIO
        COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
        REAL ZX,WX,ZY,WY,ROUND
1       PRINT*,'ENTER THE SCREEN SCALE FACTOR:'
        PRINT*,'  LESS THAN 1.0 TO ZOOM IN'
        PRINT*,'  GREATER THAN 1.0 TO ZOOM OUT'
        PRINT*,'THEN LOCATE THE CROSSHAIRS FOR THE NEW CENTER OF THE PICTURE'
        READ*,SCAL
        IF(SCAL.LT.0.1.OR.SCAL.GT.10)GOTO 1
        CALL LOCATE(1,X,Y,IGOT,IDAT)
        DIST=(WX-ZX)*SCAL/2
        ZX=X-DIST
        WX=DIST*2+ZX
        ZY=Y-DIST
        WY=DIST*2+ZY
        CALL WINDOW(ZX,WX,ZY,WY)
        CALL VWPORT(5.,105.,0.,100.)
        RETURN
        END

C       SUBROUTINE TO REDRAW THE SCREEN AND THE STRUCTURE
C       AXIS,TIC MARKS AND NODE AND MEMBER NUMBERS
        SUBROUTINE REDR
        COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
        COMMON /GEOM  / MT,TALLY,NLOC,NT
        REAL ZX,WX,ZY,WY,ROUND
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        REAL XI,YI,MIDX,MIDY,XJ,YJ
        INTEGER I,J,K
C    REDRAW THE PAGE AND AXIS AND TIC MARKS
        CALL PAGE(ZX,WX,ZY,WY,ROUND)
        CALL TXSIZE(3,0,0)
        CALL TXICUR(1)
C    DRAW THE NODES
        DO 10 I=1,NT
         CALL MOVE(NLOC(I,1),NLOC(I,2))
         CALL INUMBR(I,3)
10      CONTINUE
        CALL TXSIZE(2,0,0)
C    DRAW THE MEMBERS AND NUMBER THEM
        DO 20 I=1,TALLY
         IF(MT(I,1).EQ.0) GOTO 20
        J=MT(I,1)
        K=MT(I,2)
        XI=NLOC(J,1)
        YI=NLOC(J,2)
        XJ=NLOC(K,1)
        YJ=NLOC(K,2)
C    CALCULATE THE MIDPOINT OF THE MEMBER AND NUMBER IT
        MIDX=(XJ-XI)/4+XI
        MIDY=(YJ-YI)/4+YI
        CALL MOVE(XI,YI)
        CALL DRAW(XJ,YJ)
        CALL MOVE(MIDX,MIDY)
        CALL INUMBR(I,3)
20      CONTINUE
        CALL TXSIZE(3,0,0)
        CALL TXICUR(1)
        CALL HOME
        CALL CMCLOS
        CALL CMOPEN
        RETURN
        END
```

```
C        THIS IS THE SUBROUTINE THAT DRIVES THE SHEAR AND MOMENT
C        CALCULATIONS FOR ALL MEMBERS AND ALL LOADS
C      NOTE:  SIGN CONVENTIONS DIFFER ::: FEMDIS -(FROM FORCES) EXPECTS ALL
C        CCW MOMENTS AS "+"
C        BUT !!! MOMSHE EXPECTS THE DIRECTIONS OF THE MOMENTS TO BE THE
C        VALUES IN THE MOMENT DIAGRAM ... THEREFORE CHANGE M1=-FEMDIS(3)
C
         SUBROUTINE CASEMOSH
C
         COMMON /GEOM/   MT,TALLY,NLOC,NT
         COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
         COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
         COMMON /RELEASE/MBREL,SREL,STALLY
         REAL MT(40,12),NLOC(40,2)
         INTEGER TALLY,NT
         REAL MCASE(5,40,6),JCASE(5,40,3)
         INTEGER CASES,NMCASE(5),NJCASE(5)
         REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
         REAL  FEMDIS(5,40,6)
         INTEGER MBREL(40),SREL(40),STALLY
         REAL MEMMOM(21),MEMSHE(21),TMEMMOM(21),TMEMSHE(21)
         REAL J,L,MAG,A,B,M1,M2,S1,S2
         INTEGER K,I,MQ(40),TYPE,DIR,CASE
         INTEGER J1,J2,SP1,SP2,MR,JR1,JR2,LCASE,M
C   BEGIN BY DOING ALL LOAD CASES
         DO 1 LCASE=1,CASES
         DO 10 I=1,TALLY
          M1=-FEMDIS(LCASE,I,3)
          M2=FEMDIS(LCASE,I,6)
          S1=FEMDIS(LCASE,I,2)
          S2=FEMDIS(LCASE,I,5)
          DO 20 J=1,21
           SECTFORC(LCASE,I,3,J)=M1+(((-M1+M2)/20)*(J-1))
            SECTFORC(LCASE,I,2,J)=S1
20         CONTINUE
10       CONTINUE
         DO 30 I=1,TALLY
         CASE=MBREL(I)+1
         DO 35 M=1,21
          MEMSHE(M)=0
35        MEMMOM(M)=0
         L=MT(I,5)/12.0    ! NOW IN FT
         DO 40 K=1,NMCASE(LCASE)
          IF(MCASE(LCASE,K,1).EQ.I) THEN
          TYPE=MCASE(LCASE,K,2)
          DIR =MCASE(LCASE,K,3)
          MAG =MCASE(LCASE,K,4)     ! IN K,K,K/F
          A    =MCASE(LCASE,K,5)
          B    =MCASE(LCASE,K,6)
           IF(TYPE.EQ.1) THEN    ! CONCENTRATED
             IF(DIR.EQ.1) THEN   !FX MOM & SHE =0
              GOTO 40
             ELSE
                 CALL PYMOMSHE(CASE,MAG,A,L,TMEMMOM,TMEMSHE)
             END IF
            ELSE IF(TYPE.EQ.2) THEN   !UNIFORM
             IF(DIR.EQ.1) THEN        !WX  SHE & MOM =0
                 GOTO 40
             ELSE
```

```
            IF(A.EQ.0.AND.B.EQ.0.OR.A.EQ.0.AND.B.EQ.1) THEN
               CALL MYMOMSHE(CASE,MAG,A,B,L,TMEMMOM,TMEMSHE)
            ELSE
               CALL MPYMOMSHE(MAG,L,A,B,TMEMMOM,TMEMSHE,CASE)
            END IF
           END IF
          ELSE   ! APPLIED MOMENT
               CALL MMZ(CASE,MAG,A,L,TMEMMOM,TMEMSHE)
          ENDIF
         DO 50 M=1,21
               MEMMOM(M)=MEMMOM(M)+TMEMMOM(M)
50             MEMSHE(M)=MEMSHE(M)+TMEMSHE(M)
        END IF
40      CONTINUE
        DO 60 M=1,21
         SECTFORC(LCASE,I,3,M)=SECTFORC(LCASE,I,3,M)+MEMMOM(M)
         SECTFORC(LCASE,I,2,M)=SECTFORC(LCASE,I,2,M)+MEMSHE(M)
60      CONTINUE
30      CONTINUE
1       CONTINUE
        RETURN
        END
```

```
C    THIS SUBROUTINE WILL DRAW THE DEFLECTED SHAPE
C    OF THE STRUCTURE ACCORDING TO THE DIFFERENT LOAD CASES
C    AND COMBINATIONS
C
        SUBROUTINE DEFL(CASES,NCOMB,NAME)
C
        COMMON /SCREEN/  ZX,WX,ZY,WY,ROUND
        COMMON /GEOM/    MT,TALLY,NLOC,NT
        COMMON /FORC1/   SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
        REAL ZX,WX,ZY,WY,ROUND
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
        REAL FEMDIS(5,40,6)
        CHARACTER*30 NAME(10)
        CHARACTER*1 RES
        REAL XD1,XD2,YD1,YD2,MAX,DIST,FACT
        INTEGER I,J,K,N,N1,N2,CASES,NCOMB,M
C   FIND THE MAX DEFLECTION
        K=0
        DO 40 J=1,CASES+NCOMB
         M=J
         IF(J.GT.CASES) THEN
          K=K+1
          M=J+5
         END IF
         DO 30 I=1,NT*3
30        IF(ABS(ACT(M,I)).GT.MAX) MAX=ABS(ACT(M,I))
40      CONTINUE
C    THE MAX TRANSLATIONAL DEFLECTION WILL SCALE TO
C    2.5% OF THE SCREEN WIDTH
        DIST=.025*(WX-ZX)
        FACT=DIST*12/MAX
        CALL NEWPAG
        CALL TXSIZE(2,0,0)
        CALL TXICUR(2)
        CALL TXAM
        CALL TXFCUR(1)
        CALL CMCLOS
        CALL CMOPEN
2       PRINT*,'DEFLECTED STRUCTRUE ROUTINE'
        PRINT*,' ONLY JOINT DISPLACEMENTS'
C    PRINT LIST OF LOAD COMB AND CASES
        K=0
        DO 10 I=1,CASES+NCOMB
         J=I
         IF(I.GT.CASES) THEN
          K=K+1
          J=5+K
         END IF
        PRINT 100,J,NAME(J)
100     FORMAT(1X,2X,I3,2X,A30)
10      CONTINUE
1       PRINT*,'>>ENTER LOAD NUMBER'
        CALL CMCLOS
        CALL CMOPEN
        READ*,N
        IF(N.EQ.0) GOTO 2
        IF(N.LT.0) RETURN
        IF(N.LE.CASES) GOTO 11
```

```
      IF(N.GE.6.AND.N.LE.NCOMB+5) GOTO 11
      PRINT*,'*** BAD LOAD NUMBER ***'
      GOTO 1
11    I=I
      CALL REDR
      CALL DASHPT(3)
      DO 20 I=1,TALLY
       N1=MT(I,1)
       N2=MT(I,2)
       XD1=(ACT(N,((N1-1)*3+1)))/12*FACT
       XD2=(ACT(N,((N2-1)*3+1)))/12*FACT
       YD1=(ACT(N,((N1-1)*3+2)))/12*FACT
       YD2=(ACT(N,((N2-1)*3+2)))/12*FACT
        CALL MOVE((NLOC(N1,1)+XD1),(NLOC(N1,2)+YD1))
        CALL DRAW((NLOC(N2,1)+XD2),(NLOC(N2,2)+YD2))
        CALL CMCLOS
        CALL CMOPEN
20    CONTINUE
      CALL DASHPT(0)
      CALL MOVE(ZX+DIST,ZY+(DIST/2))
      CALL VECREL
      DIST=MAX*FACT/12
      CALL DRAW(DIST,0.0)
      CALL VECABS
      CALL TXICUR(1)
      CALL TXFCUR(2)
      CALL TXSIZE(3,0,0)
      CALL RNUMBR(MAX,3,6)
      CALL TEXT(17,'  Inch Deflection')
      CALL TXFCUR(1)
      CALL HOME
      CALL CMCLOS
      CALL CMOPEN
      GOTO 1
      END
```

```
C     THIS IS THE SUBROUTINE TO CALCULATE THE MAX AND MIN ENVELOPE
         SUBROUTINE ENVEL(CASES,TALLY,MEMMAX)
         COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
         COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
         INTEGER NCOMB,ACTLIST(10),ACASES
         REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
         REAL  FEMDIS(5,40,6),COMB(5,5)
         REAL MAXX,MAXY,MAXZ,MINX,MINY,MINZ,MEMMAX(40,3)
         INTEGER CASES,TALLY,LC,I,K,J
         DO 10 I=1,TALLY
          DO 20 K=1,21
           MAXX=-10000000
           MINX= 10000000
           MAXY=-10000000
           MINY= 10000000
           MAXZ=-10000000
           MINZ= 10000000
           X=0
           Y=0
           Z=0
            DO 30 J=1,ACASES
             LC=ACTLIST(J)
             X=SECTFORC(LC,I,1,K)
             Y=SECTFORC(LC,I,2,K)
             Z=SECTFORC(LC,I,3,K)
             IF(X.GT.MAXX) MAXX=X
             IF(X.LT.MINX) MINX=X
             IF(Y.GT.MAXY) MAXY=Y
             IF(Y.LT.MINY) MINY=Y
             IF(Z.GT.MAXZ) MAXZ=Z
             IF(Z.LT.MINZ) MINZ=Z
30         CONTINUE
                 SECTFORC(11,I,1,K)=MAXX
                 SECTFORC(12,I,1,K)=MINX
                 SECTFORC(11,I,2,K)=MAXY
                 SECTFORC(12,I,2,K)=MINY
                 SECTFORC(11,I,3,K)=MAXZ
                 SECTFORC(12,I,3,K)=MINZ
20       CONTINUE
          MAXX=-10000000
          MINX= 10000000
          MAXY=-10000000
          MINY= 10000000
          MAXZ=-10000000
          MINZ= 10000000
          X=0
          Y=0
          Z=0
        DO 15 K=1,21
         IF(ABS(SECTFORC(11,I,1,K)).GT.MAXX)  MAXX=ABS(SECTFORC(11,I,1,K))
         IF(ABS(SECTFORC(11,I,2,K)).GT.MAXY)   MAXY=ABS(SECTFORC(11,I,2,K))
         IF(ABS(SECTFORC(11,I,3,K)).GT.MAXZ)   MAXZ=ABS(SECTFORC(11,I,3,K))
         IF(ABS(SECTFORC(12,I,1,K)).GT.MAXX)   MAXX=ABS(SECTFORC(12,I,1,K))
         IF(ABS(SECTFORC(12,I,2,K)).GT.MAXY)   MAXY=ABS(SECTFORC(12,I,2,K))
         IF(ABS(SECTFORC(12,I,3,K)).GT.MAXZ)   MAXZ=ABS(SECTFORC(12,I,3,K))
15      CONTINUE
         MEMMAX(I,1)=ABS(MAXX)
         MEMMAX(I,2)=ABS(MAXY)
         MEMMAX(I,3)=ABS(MAXZ)    ! ALREADY IN F-K
10      CONTINUE
```

```
C    THIS IS A SUBROUTINE THAT WILL COMPLETE ALL OF THE LOAD
C    COMBINATIONS  BY FACTORING THE INDIVIDUAL LOAD CASES
C    AND SUMMING UP LOAD COMBINATIONS
C    THERE ARE A MAXIMUM OF 5 INDEPENDANT LOAD CASES
C     AND 5 DEPENDANT LOAD COMBINATIONS
C    ACT  HOLDS THE DISPLACEMENTS FOR THE 120 DOF FOR 10 CASES
C    EMCASE HOLDS THE 6 RESULTANT MEMBER END FORCES
C    SECTFORC HOLDS THE SECTIONAL FORCES AT 21 SECTIONS
C    SUPCASE HOLDS THE SUPPORT REACTIONS FOR THE 3 DOF OF EACT SUPP.
C    COMB IS A 5X5 MATRIX THAT SPECIFIES THE FACTORS
C    NCOMB IS THE NUMBER OF DEPENDANT LOAD COMBINATIONS
C
        SUBROUTINE FACTOR(NT,TALLY,CASES)
C
        COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
        COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
        COMMON /RELEASE/MBREL,SREL,STALLY
        INTEGER TALLY,NT
        REAL COMB(5,5)
        INTEGER NCOMB,ACTLIST(10),ACASES
        REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
        REAL  FEMDIS(5,40,6)
        INTEGER MBREL(40),SREL(40),STALLY,CASES
        REAL SUM(3),SUMX,SUMY,SUMZ,SUM2(21,3),FACT
        INTEGER I,J,K,KK,L,M,N
        DO 1 I=1,CASES
         DO 2 J=1,NT
          L=(J-1)*3+1
          M=L+1
          N=L+2
2        CONTINUE
1       CONTINUE
C    FACTOR THE JOINT DISPLACEMENTS
        DO 10 I=1,NT
         DO 11 K=1,NCOMB
          KK=K+5          ! THIS IS SO THAT THE FIRST COMB IS # 6
          L=(I-1)*3+1     ! THIS IS THE DOF NUMBER FOR THE JOINT
          M= L + 1
          N= L + 2
          SUMX=0
          SUMY=0
          SUMZ=0
          DO 12 J=1,CASES
           FACT=COMB(K,J)
           IF(FACT.EQ.0.0) GOTO 12
           SUMX=SUMX+ACT(J,L)*FACT
           SUMY=SUMY+ACT(J,M)*FACT
           SUMZ=SUMZ+ACT(J,N)*FACT
12        CONTINUE
          ACT(KK,L)=SUMX
          ACT(KK,M)=SUMY
          ACT(KK,N)=SUMZ
11       CONTINUE
10      CONTINUE
C     FACTOR THE MEMBER END ACTIONS IN EMCASE
        DO 19 I=1,6
19       SUM(I)=0
        DO 20 I=1,TALLY
         DO 21 K=1,NCOMB
          KK=K+5
```

```
          DO 22 J=1,CASES
           FACT=COMB(K,J)
           IF(FACT.EQ.0.0) GOTO 22
           DO 23 M=1,6
23          SUM(M)=SUM(M)+EMCASE(J,I,M)*FACT
22        CONTINUE
           DO 24 M=1,6
            EMCASE(KK,I,M)=SUM(M)
            SUM(M)=0.0
24         CONTINUE
21        CONTINUE
20      CONTINUE
C    FACTOR THE SUPPORT REACTIONS
        DO 30 I=1,NT
         IF(SREL(I).EQ.0) GOTO 30
          DO 31 K=1,NCOMB
           KK=K+5
           DO 32 J=1,CASES
            FACT=COMB(K,J)
            IF(FACT.EQ.0.0) GOTO 32
            DO 33 M=1,3
             SUPCASE(KK,I,M)=SUPCASE(KK,I,M)+SUPCASE(J,I,M)*FACT
33          CONTINUE
32         CONTINUE
31        CONTINUE
30       CONTINUE
C    FACTOR THE SECTIONAL FORCES FOR EACH MEMBER
        DO 40 I=1,TALLY
         DO 41 K=1,NCOMB
          KK=K+5
          DO 42 J=1,CASES
           FACT=COMB(K,J)
           IF(FACT.EQ.0.0) GOTO 42
           DO 43 M=1,21
            DO 44 N=1,3
             SUM2(M,N)=SUM2(M,N)+SECTFORC(J,I,N,M)*FACT
44          CONTINUE
43         CONTINUE
42        CONTINUE
          DO 45 M=1,21
           DO 46 N=1,3
            SECTFORC(KK,I,N,M)=SUM2(M,N)
            SUM2(M,N)=0.0
46         CONTINUE
45        CONTINUE
41       CONTINUE
40      CONTINUE
        RETURN
        END
```

```
C       THIS SUBROUTINE IS TO PROCESS THE INDIVIDUAL SHEAR AND
C       MOMENT AND LOAD AND DEFLECTION DIAGRAMS FOR THE MEMBERS
C
        SUBROUTINE INDIV2(NAME,MEMMAX,I4014)
C
        COMMON /GEOM/ MT,TALLY,NLOC,NT
        COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
        COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
        COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
        REAL MCASE(5,40,6),JCASE(5,40,3)
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        INTEGER CASES,NMCASE(5),NJCASE(5)
        REAL MLOAD(40,6),JLOAD(40,3)
        REAL COMB(5,5)
        INTEGER NCOMB,ACTLIST(10),ACASES
        REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
        REAL  FEMDIS(5,40,6)
        REAL ZX,WX,ZY,WY,L,LL,LA,LB,Q,MEMMAX(40,3)
        REAL XP(6),YP(6),MAG,DEG,PLUS,LPLUS,VZY,ZWY,Y1,Y2,Y3,Y4,Y5,Y,X
        REAL MAX,MIN,ABB,MAXLOC,MINLOC,LOCABB,MEMB(21),K0,K1
        LOGICAL LOAD,SHEAR,MOMENT,ENV,OVER,I4014
        INTEGER TYPE,DIR,K,I,J,N,H,S(3,6),TITLE(6),DIAGRAM,IUNITS(3)
        REAL AUX(4),AUY(4),ADX(4),ADY(4)
        CHARACTER*1 RES
        CHARACTER*30 NAME(10)
        DATA ADX/0,-.025,.05,-.025/
        DATA ADY/0,.05,0,-.05/
        DATA AUX/0,-.025,.05,-.025/
        DATA AUY/0,-.05,0,.05/
        LOAD=.FALSE.
        ENV =.FALSE.
        OVER=.FALSE.
C      SHEAR
        S(1,1)=83
        S(1,2)=72
        S(1,3)=69
        S(1,4)=65
        S(1,5)=82
        S(1,6)=32
C     MOMENT
        S(2,1)=77
        S(2,2)=79
        S(2,3)=77
        S(2,4)=69
        S(2,5)=78
        S(2,6)=84
        CALL NEWPAG
        CALL CMCLOS
        CALL CMOPEN
2       PRINT*,'INDIVIDUAL SHEAR AND MOMENT DIAGRAMS'
        PRINT*,'LISTING OF THE LOAD CASES AND COMBINATIONS'
        PRINT*,' '
        PRINT*,'THE CASES:'
        DO 3 J=1,CASES
3        PRINT 201,J,NAME(J)
        PRINT*,'THE COMBINATIONS:'
        IF(NCOMB.LE.0) GOTO 7
        DO 4 J=1,NCOMB
         K=J+5
```

```
4        PRINT*,K,NAME(K)
201     FORMAT(' ',3X,I2,3X,A30)
7       PRINT*,'THE ENVELOPES:'
        PRINT*,'     11  MAXIMUM ENVELOPE'
        PRINT*,'     12  MINUMUM ENVELOPE'
        PRINT*,' '
        PRINT*,' '
5       PRINT*,'>>ENTER LOAD NUMBER'
17      READ*,J
        IF(J.LT.0) RETURN
        IF(J.EQ.11) THEN
         ENV=.TRUE.
         J=11 ! FOR THE SECTFORC
         GOTO 111
        END IF
        IF(J.EQ.12) THEN
         ENV=.TRUE.
         J=12
         GOTO 111
        END IF
        IF(J.GT.NCOMB+5) GOTO 6
        IF(J.GT.CASES.AND.J.LT.6) GOTO 6
        GOTO 111
6       PRINT*,'*** INVALID LOAD CASE OR COMBINATION ***'
        GOTO 5
111     IF(OVER) GOTO 8
        CALL NEWPAG
        CALL CMCLOS
        CALL CMOPEN
        SHEAR=.FALSE.
        MOMENT=.FALSE.
1       CALL TEXT(21,'>>ENTER MEMBER NUMBER')
        CALL CMCLOS
        CALL CMOPEN
        READ*,N
        IF(N.LT.0) GOTO 111
        IF(N.EQ.0) GOTO 2
        IF(N.LE.0.OR.N.GT.TALLY) THEN
         PRINT*,'*** INVALID MEMBER NUMBER ***'
         GOTO 1
        END IF
8       IF(J.EQ.11.OR.J.EQ.12) GOTO 9
100     FORMAT(A1)
        CALL TEXT(14,'>>>PLOT LOADS?')
        CALL CMCLOS
        CALL CMOPEN
        READ 100,RES
        IF(RES.EQ.'Y') LOAD=.TRUE.
9       CALL TEXT(14,'>>>PLOT SHEAR?')
        CALL CMCLOS
        CALL CMOPEN
        READ 100,RES
        IF(RES.EQ.'Y') SHEAR=.TRUE.
        CALL TEXT(15,'>>>PLOT MOMENT?')
        CALL CMCLOS
        CALL CMOPEN
        READ 100,RES
        IF(RES.EQ.'Y') MOMENT=.TRUE.
        IF(OVER) GOTO 18
        CALL NEWPAG
```

```
      CALL CMCLOS
      CALL CMOPEN
      CALL WINDOW(0.,100.,0.,100.)
      CALL VWPORT(5.,105.,0.,100.)
      CALL MOVE(0.,0.)
      CALL DRAW(100.,0.)
      CALL DRAW(100.,100.)
      CALL DRAW(0.,100.)
      CALL DRAW(0.,0.)
      CALL CMCLOS
      CALL CMOPEN
      L=MT(N,5)/12
      LPLUS=L*1.2
      PLUS =L*.1
      ZX   =-PLUS
      WX   =L+PLUS
18    IF(J.GT.5) GOTO 13     !  DO NOT PLOT LOADS
      IF(LOAD) THEN
       CALL WINDOW(-0.1,1.1,-7.5,7.5)
       CALL VWPORT(5.,105.,85.,100.)
       CALL MOVE(0.,0.)
       CALL DRAW(1.,0.)
       DEG=0
       LL=1.0
       DO 10 I=1,NMCASE(J)
        IF(MCASE(J,I,1).EQ.N) THEN
         TYPE=MCASE(J,I,2)
         DIR =MCASE(J,I,3)
         MAG =MCASE(J,I,4)
         LA  =MCASE(J,I,5)
         LB  =MCASE(J,I,6)
         ZY  =-WX
         WY  =-ZX
         Y   =0
         IF(TYPE.EQ.1.OR.TYPE.EQ.3) THEN
          X=LA*L   ! WINDOW IS IN  LPLUS
          CALL DRWARROW(X,Y,DIR,MAG,DEG,ZX,WX,ZY,WY)
         ELSE
          CALL WINDOW(-.1,1.1,-.5,.5)
          CALL VWPORT(5.,105.,85.,100.)
          DO 11 K=1,6
           XP(K)=LA+((LB-LA)*.2*(K-1))
           YP(K)=0
11        CONTINUE
          DO 12 K=1,6
           CALL MOVE(XP(K),YP(K))
           CALL TRANSL(XP(K),YP(K))
           CALL VECREL
           IF(MAG.GT.0) THEN
            CALL POLY(4,AUX,AUY)
           ELSE
            CALL POLY(4,ADX,ADY)
           END IF
           CALL VECABS
           CALL TRANSL(-XP(K),-YP(K))
12        CONTINUE
         END IF
        END IF
10    CONTINUE
      END IF
```

```
C    THIS IS THE SECTION TO SOTRT AND CREATE THE SHEAR
C    AND/OR MOMENT PART
13       IF(SHEAR) THEN
          VZY=45
          VWY=85
          IUNITS(1)=75
          IUNITS(2)=73
          IUNITS(3)=80
          DO 21 I=1,21
           MEMB(I)=SECTFORC(J,N,2,I)
21        CONTINUE
         DO 22 K=1,6
22        TITLE(K)=S(1,K)
          Y1=66
          Y2=62
          Y3=58
          Y4=54
          Y5=50
          DIAGRAM=2
          SHEAR=.FALSE.
         GOTO 60
         END IF
30       IF(MOMENT) THEN
          VZY=3
          VWY=43
          IUNITS(1)=70
          IUNITS(2)=45
          IUNITS(3)=75
          DO 31 I=1,21
           MEMB(I)=SECTFORC(J,N,3,I)     ! ALREADY IN F-K
31        CONTINUE
          DO 32 K=1,6
32         TITLE(K)=S(2,K)
          Y1=26
          Y2=22
          Y3=18
          Y4=14
          Y5=10
          DIAGRAM=3
          MOMENT=.FALSE.
         GOTO 60
         END IF
C     THIS IS THE ROUTINE TO DO THE SHEAR,MOMENT
60       MAX=-100000
         MIN= 100000
         ABB= 0.
         DO 61 I=1,21
          IF(MEMB(I).LT.MIN) THEN
           MIN=MEMB(I)
           MINLOC=(I-1)/20.0
          END IF
          IF(MEMB(I).GT.MAX) THEN
           MAX=MEMB(I)
           MAXLOC=(I-1)/20.0
          END IF
          IF(ABS(MIN).GT.ABS(MAX)) THEN
            IF(ABS(MIN).GT.ABS(ABB)) THEN
             LOCABB=(I-1)/20.0
             ABB=MIN
            END IF
```

```
         ELSE
         IF(ABS(MAX).GT.ABS(ABB)) THEN
          ABB=MAX
          LOCABB=(I-1)/20.0
         END IF
        END IF
61      CONTINUE
        ZY=-1*MEMMAX(N,DIAGRAM)
        WY=-ZY
        CALL WINDOW(-.1,1.1,-1.0,1.0)
        CALL VWPORT(5.,105.,VZY,VWY)
C     THIS WILL DRAW THE GRID LINES
        IF(OVER) GOTO 650
        CALL MOVE(0.,0.)
        CALL DRAW(1.0,0.)
        CALL MOVE(0.,-1.)
        CALL DRAW(0.,1.)
        DO 65 Q=0,1,0.1
         CALL MOVE(Q,-0.01)
         CALL DRAW(Q, 0.01)
         CALL MOVE(-0.01,Q)
         CALL DRAW( 0.0 ,Q)
         CALL MOVE(-0.01,-Q)
         CALL DRAW( 0.0 ,-Q)
65      CONTINUE
650     CALL WINDOW(-0.1,1.1,ZY,WY)
        CALL VWPORT(5.,105.,VZY,VWY)
        IF(OVER) GOTO 620
         YSCAL=WY/10.0
         CALL MOVE(-0.05,0.0)
         CALL DRAW(-0.05,YSCAL)
         CALL MOVE(-0.05,-YSCAL)
         CALL TXICUR(2)
         CALL TXADE
         CALL TXSIZE(4,0,0)
         CALL RNUMBR(YSCAL,3,8)
         CALL MOVE(-0.05,-2*YSCAL)
         CALL TEXT(3,IUNITS)
620     DO 62 I=2,21
         K0=(I-2)*.05
         K1=(I-1)*.05
         H=I-1
         CALL MOVE(K0,MEMB(H))
         CALL DRAW(K1,MEMB(I))
62      CONTINUE
         CALL MOVE(1.0,MEMB(21))
         CALL DRAW(1.,0.)
C     LABLES FOR THE BOX
        CALL WINDOW(103.,131.2,0.,100.)
        CALL VWPORT(103.,131.2,0.,100.)
        CALL MOVE(110.,95.)
        CALL TXICUR(1)
        CALL TXFCUR(2)
        CALL TXSIZE(2,0,0)
        CALL TXAM
        CALL TEXT(7,'MEMBER ')
        CALL INUMBR(N,3)
        CALL WINDOW(103.,131.2,VZY,VWY)
        CALL VWPORT(103.,131.2,VZY,VWY)
        CALL TXSIZE(3,0,0)
```

```
      CALL TXICUR(2)
      CALL TXFCUR(1)
      IF(J.EQ.12) CALL TXICUR(8)
      CALL TXADE
      CALL MOVE(109.,Y1)
      CALL TEXT(6,TITLE)
      CALL TXAM
      CALL MOVE(117.,Y1)
      CALL TEXT(3,'MAG')
      IF(I4014) THEN
      CALL MOVE(125.,Y1)
      CALL TEXT(3,'LOC')
      END IF
      CALL MOVE(109.,Y2)
      CALL TEXT(7,'ABS MAX')
      CALL MOVE(109.,Y3)
      CALL TEXT(7,'MAXIMUM')
      CALL MOVE(109.,Y4)
      CALL TEXT(7,'MINIMUM')
      CALL MOVE(112.,Y5)
      CALL TEXT(9,'LOAD CASE')
      CALL MOVE(120.,Y5)
      CALL INUMBR(J,3)
      CALL MOVE(120.,Y2)
      CALL RNUMBR(ABB,2,9)
      CALL MOVE(120.,Y3)
      CALL RNUMBR(MAX,2,9)
      CALL MOVE(120.,Y4)
      CALL RNUMBR(MIN,2,9)
      IF(I4014) THEN
      CALL MOVE(128.,Y2)
      CALL RNUMBR(LOCABB,1,3)
      CALL MOVE(128.,Y3)
      CALL RNUMBR(MAXLOC,1,3)
      CALL MOVE(128.,Y4)
      CALL RNUMBR(MINLOC,1,3)
      END IF
      CALL CMCLOS
      CALL CMOPEN
      IF(MOMENT)GOTO 30
      CALL WINDOW(0.,100.,0.,100.)
      CALL VWPORT(5.,105.,0.,100.)
      CALL TXICUR(1)
      CALL HOME
63    CALL TEXT(27,'>>ANOTHER MEMBER, OVERWRITE')
      CALL TEXT(31,'OR ANOTHER LOAD CASE (M/O/L/NO)')
      CALL CMCLOS
      CALL CMOPEN
      READ 100,RES
      OVER=.FALSE.
      IF(RES.EQ. 'M') GOTO 111
      IF(RES.EQ.'L') GOTO 2
      IF(RES.EQ.'O') THEN
       OVER=.TRUE.
       CALL TEXT(20,'>>ENTER LOAD NUMBER ')
       CALL CMCLOS
       CALL CMOPEN
       GOTO 17
      END IF
      IF(RES.NE.'N') GOTO 63
```

```
C     THIS IS THE SUB TO CALCULATE THE SHEAR AND
C     MOMENTS ALONG THE MEMBER DUE TO A UNIFORM LOAD
C     OVER PART OF THE MEMBER
C
        SUBROUTINE MPYMOMSHE(MAG,L,A,B,TMEMMOM,TMEMSHE,CASE)
C
        REAL MAG,A,B,C,D,E,L,X,J,RL1,RL2,RL,RR1,RR2,RR,ML1,ML2,ML,MR1,MR2,MR
        REAL TMEMMOM(21),TMEMSHE(21),BEG,INC,EN,RI
        INTEGER CASE,I
        I=0
        MAG=-MAG    !  IN K/F
        BEG=0
        INC=0.05
        EN =1.0
        A=L*A
        B=L*B
        C=B-A
        D=L-A
        E=L-B
        GOTO (10,20,30,40),CASE
10      RL1=MAG*B/2*(2*(1-(B/L)**2)+(B/L)**3)
        RL2=MAG*A/2*(2*(1-(A/L)**2)+(A/L)**3)
        RL =RL1-RL2
        RR =-MAG*C+RL
        ML1=-MAG*B**2/12*(1+2*E/L+3*(E/L)**2)
        ML2=-MAG*A**2/12*(1+2*D/L+3*(D/L)**2)
        ML =ML1-ML2
        MR1=-MAG*B**2/12*(1+3*E/L)
        MR2=-MAG*A**2/12*(1+3*D/L)
        MR =MR1-MR2
        I=0
        DO 11 J=BEG,EN,INC
         I=I+1
         X=J*L
         IF(J.LT.A/L) THEN
          TMEMSHE(I)=RL
          TMEMMOM(I)=ML+RL*X
         ELSE IF(J.LT.B/L) THEN
          TMEMSHE(I)=RL-MAG*(X-A)
          TMEMMOM(I)=ML-MAG*(X-A)**2
         ELSE
          TMEMSHE(I)=RR
          TMEMMOM(I)=MR+MAG*(L-X)
        END IF
11      CONTINUE
        RETURN
20      RL=MAG*C/(2*L)*(2*E-C)
        RR=-MAG*C+RL
        I=0
        DO 21 J=BEG,EN,INC
         I=I+1
         X=J*L
         IF(J.LT.A/L) THEN
          TMEMSHE(I)=RL
          TMEMMOM(I)=RL*X
         ELSE IF(J.LT.B/L) THEN
          TMEMSHE(I)=RL-MAG*(X-A)
          TMEMMOM(I)=RL*X-MAG*(X-A)**2/2
         ELSE
          TMEMSHE(I)=RR
```

```
          TMEMMOM(I)=-RR*(L-X)
         END IF
21      CONTINUE
        RETURN
30      RL1=MAG*B/8*(8-6*B/L+(B/L)**3)
        RL2=MAG*A/8*(8-6*A/L+(A/L)**3)
        RL=RL1-RL2
        RR=-MAG*C+RL
        ML=0
        MR1=-MAG*B**2/8*(2-(B/L)**2)
        MR2=-MAG*A**2/8*(2-(A/L)**2)
        MR =MR1-MR2
        I=0
        DO 31 J=BEG,EN,INC
         I=I+1
         X=J*L
          IF(J.LT.A/L) THEN
           TMEMSHE(I)=RL
           TMEMMOM(I)=RL*X
          ELSE IF(J.LT.B/L) THEN
           TMEMSHE(I)=RL-MAG*(X-A)
           TMEMMOM(I)=RL*X-(MAG*(X-A)**2/2)
          ELSE
           TMEMSHE(I)=RR
           TMEMMOM(I)=MR-RR*(L-X)
          END IF
31      CONTINUE
        RETURN
40      RL1=MAG*B*((L-B/2)/L)/2*(2+B/L*((L-B/2)/L))
        RL2=MAG*A*((L/A/2)/L)/2*(2+A/L*((L-A/2)/L))
        RL =RL1-RL2
        RR =- MAG*C+RL
        ML1=-MAG*B**2*((L-B/2)/L)**2/2
        ML2=-MAG*A**2*((L-A/2)/L)**2/2
        ML =ML1-ML2
        MR=0
        I=0
        DO 41 J=BEG,EN,INC
         I=I+1
         X=J*L
         IF(J.LT.A/L) THEN
          TMEMSHE(I)=RL
          TMEMMOM(I)=ML+RL*X
         ELSE IF(J.LT.B/L) THEN
          TMEMSHE(I)=RL-MAG*(X-A)
          TMEMMOM(I)=ML+MAG*(X-A)**2/2
         ELSE
          TMEMSHE(I)=RR
          TMEMMOM(I)=-RR*(L-X)
         END IF
41      CONTINUE
        RETURN
        END
```

```
C        THIS IS A SUBROUTINE TO CALCULATE THE SHEAR AND MOMENT
C        AT SPECIFIED SECTIONS ALONG THE MEMBER
C        *** THIS WILL GIVE THE VALUE OF THE SHEAR AND MOMENT DIAGRAM
C
         SUBROUTINE MYMOMSHE(CASE,MAG,A,B,L,TMEMMOM,TMEMSHE)
C
         REAL MAG,A,B,L,TMEMMOM(21),TMEMSHE(21),J,X
         REAL BEG,EN,INC,RI
         INTEGER I,CASE
         I=0
         MAG=-MAG   !  IN K/F
         BEG=0
         EN =1.025
         INC=.05
         GOTO (10,20,30,40),CASE
10       DO 11 J=BEG,EN,INC
          I=I+1
          X=J*L
          TMEMMOM(I)=(MAG*L**2)/2*(J-J**2-.166666667)
11       TMEMSHE(I)=MAG*L/2*(1-2*(X/L))
         RETURN
20       DO 21 J=BEG,EN,INC
          X=J*L
          I=I+1
         TMEMMOM(I)=MAG*L**2/2*((X/L)-(X/L)**2)
21       TMEMSHE(I)=MAG*L/2*(1-2*(X/L))
         RETURN
30       I=22
         DO 31 J=BEG,EN,INC
          I=I-1
         X=L*J
         TMEMMOM(I)=-MAG*L**2/8*(-5*(X/L)+4*(X/L)**2+1)
31       TMEMSHE(I)=-(MAG*L)*(.625-J)
         RETURN
40       DO 41 J=BEG,EN,INC
          I=I+1
         X=L*J
         TMEMMOM(I)=-MAG*L**2/8*(-5*(X/L)+4*(X/L)**2+1)
41       TMEMSHE(I)=MAG*L*(.625-J)
         RETURN
         END
```

```
C         THIS IS A SUBROUTINE TO CALCULATE THE SHEAR AND MOMENTS
C         ALONG A MEMBER WITH A SINGLE CONCENTRATED LOAD
C         THE RESULTS GIVEN ARE THE VALUES OF THE S AND M DIAGRAM
C
          SUBROUTINE PYMOMSHE(CASE,MAG,A,L,TMEMMOM,TMEMSHE)
C
          REAL MAG,L,TMEMMOM(21),TMEMSHE(21),R1,R2,J,A,B,X,ML
          INTEGER I,K,CASE
          REAL BEG,EN,INC
          B=1-A
          MAG=-MAG
          BEG=0.0
          EN =1.025
          INC=0.05
          I=0
          GOTO (10,20,30,40),CASE
10        R1=MAG*B**2*(1+2*A)
          R2=-MAG+R1
          ML=-MAG*L*A*B**2
           DO 11 J=BEG,EN,INC
           I=I+1
           X=L*J
           IF(J.LE.A)THEN
            TMEMSHE(I)=R1
            TMEMMOM(I)=R1*X+ML
           ELSE
            TMEMSHE(I)=R2
            TMEMMOM(I)=R1*X+ML-(MAG*(J-A)*L)
           END IF
11        CONTINUE
          RETURN
20        R1=MAG*B
          R2=-MAG+R1
          DO 21 J=BEG,EN,INC
           I=I+1
           X=L*J
           IF(J.LT.A)THEN
            TMEMSHE(I)=R1
            TMEMMOM(I)=R1*X
           ELSE
            TMEMSHE(I)=R2
            TMEMMOM(I)=R1*X-(MAG*(J-A)*L)
           END IF
21        CONTINUE
          RETURN
30        R1=MAG*B**2*((2+A)/2)
          R2=-MAG+R1
          DO 31 J=BEG,EN,INC
           I=I+1
           X=L*J
           IF(J.LT.A) THEN
            TMEMSHE(I)=R1
            TMEMMOM(I)=R1*X
           ELSE
            TMEMSHE(I)=R2
            TMEMMOM(I)=R1*X-MAG*(J-A)*L
           END IF
31        CONTINUE
          RETURN
40        R1=MAG*B*((3-B**2)/2)
```

```
      R2=-MAG+R1
      ML=-MAG*L*B*((1-B**2)/2)
      I=0
      DO 41 J=BEG,EN,INC
       I=I+1
       X=L*J
       IF(J.LE.A)THEN
         TMEMSHE(I)=R1
         TMEMMOM(I)=R1*X+ML
        ELSE
         TMEMSHE(I)=R2
         TMEMMOM(I)=ML+R1*X-(MAG*(J-A)*L)
        END IF
41    CONTINUE
      RETURN
      END
```

****

```
C       SUBROUTINE TO DRAW A SERIES OF BAYS FROM ONE BAY
C       NOTE: THE BAY TO BECOPIED MUST BE A SIMPLE PORTAL
C             BAY WITH ALL 90 DEG CORNERS
        SUBROUTINE BAYS
        COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        REAL ZX,WX,ZY,WY,ROUND
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        REAL LOCX(4),LOCY(4),MATCH(2,2)
        REAL BDIST,BHIEG,XN,YN
        INTEGER N,I,J,K,DIR,NOD1,NOD2,SAMECOL,SAMEBEAM
        LOGICAL NBEAM,NCOL
        INTEGER NOD(4),STORT
        CHARACTER*1 STR
        CALL WINDOW(ZX,WX,ZY,WY)
        CALL VWPORT(5.,105.,0.,100.)
        NBEAM=.TRUE.
        NCOL =.TRUE.
1       PRINT*,'CREATE HOW MANY ADDITIONAL BAYS?'
        READ*,N
        IF(N.LE.0)GOTO 1
        PRINT*,'TO THE RIGHT OR LEFT? R/L'
        READ 100,STR
100     FORMAT(A1)
        IF(STR.EQ.'L'.OR.STR.EQ.'l') then
         DIR=1
         ELSE
         DIR=2
        END IF
        PRINT*,'WOULD YOU LIKE THE COLUMN PROPERTIES AUTOMATICALLY COPIED ?'
        READ 100,STR
        IF(STR.EQ.'Y'.OR.STR.EQ.'y')THEN
          NCOL=.FALSE.
        END IF
        PRINT*,'WOULD YOU LIKE THE BEAM PROPERTIES AUTOMATICALLY COPIED?'
        READ 100,STR
        IF(STR.EQ.'Y'.OR.STR.EQ.'y') then
         NBEAM=.FALSE.
        END IF
        PRINT*,'LOCATE THE 4 CORNERS OF THE BAY:'
        PRINT*,'STARTING AT THE LOWER LEFT CORNER AND GO CLOCK-WISE'
        DO 10 I=1,4
11       CALL LOCATE(1,XN,YN,IGOT,IDAT)
         IF(XN.GT.WX) GOTO 80     !QUIT THIS SECTION
         CALL SAMENODES(XN,YN,NODE,NT,NLOC,ROUND)
         IF(NODE.EQ.0) THEN
          PRINT*,'SORRY--NODE ',I,' WAS NOT MATCHED--TRY AGAIN'
          GOTO 11
         END IF
         LOCX(I)=XN
         LOCY(I)=YN
         NOD(I) =NODE
10      CONTINUE
        BDIST=ABS(LOCX(1)-LOCX(4))
        IF(DIR.EQ.1) THEN
         BHEIG=ABS(LOCY(2)-LOCY(1))
         BDIST=-BDIST
         MATCH(1,1)=LOCX(1)
```

```
       MATCH(1,2)=LOCY(1)
       MATCH(2,1)=LOCX(2)
       MATCH(2,2)=LOCY(2)
       NOD1=NOD(1)
       NOD2=NOD(2)
      ELSE
       BHEIG=ABS(LOCY(3)-LOCY(4))
       MATCH(1,1)=LOCX(4)
       MATCH(1,2)=LOCY(4)
       MATCH(2,1)=LOCX(3)
       MATCH(2,2)=LOCY(3)
       NOD1=NOD(4)
       NOD2=NOD(3)
      END IF
      STORT=TALLY
      BSAVE=BDIST
C     ADD THE ADDITIONAL BAYS

      DO 20 I=1,N
      NLOC(NT+1,1)=MATCH(1,1)+BDIST
      NLOC(NT+1,2)=MATCH(1,2)
      NLOC(NT+2,1)=NLOC(NT+1,1)
      NLOC(NT+2,2)=MATCH(2,2)
C
      MT(TALLY+1,1)=NT+1
      MT(TALLY+1,2)=NT+2
      IF(DIR.EQ.1) THEN
       MT(TALLY+2,1)=NT+2
       MT(TALLY+2,2)=NOD2
      ELSE
       MT(TALLY+2,1)=NOD2
       MT(TALLY+2,2)=NT+2
      END IF
      MT(TALLY+1,5)=(SQRT((NLOC(MT(TALLY+1,1),1)-NLOC(MT(TALLY+1,2),1))**2
     *  +(NLOC(MT(TALLY+1,1),2)-NLOC(MT(TALLY+1,2),2))**2))*12
      MT(TALLY+2,5)=(SQRT((NLOC(MT(TALLY+2,1),1)-NLOC(MT(TALLY+2,2),1))**2
     *  +(NLOC(MT(TALLY+2,1),2)-NLOC(MT(TALLY+2,2),2))**2))*12
      nod2=nt+2
      NT=NT+2
      TALLY=TALLY+2
      BDIST=BDIST+BSAVE
20    CONTINUE
C   DRAW THE NEW MEMBERS AND NUMBER THE NEW NODES
C              !AND NUMBER THE NODES AND MEMBERS
C     COPY THE MEMBER PROPERTIES TO THE NEW MEMBERS
C     FIND THE MEMBER NUMBER OF THE OLD COLUMN
      IF(NCOL)GOTO 66 !DO NOT COPY COLUMN PROPERTIES
      DO 60 I=1,TALLY
       IF(INT(MT(I,1)).EQ.NOD1.AND.INT(MT(I,2)).EQ.NOD2) THEN
        SAMECOL=I
        GOTO 70       !MEMBER FOUND
       END IF
60    CONTINUE
      PRINT*,'COLUMN NOT FOUND--NO PROPERTIES COPIED'
      GOTO 66
70    J=TALLY+1-(2*N)
      DO 68 I=J,TALLY,2
       DO 67 K=6,12
        MT(I,K)=MT(SAMECOL,K)
67     CONTINUE
```

```
66      CONTINUE
66      IF(NBEAM)GOTO 80          !DO NOT COPY PROPERTIES
        DO 71 I=1,TALLY
          IF(INT(MT(I,1)).EQ.NOD(2).AND.INT(MT(I,2)).EQ.NOD(3))  THEN
           SAMEBEAM=I
           GOTO 79
          END IF
71      CONTINUE
        PRINT*,'BEAM NOT FOUND--NO PROPERTIES COPIED'
        GOTO 80
79      J=TALLY+1-(2*N)+1
        DO 78 I=J,TALLY,2
          DO 77 K=6,12
           MT(I,K)=MT(SAVEBEAM,K)
77        CONTINUE
78      CONTINUE
80      I=I
        RETURN
        END
```

```
C     THIS IS THE MAIN SUBROUTINE TO BUILT THE STRUCTURE
C
        SUBROUTINE BUIL
C
        COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        REAL ZX,WX,ZY,WY,ROUND
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        REAL X,Y,LONGEST
        INTEGER NODECOUNT
        REAL TX,TY,PTX,PTY,RTX,RTY,Q,R,MIDX,MIDY
        LOGICAL PREDIFF,PREDIFS,NOINST
        INTEGER*4 SYS$ASSIGN,SYS$QIOW,IFUNC
        CHARACTER*1 ICHAR,I2
        INTEGER N1,N2
        INTEGER RESPONSE,ENDS,ENDF,RES
        REAL XN1,XN2,YN1,YN2,LENGTH,ANGLE,SUMX,SUMY
        IFUNC=113
        IRET=SYS$ASSIGN(%DESCR('TT'),ICH,,)
        CALL NEWPAG
        CALL CMCLOS
        CALL CMOPEN
        PRINT*,'>>BUILD SECTION'
        PRINT*,'  DO YOU NEED INSTRUCTIONS? Y/N'
        READ 100,I2
        IF(I2.NE.'Y') THEN
         NOINST=.TRUE.
        ELSE
         NOINST=.FALSE.
        END IF
        PRINT*,'**************************************************'
        PRINT*,'READY TO BEGIN:'
        PRINT*,'ALL LENGTHS WILL BE ROUNDED'
        PRINT*,' TO THE NEAREST INCRIMENT THAT YOU SPECIFY'
        PRINT*,'INPUT THE ROUNDING INCRIMENT IN FEET'
        READ*,ROUND
        PRINT*,'**************************************************'
        PRINT*,'NODES WILL BE NUMBERED IN THE ORDER CREATED'
        PRINT*,'MEMBERS ON EACH NODE WILL BE NUMBERED IN '
        PRINT*,' THE ORDER CREATED'
C       PRINT*,'LATER YOU WILL BE ABLE TO RENUMBER THE NODES'
        PRINT*,'**************************************************'
        PRINT*,'INPUT THE LARGEST OVERALL DIMENSION'
        READ*,LONGEST
        CALL NEWPAG
        CALL CMCLOS
        CALL CMOPEN
        LONGEST=LONGEST*1.5
        IF(NOINST) GOTO 6
        PRINT*,'**************************************************'
        PRINT*,'  THIS SECTION WILL ASSIST IN CREATING A '
        PRINT*,'  2-D FRAME IN AN INTERACTIVE GRAPHIC MODE'
        PRINT*,'  THE STRUCTURE CAN BE CREATED, IN PIECES,'
        PRINT*,'  IN A COMBINATION OF METHODS'
        PRINT*,'**************************************************'
        PRINT*,'  WHEN SPECIFYING THE FIRST END F A MEMBER'
        PRINT*,'  YOU CAN LOCATE IT BY:'
        PRINT*,'   1. X,Y COORDINATE'
        PRINT*,'   2. NODE NUMBER (THAT ALREADY HAS AN'
```

```
      PRINT*,'        X,Y COORDINATE ASSOCIATED WITH IT)'
      PRINT*,'    3.POINT TO IT WITH A LOCATE COMMAND'
      PRINT*,'TO LOCATE THE ENDING POINT OF THE MEMBER:'
      PRINT*,'    1. X,Y COORDINATE'
      PRINT*,'    2. NODE NUMBER'
      PRINT*,'    3. POINT TO IT WITH A LOCATE COMMAND'
      PRINT*,'    4. SPECIFY AN ANGLE AND A LENGTH'
      PRINT*,'    5. MOVE TO IT IN INCREMENTED STEPS'
      PRINT*,' IT WILL AUTOMATICALLY CALCULATE THE LENGTH'
      PRINT*,'**********************************************'
6     PRINT*,'NOTE: DO YOU ALWAYS PLAN TO ENTER END'
      PRINT*,' ONE OF THE MEMBER BY THE SAME FORMAT?'
      READ 100,I2
100   FORMAT(A1)
       IF(I2.EQ.'N') THEN
        PREDIFF=.FALSE.
        GOTO 5
       END IF
      PRINT*,' INPUT THE NUMBER OF THAT METHOD NOW:'
      PRINT*,'FOR FIRST END:'
      PRINT*,' 1=X,Y    2=NODE #    3=LOCATE IT'
      READ*,ENDF
      PREDIFF=.TRUE.
5     CALL NEWPAG
      CALL CMCLOS
      CALL CMOPEN
      IF(NOINST) GOTO 8
      PRINT*,'METHODS TO IDENTIFY END 2'
      PRINT*,' 1=X,Y    2=NODE #    3=LOCATE IT'
      PRINT*,' 4=ANGLE AND LENGTH 5=STEP TO IT'
      PRINT*,'FOR METHOD 4, HORIZONTL TO THE RIGHT'
      PRINT*,' IS 0.0 DEGREES STRAIGHT UP IS +90.0'
      PRINT*,' DEGREES, NEGITIVE ANGLES ACCEPTED'
      PRINT*,'FOR METHOD 5, USE THE KEYBOARD:'
      PRINT*,' U=UP'
      PRINT*,' D=DOWN'
      PRINT*,' R=RIGHT'
      PRINT*,' L=LEFT'
      PRINT*,'SHIFT AND THE LETTER IS 5 TIMES THE AMOUNT'
      PRINT*,'E=ENTER THIS POINT AS THE MEMBER END'
8     PRINT*,'**********************************************'
      PRINT*,' '
      PRINT*,' DO YOU PLAN TO ENTER THE SECOND END BY'
      PRINT*,' THE SAME METHOD??'
      READ 100,I2
       IF(I2.EQ.'N') THEN
        PREDIFS=.FALSE.
       ELSE
         PREDIFS=.TRUE.
      PRINT*,'INPUT THE METHOD FOR THE SECOND END'
       READ*,ENDS
      END IF
      IF(TALLY.GT.0) THEN
       PRINT*,'>>> READY TO BEGIN: PRESS <RETURN> TO CONTINUE'
       READ 123, I2
       CALL WINDOW(ZX,WX,ZY,WY)
       CALL VWPORT(5.,105.,0.,100.)
       CALL REDR
123    FORMAT(A)
       GOTO 7
```

```
      END IF
      PRINT*,'LOCATE THE ORIGIN OF THE GLOBAL AXES'
      CALL WINDOW(0.,LONGEST,0.,LONGEST)
      CALL VWPORT(5.,105.,0.,100.)
      CALL LOCATE(1,X,Y,IDAT,IGOT)
      ZX=-X
      WX=ZX+LONGEST
      ZY=-Y
      WY=ZY+LONGEST
      CALL WINDOW(ZX,WX,ZY,WY)
      CALL VWPORT(5.,105.,0.,100.)
      CALL PAGE(ZX,WX,ZY,WY,ROUND)
7     CALL CMCLOS
      CALL CMOPEN
      CALL HOME
      CALL WHERE(TX,TY)
      CALL TXSIZE(2,0.,0.)
C   BEGIN THE DRAWING
10    TALLY=TALLY
      CALL TXICUR(1)
      CALL TXFCUR(1)
111   CALL MOVE(TX,TY)
      CALL TEXT(5,'*END1')
      IF(PREDIFF)GOTO 11
      CALL TEXT(11,'END1 1,2,3?')
      CALL WHERE(TX,TY)
      CALL CMCLOS
      CALL CMOPEN
      IRET=SYS$QIOW(,%VAL(ICH),%VAL(IFUNC),,,,%REF(ICHAR),%VAL(1),,,,)
      ENDF=INDEX('1230',ICHAR)
      IF(ENDF.EQ.0) GOTO 111
11    GOTO(15,16,17,291),ENDF
C    (X,Y)SPECIFY
15    CALL TEXT(9,'**XF,YF??')
      CALL WHERE(TX,TY)
      CALL CMCLOS
      CALL CMOPEN
      READ*,XN1,YN1
      CALL SAMENODES(XN1,YN1,N1,NT,NLOC,ROUND)
      IF(XN1.GT.WX.OR.XN1.LT.ZX)GOTO 111
      IF(YN1.GT.WY.OR.YN1.LT.ZY)GOTO 111
C
      IF(N1.EQ.0) THEN
        NT=NT+1
        N1=NT
        NLOC(NT,1)=XN1
        NLOC(NT,2)=YN1
      END IF
      GOTO 18
C    NODE #
16    CALL MOVE(TX,TY)
      CALL TEXT(10,'**NODEF #?')
      CALL WHERE(TX,TY)
      CALL CMCLOS
      CALL CMOPEN
      READ*,N1
      IF(NT.EQ.0)GOTO 111
      IF(N1.LE.0.OR.N1.GT.NT) GOTO 15
      XN1=NLOC(N1,1)
      YN1=NLOC(N1,2)
```

```
        GOTO 18
C      LOCATE IT
17      CALL TEXT(9,'**LOCATE1')
        CALL WHERE(TX,TY)
        CALL LOCATE(1,XN1,YN1,IDAT,IGOT)
        IF(XN1.GT.WX.OR.XN1.LT.ZX)GOTO 111
        CALL SAMENODES(XN1,YN1,N1,NT,NLOC,ROUND)
        IF(N1.EQ.0) THEN
         NT=NT+1
         N1=NT
         NLOC(NT,1)=XN1
          NLOC(NT,2)=YN1
        END IF
        GOTO 18
C      DRAW BOX AND NUMBER NODE
18      CALL MOVE(XN1,YN1)
        CALL TXSIZE(3,0,0)
        CALL TXICUR(2)
        CALL TXFCUR(3)
        CALL INUMBR(N1,3)
        CALL TXSIZE(2,0,0)
C     SECOND END
20      CALL TXICUR(1)
        CALL TXFCUR(1)
        CALL MOVE(TX,TY)
        CALL TEXT(5,'*END2')
        IF(PREDIFS)GOTO 21
        CALL TEXT(16,'CHOOSE 1,2,3,4,5')
        CALL WHERE(TX,TY)
        CALL CMCLOS
        CALL CMOPEN
        IRET=SYS$QIOW(,%VAL(ICH),%VAL(IFUNC),,,,%REF(ICHAR),%VAL(1),,,,)
        ENDS=INDEX('12345',ICHAR)
        IF(ENDS.EQ.0) GOTO 20
21      GOTO (23,24,25,26,27),ENDS
C      (X,Y) SPECIFY
23      CALL TEXT(9,'**XS,YS??')
        CALL WHERE(TX,TY)
        CALL CMCLOS
        CALL CMOPEN
        READ*,XN2,YN2
        CALL SAMENODES(XN2,YN2,N2,NT,NLOC,ROUND)
        IF(XN2.GT.WX.OR.XN2.LT.ZX)GOTO 20
        IF(YN2.GT.WY.OR.YN2.LT.ZY)GOTO 20
        IF(N2.EQ.0) THEN
         NT=NT+1
         N2=NT
         NLOC(NT,1)=XN2
         NLOC(NT,2)=YN2
        END IF
          GOTO 28
C     NODE #
24      CALL MOVE(TX,TY)
        CALL TEXT(10,'**NODES #?')
        CALL WHERE(TX,TY)
        CALL CMCLOS
        CALL CMOPEN
        READ*,N2
        IF(NT.EQ.1)GOTO 20
        IF(N2.LE.0.OR.N2.GT.NT) GOTO 20
```

```
      XN2=NLOC(N2,1)
      YN2=NLOC(N2,2)
      GOTO 28
C    LOCATE IT
25    CALL TEXT(9,'**LOCATE2')
      CALL WHERE(TX,TY)
      CALL LOCATE(1,XN2,YN2,IGOT,IDAT)
      CALL SAMENODES(XN2,YN2,N2,NT,NLOC,ROUND)
      IF(XN2.GT.WX.OR.XN2.LT.ZX) GOTO 20
      IF(N2.EQ.0) THEN
       NT=NT+1
       N2=NT
       NLOC(NT,1)=XN2
       NLOC(NT,2)=YN2
      END IF
      GOTO 28
C    ANGLE AND LENGTH
26    CALL MOVE(TX,TY)
      CALL TEXT(19,'**ANGLE , LENGTH ??')
      CALL CHCLOS
      CALL CHOPEN
      READ*,ANGLE,LENGTH
      ANGLE=ANGLE/57.2957795131    !TO GET DEGREES TO RADIANS
      XN2=LENGTH*(COS(ANGLE))
      YN2=LENGTH*(SIN(ANGLE))
      XN2=XN1+XN2
      YN2=YN1+YN2
C      CHECK TO SEE X,Y IS IN BOUNDS
      IF(XN2.GT.WX.OR.XN2.LT.ZX)GOTO 20
      IF(YN2.GT.WY.OR.YN2.LT.ZY)GOTO 20
      CALL SAMENODES(XN2,YN2,N2,NT,NLOC,ROUND)
      IF(N2.EQ.0) THEN
       NT=NT+1
       N2=NT
       NLOC(NT,1)=XN2
       NLOC(NT,2)=YN2
      END IF
      GOTO 28
C   STEP TO IT
27    CALL MOVE(TX,TY)
      CALL TEXT(11,'**ENDS STEP')
      CALL WHERE(TX,TY)
      CALL CHCLOS
      CALL CHOPEN
      CALL TRANSL(XN1,YN1)
      SUMX=0
      SUMY=0
211   IRET=SYS$QIOW(,%VAL(ICH),%VAL(IFUNC),,,,%REF(ICHAR),%VAL(1),,,,)
      IF(ICHAR.EQ.'R')THEN
       SUMX=SUMX+ROUND
      ELSE IF(ICHAR.EQ.'r')THEN
       SUMX=SUMX+ROUND*5
      ELSE IF(ICHAR.EQ.'L')THEN
       SUMX=SUMX-ROUND
      ELSE IF(ICHAR.EQ.'l')THEN
        SUMX=SUMX-ROUND*5
      ELSE IF(ICHAR.EQ.'U')THEN
        SUMY=SUMY+ROUND
      ELSE IF(ICHAR.EQ.'u')THEN
        SUMY=SUMY+ROUND*5
```

```
      ELSE IF(ICHAR.EQ.'D')THEN
        SUMY=SUMY-ROUND
      ELSE IF(ICHAR.EQ.'d')THEN
        SUMY=SUMY-ROUND*5
      ELSE IF(ICHAR.EQ.'E'.OR.ICHAR.EQ.'e')THEN
        GOTO 22
      ELSE
       GOTO 211
      END IF
      CALL MOVE(SUMX,SUMY)
      CALL DRAW(SUMX,SUMY)
      CALL CMCLOS
      CALL CMOPEN
      GOTO 211
22    XN2=XN1+SUMX
      YN2=YN1+SUMY
      CALL SAMENODES(XN2,YN2,N2,NT,NLOC,ROUND)
      IF(N2.EQ.0) THEN
       NT=NT+1
       N2=NT
       NLOC(NT,1)=XN2
       NLOC(NT,2)=YN2
      END IF
      CALL TRANSL(-XN1,-YN1)
28    IF(N1.EQ.N2) THEN
       CALL BELL
       CALL CMCLOS
       CALL CMOPEN
       PRINT*,'*** CANNOT SPECIFY SAME NODE AS START AND END ***'
       GOTO 20
      END IF
      TALLY=TALLY+1
      CALL MOVE(XN2,YN2)
      CALL TXSIZE(3,0,0)
      CALL TXICUR(2)
      CALL TXFCUR(3)
      CALL INUMBR(N2,3)
      CALL TXSIZE(2,0,0)
C     DRAW MEMBER
      CALL MOVE(XN1,YN1)
      CALL DRAW(XN2,YN2)
C   CALCULATE AND NUMBER MID-POINT
      MIDX=XN1+(XN2-XN1)/4
      MIDY=YN1+(YN2-YN1)/4
      CALL MOVE(MIDX,MIDY)
      CALL INUMBR(TALLY,3)

C   FIND THE I AND J END OF THE MEMBER JUST CREATED
      MT(TALLY,1)=N1
      MT(TALLY,2)=N2
      IF(XN1.LT.XN2) GOTO 290
      IF(XN1.EQ.XN2) THEN
       IF(YN1.LT.YN2) GOTO 290
       END IF
       MT(TALLY,1)=N2
       MT(TALLY,2)=N1
C   FIND THE LENGTH OF THIS MEMBER
290   MT(TALLY,5)=(SQRT((XN2-XN1)**2+(YN2-YN1)**2))*12
C   FIND OUT IF ANOTHER MEMBER IS DESIRED
      CALL TXICUR(1)
```

```
      CALL TXFCUR(1)
      CALL MOVE(TX,TY)
      CALL TEXT(15,'ANOTHER MEMBER?')
      CALL WHERE(TX,TY)
      CALL CMCLOS
      CALL CMOPEN
      IRET=SYS$QIOW(,%VAL(ICH),%VAL(IFUNC),,,,%REF(ICHAR),%VAL(1),,,,)
      IF(ICHAR.NE.'N') goto 10
C   IF REACHES HERE  RETURN TO PARSE2
291   RETURN
      END



C     THIS SUBROUTINE WILL OBTAIN THE MEMBER CONSTANTS
C     FROM THE USER IN AN INTERACTIVE MODE
      SUBROUTINE CONS
C
      COMMON /GEOM   / MT,TALLY,NLOC,NT
      REAL MT(40,12),NLOC(40,2)
      INTEGER TALLY,NT
      REAL P1,P2,P3,P4
      INTEGER J,K,L
      PRINT*,'MEMBER CONSTANTS: E, ALPHA, DENS'
202   PRINT*,'MEMBER NUMBER>>'
      READ*,N
      IF(N.GT.TALLY)THEN
        PRINT*,'www INVALID MEMBER # www'
        GOTO 202
       END IF
      IF(N)200,201,213
200   RETURN  !GOTO PARSE1
201   PRINT*,'COPY MEMBER PROPERTIES FROM # ?'
       READ*,N
      IF(N.LE.0.OR.N.GT.TALLY)THEN
        PRINT*,'www INVALID MEMBER # www'
        GOTO 201
       END IF
       PRINT*,'  #        E        ALPHA        DENS'
       PRINT 298,N,MT(N,6),MT(N,11),MT(N,12)
298   FORMAT(' ',I3,2X,F12.0,2X,F9.8,5X,F10.4)
211   PRINT*,'COPY TO MEMBERS>>> START,END,INC'
      READ*,STAR,EN,INC
      IF(EN.GT.TALLY) EN=TALLY
      IF(INC.LE.0) GOTO 216
      IF(STAR.LE.0.OR.STAR.GT.TALLY.OR.STAR.GT.EN)THEN
216    PRINT*,'www INVALID MEMBER # www'
       GOTO 211
      END IF
      DO 212 I=STAR,EN,INC
        MT(I,6)=MT(N,6)
        MT(I,11)=MT(N,11)
        MT(I,12)=MT(N,12)
212   CONTINUE
      GOTO 202  !ANOTHER MEMBER
213   PRINT*,'>>>CONSTANTS:'
      READ*,P1
      MT(N,6)=P1
      MT(N,11)=P2
      MT(N,12)=P3
      GOTO 202  !ANOTHER MEMBER
      END
```

```
      SUBROUTINE DIGI
C     THIS ROUTINE WILL ALLOW THE USER TO DIGITIZE A FRAME FORM
C     A DIGITIZING TABLET
      COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
      COMMON /GEOM/   MT,TALLY,NLOC,NT
      REAL ZX,WX,ZY,WY,ROUND
      REAL MT(40,12),NLOC(40,2)
      INTEGER TALLY,NT
      REAL XI,XJ,YI,YJ,X,Y
      REAL FEET,AXISX,AXISY,DIMX(2),DIMY(2)
      REAL RTEN,RATST,DISTX,DISTY,MIDX,MIDY
      INTEGER ITICY,ITICX,NODJ,NODI,NOD1,NOD2,ID(2),IG(2),IGOT,IDAT
      IF(TALLY.GE.1) THEN
       CALL BELL
       I=I
       CALL BELL
       CALL CMCLOS
       CALL CMOPEN
       PRINT*,'*** CANNOT DIGITIZE IF MEMBERS ARE ALREADY PRESENT ***'
       GOTO 1000
      END IF
      CALL GRSTRT(4014,2)
      CALL SETGIN(2)
      CALL WINDOW(0.,130.,0.,100.)
      CALL VWPORT(0.,130.,0.,100.)
      CALL SQUARE
      CALL NEWPAG
      CALL CMCLOS
      CALL CMOPEN
      PRINT*,'LOCATE 2 POINTS...THE LOWER LEFT AND THE UPPER RIGHT'
      PRINT*,'THE WHOLE FRAME MUST LIE THE RECTANGLE '
      PRINT*,'OF WHICH THESE 2 POINTS ARE CORNERS'
      CALL LOCATE(2,DIMX,DIMY,IG,ID)
      PRINT*,'INPUT THE LARGEST DIMENSION LENGTH..IN FEET'
      READ*,FEET
      PRINT*,'ENTER THE ROUNDING INCREMENT..IF FEET'
      READ*,ROUND
      PRINT*,'LOCATE THE ORIGIN FOR THE AXIS'
      CALL LOCATE(1,AXISX,AXISY,IGOT,IDAT)
C  DETERMINE THE SIZE OF THE WINDOW TO ACCOMODATE THIS TABLET SIZE
C  ALSO DETERMINE THE TRANSFORMATION FACTORS FROM TABLET TO SCREEN
      DISTX=DIMX(2)-DIMX(1)
      DISTY=DIMY(2)-DIMY(1)
      IF(DISTX.GE.DISTY)THEN
       ZX=-((AXISX-DIMX(1))/DISTX)*FEET*1.25
       WX=ZX+FEET*1.25
       ZY=-1.25*(AXISY-DIMY(1))*FEET/DISTY
       WY=ZY+FEET*1.25
       RATST=FEET/DISTX
      ELSE
       ZY=-((AXISY-DIMY(1))/DISTY)*FEET*1.25
       WY=ZY+FEET*1.25
       ZX=-1.25*(AXISX-DIMX(1))*FEET/DISTX
       WX=ZX+FEET*1.25
       RATST=FEET/DISTY
      END IF
      TALLY=0
      CALL PAGE(ZX,WX,ZY,WY,ROUND)
      CALL CMCLOS
      CALL CMOPEN
```

```
C    LOCATE THE MEMBERS
C        THE END 1 AND 2 DONT MATTER THE COMPUTER WILL CALCULATE FOR YOU
C        BUT YOU HAVE TO LIVE WITH THE CONVENTION OR MOVE THE NODE TO
C        ADJUST
         PRINT*,'LOCATE THE MEMBERS NODE BY NODE'
         PRINT*,'COMPUTER WILL AUTOMATICALY SET LOCAL AXIS'
         PRINT*,'AND MEMBER START AND END'
         PRINT*,'TO QUIT.. LOCATE A POINT FAR TO THE RIGHT'
         OUT=DIMX(2)+10
         PRINT*,'NODES WILL BE EXACT AS YOU PLACE THEM...'
         PRINT*,'BUT TO REIDENTIFY A NODE YOU MUST BE WITHIN'
         PRINT*,'THE ROUNDING INCREMENT FROM THE NODE'
         PRINT*,'EACH NEW NODE LOCATED WILL BE SIGNALED BY A BEEP'
         PRINT*,'AND THEN THE NEW NODE NUMBER'
         CALL TXICUR(1)
         CALL TXFCUR(3)
         CALL WINDOW(0.,131.2,0.,100.)
         CALL VWPORT(0.,131.2,0.,100.)
11       CALL LOCATE(1,X,Y,IGOT,IDAT)
         IF(X.GT.OUT) GOTO 1000
         X=(X-AXISX)*RATST
         Y=(Y-AXISY)*RATST
         IF(X.GT.WX.OR.X.LT.ZX.OR.Y.GT.WY.OR.Y.LT.ZY) THEN
          CALL BELL
          I=I
          CALL BELL
          GOTO 11
         END IF
C          CALL ROUTINE TO SEE WHAT NODE IS SPECIFIED OR NEW
         CALL WINDOW(ZX,WX,ZY,WY)
         CALL VWPORT(5.,105.,0.,100.)
         CALL SAMENODE(X,Y,NLOC,NT,I,ROUND)
         CALL WINDOW(0.,131.2,0.,100.)
         CALL VWPORT(0.,131.2,0.,100.)
         NODI=I
C    LOCATE AND FIND NODE 2
12       CALL LOCATE(1,X,Y,IGOT,IDAT)
         X=(X-AXISX)*RATST
         Y=(Y-AXISY)*RATST
         IF(X.GT.WX.OR.X.LT.ZY.OR.Y.GT.WY.OR.Y.LT.ZY) THEN
          CALL BELL
          I=I
          CALL BELL
          GOTO 12
         END IF
         CALL WINDOW(ZX,WX,ZY,WY)
         CALL VWPORT(5.,105.,0.,100.)
         CALL SAMENODE(X,Y,NLOC,NT,I,ROUND)
         NODJ=I
C          IF SAME NODE ERROR GOTO 12
         IF(NODI.EQ.NODJ) THEN
          CALL BELL
          I=I
          CALL BELL
         CALL WINDOW(0.,131.2,0.,100.)
         CALL VWPORT(0.,131.2,0.,100.)
          GOTO 12
         END IF
C    CREATE MEMBER
C    IDENTIFY LOW END AND LENGTH AND NUMBER
```

```
      IF(NLOC(NODI,1).LT.NLOC(NODJ,1))GOTO 200   !KEEP  SAME
       IF(NLOC(NODI,1).EQ.NLOC(NODJ,1)) THEN
         IF(NLOC(NODI,2).LT.NLOC(NODJ,2)) GOTO 200 !KEEP SAME
       END IF
        ITEMP=NODJ
        NODJ=NODI
        NODI=ITEMP
200   TALLY=TALLY+1
      MT(TALLY,1)=NODI
      MT(TALLY,2)=NODJ
      XI=NLOC(NODI,1)
      YI=NLOC(NODI,2)
      XJ=NLOC(NODJ,1)
      YJ=NLOC(NODJ,2)
      MT(TALLY,5)= SQRT((XI-XJ)**2+(YI-YJ)**2)*12
C   CALCULATE THE MIDPOINT OF THE MEMBER AND NUMBER IT
      MIDX=(XJ-XI)/4+XI
      MIDY=(YJ-YI)/4+YI
      CALL WINDOW(ZX,WX,ZY,WY)
      CALL VWPORT(5.,105.,0.,100.)
      CALL TXSIZE(2,0,0)
      CALL MOVE(XI,YI)
      CALL DRAW(XJ,YJ)
      CALL MOVE(MIDX,MIDY)
      CALL INUMBR(TALLY,3)
      CALL WINDOW(0.,131.2,0.,100.)
      CALL VWPORT(0.,131.2,0.,100.)
      GOTO 11
1000  CALL SETGIN(1)
      CALL WINDOW(ZX,WX,ZY,WY)
      CALL VWPORT(5.,105.,0.,100.)
      RETURN
      END
```

```
C     THIS IS THE INTRODUCTION ROUTINE FOR THE INTERACTIVE GRAPHICS
C     STRUCTURAL ANALYSIS PROGRAM
C      WRITTEN BY DAVID E. RODGERS  1982
C
          INTEGER I,J,ITERM,IOPT
          REAL    TERM,POPT
          LOGICAL IGRA,ITAB,I4014
          CHARACTER*1  RES
          PRINT 100
100       FORMAT(1X,60('*'))
          PRINT 101
101       FORMAT(1X)
          PRINT 102,'  INTERACTIVE GRAPHIC STRUCTURAL ANALYSIS  '
102       FORMAT(1X,9('*'),A43,8('*'))
          PRINT 101
          PRINT 100
          PRINT 101
          PRINT*,'>DO YOU NEED INSTRUCTIONS? Y/N'
          READ 201,RES
201       FORMAT(A1)
          IF(RES.EQ.'N') GOTO 10
          PRINT 101
          PRINT*,'This program will create and analyse a 2-dimensional'
          print*,' plane frame structure in an interactive graphic mode'
          print 101
          print*,' -A TEKTRONIX 4014 or 4051 is needed to obtain graphics'
          print*,'    a digitizing tablet is optional for the 4014'
          print*,' -Responses for YES and NO may be shortened to 1 letter'
          print*,' -All commands must be at least 4 characters long'
          print*,' -Remember to SWITCH or STORE your Load Case before you '
          print*,'    execute the SOLUTION phase'
          print*,' -HELP sections are provided in all routines that ask'
          print*,'    for word commands'
          print*,' -The user is referred to the USERS MANUAL for'
          print*,'    further documentation'
10        PRINT*,'ARE YOU ON A GRAPHICS TERMINAL? Y/N'
          READ 201,RES
          IF(RES.EQ.'Y') THEN
           PRINT*,'>ENTER YOUR TERMINAL TYPE AND OPTION -- one of the following:'
           PRINT*,'  1) 4014 1      2) 4014 2      3) 4051 1'
           READ*,TERM,POPT
           ITERM=INT(TERM)
           IOPT =INT(POPT)
           IF(ITERM.EQ.4014.OR.ITERM.EQ.4051) GOTO 11
            PRINT*,'*** ERROR - BAD TERMINAL TYPE ***'
            GOTO 10
11         IF(IOPT.EQ.1.OR.IOPT.EQ.2) GOTO 12
            PRINT*,'*** ERROR - BAD TERMINAL OPTION ***'
            GOTO 10
12         IF(ITERM.LT.4015.AND.ITERM.GT.4013) THEN
            ITERM=4014
            I4014=.TRUE.
            PRINT*,'> DO YOU HAVE A DIGITIZING TABLET? Y/N'
            READ 201,RES
            IF(RES.EQ.'Y') THEN
             ITAB=.TRUE.
            ELSE
             ITAB=.FALSE.
            END IF
           ELSE
```

```
   IOPT=1
   ITERM=4051
   I4014=.FALSE.
   ITAB=.FALSE.
  END IF
  IGRA=.TRUE.
  CALL GRSTRT(ITERM,IOPT)
  CALL TXAM
  CALL TXSIZE(2,0,0)
  CALL NEWPAG
  CALL CMCLOS
  CALL CMOPEN
  CALL TEXT(15,'READY TO BEGIN:')
 ELSE
  IGRA=.FALSE.
  PRINT*,'READY TO BEGIN'
 END IF
 CALL PARSE(IGRA,ITAB,I4014)
 END
```

```
      SUBROUTINE LOAD(TALLY,NT)
      COMMON /LOADONE/MLTALLY,JLTALLY,MLOAD,JLOAD
      REAL MLOAD(40,6),JLOAD(40,3)
      INTEGER MLTALLY,JLTALLY
      INTEGER MC,JC,I,J,K,TALLY,NT,SET
      REAL P1,P2,P3,P4
      CHARACTER*4 STRING
      MC=MLTALLY
      JC=JLTALLY
      PRINT*,'LOAD SECTION'
600   PRINT*,'>> LOAD TYPE'
      READ 1,STRING
1     FORMAT(A4)
      K=INDEX('JPFX JPFY JMMZ MPFX MPFY MWFX MWFY MMMZ EXIT HELP',STRING)
      IF(K.EQ.0) THEN
C       CALL BEEP
        PRINT*,'*** INVALID LOAD TYPE ***'
        GOTO 600
      END IF
      K=K/5+1
      IF(K.EQ.9)GOTO 680
      IF(K.EQ.10)GOTO 691
      IF(K.LE.3) SET=NT
      IF(K.GT.3) SET=TALLY
602   PRINT*,'JOINT OR MEMBER NUMBER>>'
      READ*,N
              IF(N.LT.1.OR.N.GT.SET) THEN
        PRINT*,'*** INVALID NUMBER ***'
        GOTO 602
      END IF
601   PRINT*,'LOAD MAGNITUDE,LOC,LOC,LOC'
      P1=0
      P2=0
      P3=0
      P4=0
C    P1 TO P4 REPRESENT 'MAG,LOC,LOC,LOC'
      READ*,P1,P2,P3,P4
      IF(P1.EQ.0)THEN
       PRINT*,'*** INVALID LOAD MAGINITUDE ***'
       GOTO 601
      END IF
      IF(P2.LT.0.OR.P2.GT.1)GOTO 604
      IF(P3.LT.0.OR.P3.GT.1)GOTO 604
      IF(P4.LT.0.OR.P4.GT.1)GOTO 604
      GOTO (610,610,610,640,640,660,660,670,680,691),K
604    PRINT*,'*** INVALID LOCATION ***'
        GOTO 601
610   JC=JC+1
      JLOAD(JC,1)=N      !NODE #
      JLOAD(JC,2)=K      !LOAD TYPE  FX FY MZ
      JLOAD(JC,3)=P1     !MAGINITUDE
      GOTO 690
640   IF(P2.LE.0) GOTO 604
       MC=MC+1
       MLOAD(MC,1)=N
       MLOAD(MC,2)=1
       MLOAD(MC,3)=K-3
       MLOAD(MC,4)=P1
       MLOAD(MC,5)=P2
        IF(P3.LE.0) GOTO 690
```

```
       MC=MC+1
      MLOAD(MC,1)=N
      MLOAD(MC,2)=1
      MLOAD(MC,3)=K-3
      MLOAD(MC,4)=P1
      MLOAD(MC,5)=P3
       IF(P4.LE.0) GOTO 690
             MC=MC+1
      MLOAD(MC,1)=N
      MLOAD(MC,2)=1
      MLOAD(MC,3)=K-3
      MLOAD(MC,4)=P1
      MLOAD(MC,5)=P4
      GOTO 690
660  IF(P2.GE.P3) GOTO 604
     MC=MC+1
     MLOAD(MC,1)=N
     MLOAD(MC,2)=2
     MLOAD(MC,3)=K-5
     MLOAD(MC,4)=P1
     MLOAD(MC,5)=P2
     MLOAD(MC,6)=P3
      GOTO 690
670  IF(P2.LE.0) GOTO 604
     MC=MC+1
      MLOAD(MC,1)=N
      MLOAD(MC,2)=3
      MLOAD(MC,3)=3
      MLOAD(MC,4)=P1
      MLOAD(MC,5)=P2
       GOTO 690
690   GOTO 600
680  MLTALLY=MC
     JLTALLY=JC
     RETURN
691  PRINT*,'**LOAD HELP SECTION COMMANDS AVAILABLE:'
     PRINT*,'**JPFX  JPFY  JMMZ  MPFX  MPFY  MWFX  MWFY  MMMZ  EXIT'
     GOTO 600
     END
```

```
C     THIS IS A SUBROUTINE TO HANDLE THE DIFFERENT LOAD CASES
C     MAXIMUM OF 5 - INDEPENDANT LOAD CASES
C                5 -   DEPENDANT LOAD CASES
        SUBROUTINE LOADCASE(LCASE,NAME)
        COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
        COMMON /LOADONE/MLTALLY,JLTALLY,MLOAD,JLOAD
        COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
        REAL MCASE(5,40,6),JCASE(5,40,3)
        INTEGER CASES,NMCASE(5),NJCASE(5)
        REAL MLOAD(40,6),JLOAD(40,3)
        INTEGER MLTALLY,JLTALLY
        REAL COMB(5,5)
        INTEGER NCOMB,ACTLIST(10),ACASES
        CHARACTER*1 RES
        CHARACTER*30 NAME(10),NN
        CHARACTER*4 PAR
        INTEGER I,K,J,N,LCASE
        PRINT*,'--- LOAD CASES SECTION ---'
1001    PRINT 120,'OUT OF',CASES,'LOAD CASES'
        PRINT 121,LCASE,'LOAD CASE IS THE WORKING CASE'
120     FORMAT(' ',3X,A6,2X,I3,2X,A10)
121     FORMAT(' ',5X,I3,2X,A34)
        PRINT*,'LIST OF CURRENT LOAD CASES'
         DO 1000 I =1,CASES
1000      PRINT 100,I,NAME(I)
100     FORMAT(' ',3X,I3,3X,A30)
        PRINT*,' '
        PRINT*,'LIST OF CURRENT LOAD COMBINATIONS'
         DO 1002 I=1,NCOMB
          J=I+5
          PRINT 100,I,NAME(J)
1002     CONTINUE
        PRINT*,' '
1       PRINT*,'>>NEXT LOAD CASE OR EXIT'
        READ 101,PAR
101     FORMAT(A4)
        K=INDEX('CREA SWIT STOR RENA LIST COMB ACTI HELP EXIT',PAR)
        IF(K.EQ.0) THEN
         PRINT*,'*** LIST WHAT ?? ***'
         GOTO 1
        END IF
        K=(K+4)/5
        GOTO (10,20,29,30,1001,50,60,70,80),K
10      PRINT*,'CREATE A NEW LOAD CASE'
        IF(CASES+1.GT.5) THEN
         PRINT*,'*** SORRY MAXIMUM LOAD CASES ***'
         PRINT*,'    ONLY 5 INDEPENDANT LOAD '
         PRINT*,'    ALLOWED -- NEW CASE NOT CREATED'
        GOTO 1
        END IF
        CASES=CASES+1
        PRINT 102,'CASE NUMBER => ',CASES
102     FORMAT(' ', A15,I3)
        PRINT*,'ENTER A NAME FOR THIS CASE -- 30 CHAR MAX'
        READ 103,NN
103     FORMAT(A30)
        NAME(CASES)=NN
        PRINT 104 ,'LOAD CASE ',CASES,' SUCCESSFULLY CREATED'
104     FORMAT(' ',A10,I3,A21)
        PRINT *,'NOTE -- OLD LOAD CASE STILL ACTIVATED'
```

```
      PRINT*,'  LOAD CASE ',LCASE,' STILL ACTIVE'
      GOTO 1
20    PRINT*,'SWITCH TO LOAD CASE # ?'
      READ*,N
      IF(N.EQ.0) GOTO 1
      IF(N.LT.0.OR.N.GT.CASES) THEN
       PRINT*,'*** INVALID LOAD CASE NUMBER ***'
       GOTO 20
      END IF
      GOTO 28
29    N=LCASE
28    NJCASE(LCASE)=JLTALLY
      NMCASE(LCASE)=MLTALLY
      DO 21 I=1,JLTALLY
       JCASE(LCASE,I,1)=JLOAD(I,1)
       JCASE(LCASE,I,2)=JLOAD(I,2)
       JCASE(LCASE,I,3)=JLOAD(I,3)
21    CONTINUE
      DO 22 I=1,MLTALLY
       DO 23 K=1,6
23      MCASE(LCASE,I,K)=MLOAD(I,K)
22    CONTINUE
      LCASE=N
      JLTALLY=NJCASE(LCASE)
      MLTALLY=NMCASE(LCASE)
      DO 24 I=1,JLTALLY
       JLOAD(I,1)=JCASE(LCASE,I,1)
       JLOAD(I,2)=JCASE(LCASE,I,2)
24     JLOAD(I,3)=JCASE(LCASE,I,3)
      DO 25 I=1,MLTALLY
       DO 26 K=1,6
26      MLOAD(I,K)=MCASE(LCASE,I,K)
25    CONTINUE
      RETURN
30    PRINT*,'>>> RENAME LOAD CASE # ?'
      READ*,N
      IF(N.EQ.0) GOTO 1
      IF(N.LT.0.OR.N.GT.CASES) THEN
       PRINT*,'*** INVALID LOAD CASE NUMBER ***'
       GOTO 30
      END IF
      PRINT 106,'OLD NAME -- ',NAME(N)
106   FORMAT(' ',A13,A30)
      PRINT*,'ENTER NEW NAME -- 30 CHAR MAX'
      READ 103,NN
      NAME(N)=NN
      GOTO 1
50    PRINT*,'COMBINE LOADING CASES'
      PRINT 110,' CURRENTLY ',CASES,' LOAD CASES        '
      PRINT  110,' CURRENTLY ',NCOMB ,' LOAD COMBINATIONS'
110   FORMAT(' ',A10,I3,A17)
      PRINT*,'                           1      2      3      4      5'
      DO 51 I=1,NCOMB
       PRINT 116,'LOAD COMBINATION ',I,' = ',(COMB(I,J),J=1,5)
116    FORMAT(' ',A17,I3,A3,5(F5.3,2X))
51    CONTINUE
57    PRINT*,'LOAD COMBINATION => 1 TO 5 '
      PRINT*,'>>> ENTER LOAD COMBINATION # '
      READ*,N
      IF(N.EQ.0) GOTO 1
```

```
      IF(N.LT.0.OR.N.GT.NCOMB+1) THEN
       PRINT*,'*** INVALID LOAD COMBINATION ***'
       GOTO 57
      END IF
      IF(N.EQ.NCOMB+1) GOTO 52
      PRINT 111,'ALTERING LOAD COMBINATION  => ',N
111   FORMAT(' ',A26,I3)
56    PRINT 112,'LOAD COMBINATION ',N,' NOW ALL 0.0 '
112   FORMAT(' ',A17,I3,A13)
      DO 53 I=1,CASES
54     PRINT 113,' LOAD CASE ',I,' TIMES X.XX'
113    FORMAT(' ',A11,I3,A11)
       READ*,X
       IF(X.GT.3.0) THEN
        PRINT*,'*** BAD LOAD FACTOR ***'
        GOTO 54
       END IF
      COMB(N,I)=X
53    CONTINUE
      DO 55 I=CASES+1,5
55     COMB(N,I)=0
      IF(COMB(N,1).EQ.0.AND.COMB(N,2).EQ.0.AND.COMB(N,3).EQ.0.
     * AND.COMB(N,4).EQ.0.AND.COMB(N,5).EQ.0) THEN
       PRINT*,'*** ERROR IN COMBINATION FACTORS ***'
       PRINT*,'        ALL FACTORS = 0.0'
      GOTO 56
      END IF
      GOTO 1
52    PRINT*,'CREATE A NEW LOAD COMBINATION'
      NCOMB=NCOMB+1
      IF(NCOMB.GT.5) THEN
       PRINT*,'*** ERROR TOO MANY COMBINATIONS ***'
       PRINT*,'    ONLY 5 DEPENDANT COMB ALLOWED '
       NCOMB=NCOMB-1
       GOTO 1
      END IF
      PRINT*,'   ENTER A NAME FOR THIS LOAD COMBINATION'
      READ 103,NAME(NCOMB+5)
      GOTO 56
60    PRINT*,'ACTIVATE LOAD CASES'
      PRINT*,' NOTE: ALL LOAD CASES ARE ACTIVE FOR THE SOLUTION'
      PRINT*,'       ALL LOAD COMBINATIONS ARE ACTIVE FOR POST-PROCESS'
      PRINT*,'>>THIS SECTION TO ACTIVATE ONLY CERTAIN INDEPENTANT LOAD'
      PRINT*,' CASES FOR THE POST-PROCESSING'
      ACASES=0
      DO 62 I=1,10
62     ACTLIST(I)=0
      DO 61 I=1,CASES
       PRINT 114,'LOAD CASE ',I,' ACTIVATE FOR POST-PROCESS ? Y/N'
114    FORMAT(' ',A10,I3,A32)
       READ 115,RES
115    FORMAT(A1)
       IF(RES.EQ.'Y') THEN
        ACASES=ACASES+1
        ACTLIST(ACASES)=I
       END IF
61    CONTINUE
      IF(NCOMB.EQ.0) GOTO 65
      DO 63 I=1,NCOMB
       ACASES=ACASES+1
```

```
          ACTLIST(ACASES)=I+5
63       CONTINUE
65       PRINT*,'THIS IS A PRINTOUT OF ACASES'
         DO 66 I=1,ACASES
          PRINT 130,'  ACTIVE LOAD CASES ',ACTLIST(I),NAME(ACTLIST(I))
130        FORMAT(1X,A20,I3,2X,A30)
66       CONTINUE
         GOTO 1
70       PRINT*,'RESULTS HELP SECTION -- COMMANDS AVAILABLE:'
         PRINT*,'  CREATE  SWITCH  RENAME  LIST  COMBINE'
         PRINT*,'  STORE  ACTIVATE  HELP  EXIT'
          GOTO 1
80       RETURN
         END
```

```
        SUBROUTINE MREL(TALLY)
        COMMON /RELEASE/ MBREL,SREL,STALLY
        INTEGER MBREL(40),SREL(40),STALLY
        CHARACTER*1 STRING
        INTEGER TALLY
        CHARACTER*5 REL
        INTEGER N,I,STAR,EN,INC,K
        PRINT*,'MEMBER END RELEASE: START OR END OR BOTH'
502     PRINT*,'MEMBER NUMBER ..'
        READ*,N
        IF(N)500,501,511
500     RETURN
501     PRINT*,'COPY MEMBER RELEASES FROM #?'
        READ*,N
        IF(N.LT.0) GOTO 502
        IF(N.LE.0.OR.N.GT.TALLY) THEN
         PRINT*,'*** INVALID MEMBER NUMBER ***'
         GOTO 501
        END IF
        IF(MBREL(N).EQ.0)  REL='NONE'
        IF(MBREL(N).EQ.2)  REL='START'
        IF(MBREL(N).EQ.3)  REL='END'
        IF(MBREL(N).EQ.1)  REL='BOTH'
        PRINT 598,'MEMBER ',N,' RELEASE ',REL
598     FORMAT(' ',A7,I3,A10,A5)
505     PRINT*,'COPY TO MEMBERS: START,END,INC '
        READ*,STAR,EN,INC
        IF(STAR.LT.0.OR.STAR.GT.EN.OR.STAR.GT.TALLY.OR.INC.LE.0) THEN
         CALL BELL
         PRINT*,'*** INVALID MEMBER NUMBER ***'
         GOTO 505
        END IF
        IF(EN.GT.TALLY) EN = TALLY
        DO 503 I=STAR,EN,INC
503      MBREL(I)=MBREL(N)
        GOTO 502   !ANOTHER JOINT
511     IF(N.GT.TALLY)THEN
         CALL BELL
         PRINT*,'*** INVALID MEMBER NUMBER ***'
         GOTO 502
        END IF
        PRINT*,'RELEASE :'
510     READ 597,STRING
597     FORMAT(A1)
        K=INDEX('BSE',STRING)
        IF(K.EQ.0) THEN
         CALL BELL
         PRINT*,'***RELEASE WHAT ? ***'
         GOTO 510
        END IF
        MBREL(N)=K
        GOTO 502        !ANOTHER MEMBER
        END
```

```
C
      SUBROUTINE PARSE(IGRA,ITAB,I4014)
C     SUBROUTINE PARSE-- THIS IS THE EXECUTIVE DRIVER PROGRAM
C     FOR THE REST OF THE SUBROUTINES
      COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
      COMMON /GEOM/   MT,TALLY,NLOC,NT
      COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
      COMMON /LOADONE/MLTALLY,JLTALLY,MLOAD,JLOAD
      COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
      COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
      COMMON /RELEASE/MBREL,SREL,STALLY
      COMMON /ASSEMB/ BMAX,BASS
      REAL ZX,WX,ZY,WY,ROUND
      REAL MT(40,12),NLOC(40,2)
      INTEGER TALLY,NT
      REAL MCASE(5,40,6),JCASE(5,40,3)
      INTEGER CASES,NMCASE(5),NJCASE(5)
      REAL MLOAD(40,6),JLOAD(40,3)
      INTEGER MLTALLY,JLTALLY
      REAL COMB(5,5)
      INTEGER NCOMB,ACTLIST(10),ACASES
      REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
      REAL  FEMDIS(5,40,6)
      INTEGER MBREL(40),SREL(40),STALLY
      REAL BASS(120,120)
      INTEGER BMAX
      REAL SM(6,6),DISP(6)
      INTEGER LCASE,NERASE,MERASE
      REAL AX,E,XLEN,ZIZ,S,C,MEMMAX(40,3)
      INTEGER I,J,K,TEMP,CASE
      CHARACTER*30 NAME(10)
      CHARACTER*4 STRING
      LOGICAL SET(7),PASS(10),IGRA,ITAB,I4014
      NT=0
      BMAX=0
      TEMP=-100
      TALLY=0
      NERASE=0
      MERASE=0
      LCASE=1
      CASES=1
      DO 1011 I=1,10
1011   NAME(I)='NONE GIVEN'
      SET(1)=.TRUE.
      SET(2)=.TRUE.
      SET(3)=.TRUE.
      SET(4)=.FALSE.
      SET(5)=.FALSE.
      SET(6)=.FALSE.
      SET(7)=.FALSE.
100   PRINT*,'COMMAND ?'
      READ 200,STRING
200   FORMAT(A4)
      K=INDEX('BUIL SETU STOR BAYS DIGI PROP CONS SUPP MREL
     * LCAS LOAD DELE CHAN PLOT',STRING)
      IF(K.LT.1) GOTO 101
      K=(K+4)/5
      GOTO(1,2,3,4,5,6,7,8,9,10,11,12,13,14),K
101   K=INDEX('REDR LIST ZOOM DATA HELP SOLV QUIT INDI RESU ZERO
     * SAVE REST,DEFL ANSW',STRING)
```

```
      IF(K.LT.1)THEN
       PRINT*,'ww COMMAND NOT FOUND ww'
       GOTO 100
      END IF
      K=(K+4)/5
      GOTO(16,17,21,22,18,19,20,23,25,26,27,28,29,30),K
      PRINT*,'**** ERROR ****'
      GOTO 100
1     CALL BUIL
      GOTO 100
2     I=I
C     CALL SETU
      GOTO 100
3     CALL STORIES
      GOTO 16
4     CALL BAYS
      GOTO 16
5     CALL DIGI
      GOTO 100
6     CALL PROP
      GOTO 100
7     CALL CONS
      GOTO 100
8     CALL SUPP(NT)
      GOTO 100
9     CALL MREL(TALLY)
      GOTO 100
10    CALL LOADCASE(LCASE,NAME)
      GOTO 100
11    CALL LOAD(TALLY,NT)
      GOTO 100
12    CALL ERASE(NERASE,MERASE)
      GOTO 100
13    CALL CHAN
      GOTO 100
14    CALL GRAPHIK(SET)
      GOTO 100
16    CALL REDR
      GOTO 100
17    CALL LIS
      GOTO 100
23    CALL INDIV2(NAME,MEMMAX,I4014)
      GOTO 100
21    CALL ZOOMIO
      GOTO 16
22    CALL OUT(NAME)
      GOTO 100
25    CALL RESULT(CASES,NCOMB)
      GOTO 100
26    CALL ZERO
      GOTO 100
27    CALL SAVE(NAME)
      GOTO 100
28    CALL RESTORE(NAME,LCASE)
      CALL REDR
      GOTO 100
29    CALL DEFL(CASES,NCOMB,NAME)
      GOTO 100
30    CALL ANSWERS(CASES,NCOMB)
      GOTO 100
```

```
19      CALL CONSIS(PASS)
         DO 192 I=1,5
          IF(PASS(I)) GOTO 192
          GOTO 100
192     CONTINUE
        CALL ZERO
        CALL JCASEACT(ACT)
        CALL MCASEACT
        DO 190 I=1,TALLY
         E   =MT(I,6)
         AX  =MT(I,7)
         ZIZ=MT(I,8)
         NI  =MT(I,1)
         NJ  =MT(I,2)
         XLEN=MT(I,5)   ! IN INCHES
          S=(NLOC(NJ,2)-NLOC(NI,2))/(XLEN/12)
          C=(NLOC(NJ,1)-NLOC(NI,1))/(XLEN/12)
          CASE=MBREL(I)+1
           CALL LOCASE(XLEN,E,ZIZ,AX,CASE,SM)
           CALL GLOBSTIF(SM,S,C)
           CALL BNASMBL(NI,NJ,SM)
190     CONTINUE
        DO 191 I=1,TALLY      ! FIND BMAX
          TEMP=ABS(MT(I,1)-MT(I,2))
          IF(TEMP.GT.BMAX) BMAX=TEMP
191     CONTINUE
        BMAX=BMAX*3+3
        CALL SOLVE(NT,CASES)
        CALL CASEFORC(CASES)
        CALL CASEMOSH
        CALL RECASE(CASES)
        CALL FACTOR(NT,TALLY,CASES)
        CALL ENVEL(CASES,TALLY,MEMMAX)
        GOTO 100
18      PRINT*,'* PARSE HELP SECTION  COMMANDS AVAILABLE:'
        PRINT*,'   BUIL BAYS STOR DIGE PROP CONS SUPP MREL'
        PRINT*,'   LOAD LCAS LIST CHAN DELE DATA RESU SAVE REST'
        PRINT*,'*  REDR PLOT ZOOM INDI DEFL ANSW'
        PRINT*,'*  SOLV ZERO HELP QUIT'
        GOTO 100
20      STOP
        END
```

```
C
      SUBROUTINE PROP
C
      COMMON /GEOM   / MT,TALLY,NLOC,NT
      REAL MT(40,12),NLOC(40,2)
      INTEGER TALLY,J,K,L,N,NT
      PRINT*,'MEMBER PROPERTIES: Ax, Iz, Sx, Q'
102   PRINT*,'>>MEMBER NUMBER '
      READ*,N
       IF(N.GT.TALLY)THEN
        PRINT*,'www INVALID MEMBER # www'
        GOTO 102
       END IF
      IF(N) 100,101,113
100   RETURN
101   PRINT*,'COPY MEMBER PROPERTIES FROM # ?'
      READ*,N
      IF(N.LE.0.OR.N.GT.TALLY)THEN
       PRINT*,'www INVALID MEMBER # www'
      GOTO 101
      END IF
        PRINT*,'  #        Ax       Iz        Sx      Q'
        PRINT 198,N,MT(N,7),MT(N,8),MT(N,9),MT(N,10)
198   FORMAT(' ',I3,2X,F6.2,2X,F8.2,2X,F8.2,2X,F8.2)
111   PRINT*,'COPY TO MEMBERS>>>START,END,INC'
      READ*,STAR,EN,INC
      IF(INC.LE.0) GOTO 115
      IF(EN.GT.TALLY) EN=TALLY
      IF(STAR.LE.0.OR.STAR.GT.TALLY.OR.STAR.GT.EN)THEN
115    PRINT*,'www INVALID MEMBER # www'
         GOTO 111
        END IF
      DO 112 I=STAR,EN,INC
        MT(I,7)=MT(N,7)
        MT(I,8)=MT(N,8)
        MT(I,9)=MT(N,9)
         MT(I,10)=MT(N,10)
112   CONTINUE
      GOTO 102  !ANOTHER MEMBER
C      HERE NEW MEMBER
113   PRINT*,'PROPERTIES>>>'
      READ*,P1,P2
      IF(P1.LE.0.OR.P2.LE.0.OR.P3.LT.0.OR.P4.LT.0)THEN
        PRINT*,'www ERROR IN PROPERTIES www'
        GOTO 113
      END IF
       MT(N,7)=P1
       MT(N,8)=P2
       MT(N,9)=P3
       MT(N,10)=P4
      GOTO 102  !ANOTHER MEMBER
      END
```

```
C    THIS IS FOR DIGI ONLY
C   SUBROUTINE SAMENODE
        SUBROUTINE SAMENODE(X,Y,NLOC,NT,I,ROUND)
        REAL NLOC(40,2),ROUND,X,Y
        INTEGER NT,I
        DO 10 I=1,NT
          IF(X.LT.NLOC(I,1)+ROUND.AND.X.GT.NLOC(I,1)-ROUND) THEN
           IF(Y.LT.NLOC(I,2)+ROUND.AND.Y.GT.NLOC(I,2)-ROUND) THEN
            X=NLOC(I,1)
            Y=NLOC(I,2)
            GOTO 20
          END IF
          END IF
10      CONTINUE
C     IF REACHES HERE IT IS A NEW NODE
        CALL BELL
        NT=NT+1
        NLOC(NT,1)=X
        NLOC(NT,2)=Y
        I=NT
C    NEXT DRAW THE NODE
20      CALL MOVE(X,Y)
        CALL TXSIZE(3,0,0)
        CALL INUMBR(I,3)
        RETURN
        END



C       THIS IS A SUBROUTINE TO FIND THE NODE THAT WAS POINTED TO
C        THEN COPY THE CORRECT COORDINATES
        SUBROUTINE SAMENODES(XN,YN,NODE,NT,NLOC,R)
        REAL XN,YN,NLOC(40,2),R
        INTEGER NT,NODE
        DO 10 I=1,NT
         IF(XN+R.GT.NLOC(I,1).AND.XN-R.LT.NLOC(I,1)) THEN
         IF(YN+R.GT.NLOC(I,2).AND.YN-R.LT.NLOC(I,2)) THEN
          XN=NLOC(I,1)
          YN=NLOC(I,2)
          NODE=I
          RETURN
         END IF
         END IF
10      CONTINUE
C    IF REACHES HERE NO MATCH
        NODE=0
        RETURN
        END
```

```
C
        SUBROUTINE SQUARE
C       THIS WILL ALLOW THE USER TO BE SURE THAT HIS DRAWING
C       IS SQUARE TO THE TABLET
C    THIS IS A SUBROUTINE TO BE SURE YOUR DRAWING
C     IS SQUARE TO THE TABLET AND DOTS ON THE SCREEN
C    THIS IS A SUBROUTINE TO BE SURE YOUR DRAWING
C     IS SQUARE TO THE TABLET AND DOTS ON THE SCREEN
        INTEGER RESPONSE
        REAL I,J,HX,XY,HXX,HYY
        CHARACTER*1 RES
        CALL WINDOW(130.,0.,100.,0.)
        CALL VWPORT(130.,0.,100.,0.)
        CALL NEWPAG
        CALL TXFCUR(1)
        CALL TXAM
        CALL WINDOW(0.,131.2,0.,100.)
        CALL VWPORT(0.,131.2,0.,100.)
C    DRAW THE DOTS
        DO 11 I=10,120,15
         DO 12 J=10,90,5
          CALL MOVE(I,J)
          CALL DRAW(I,J)
12      CONTINUE
11      CONTINUE
        CALL HOME
        CALL CMCLOS
        CALL CMOPEN
        CALL TEXT(31,'PLACE THE DRAWING ON THE TABLET')
        CALL TEXT(42,'TO BE SURE THAT YOUR PLAN IS SQUARE TO THE')
        CALL TEXT(37,' TABLET AND THE DOTS ON THE SCREEN...')
        CALL TEXT(46,'LOCATE THE ENDPOINTS OF A LONG HORIZONTAL LINE')
        CALL WHERE(TX,TY)
10      CALL LOCATE(1,HX,HY,IDAT,IGOT)
        CALL LOCATE(1,HXX,HYY,IDAT,IGOT)
        CALL MOVE(HX,HY)
        CALL DRAW(HXX,HYY)
        CALL MOVE(TX,TY)
        CALL CMCLOS
        CALL CMOPEN
        CALL TEXT(39,'DOES THIS LINE UP WITH THE DOTS?? Y/N')
        CALL CMCLOS
        CALL CMOPEN
        READ 90,RES
90      FORMAT(A1)
        CALL CMCLOS
        CALL CMOPEN
        IF(RES.EQ.'Y') GOTO 100
        CALL TEXT(43,'*ADJUST THE DRAWING AND LOCATE ANOTHER LINE')
        CALL WHERE(TX,TY)
        GOTO 10
100     RETURN
        END
```

```
C    SUBROUTINE TO CREATE STORIES TO A FRAME
C        GIVEN THE NODES OF ATTACHMENT AND THE STORY HEIGHT
         SUBROUTINE STORIES
C
         COMMON /SCREEN/ ZX,WX,ZY,WY,ROUND
         COMMON /GEOM/   MT,TALLY,NLOC,NT
         REAL ZX,WX,ZY,WY,ROUND
         REAL MT(40,12),NLOC(40,2)
         INTEGER TALLY,NT
         REAL HEIG,TX(15,15),TY(15,15),TNODE(15,15)
         INTEGER TNT,TT,UP,N,N1,IGOT,IDAT,NODE
         REAL XN,YN
         INTEGER STORT
         CALL WINDOW(ZX,WX,ZY,WY)
         CALL VWPORT(5.,105.,0.,100.)
1        PRINT*,'ENTER THE NUMBER OF ADDITIONAL STORIES'
         READ*,UP
         IF(UP.EQ.0) RETURN
         IF(UP.LT.0.OR.UP.GT.100) GOTO 1
3        PRINT*,'INPUT THE NEXT FLOOR HEIGHT'
         READ*,HEIG
         IF(HEIG.LE.0)GOTO 3
2        PRINT*,'ENTER THE NUMBER OF BAYS IN THE NEXT FLOOR'
         READ*,N
         IF(N.LE.0)GOTO 1
         N1=N+1    !NUMBER OF POINTS OF ATTACHMENT
         PRINT*,'LOCATE THE POINTS OF ATTACHMENT--FROM LEFT TO RIGHT'
         DO 10 I=1,N1
11        CALL LOCATE(1,XN,YN,IGOT,IDAT)
          CALL SAMENODES(XN,YN,NODE,NT,NLOC,ROUND)
          IF(NODE.EQ.0) THEN
           PRINT*,'SORRY--NODE NOT MATCHED--TRY AGAIN'
           GOTO 11
          END IF
          TX(1,I)=XN
          TY(1,I)=YN
          TNODE(1,I)=NODE
10       CONTINUE
         STORT=TALLY
         TNT=NT
         DO 20 I=2,UP+1
          TNODE(I,1)=TNT+1
          TX(I,1)    =TX(1,1)
          TY(I,1)    =TY(I-1,1)+HEIG
          NLOC(TNT+1,1)=TX(I,1)
          NLOC(TNT+1,2)=TY(I,1)
          TNT=TNT+1
         DO 30 J=2,N1
          TNODE(I,J)=TNT+1
          TX(I,J)    =TX(1,J)
          TY(I,J)    =TY(I,1)
          NLOC(TNT+1,1)=TX(I,J)
          NLOC(TNT+1,2)=TY(I,J)
          TNT=TNT+1
30       CONTINUE
20       CONTINUE
C   CREATE THE NEW MEMBERS
         TT=TALLY
         DO 40 I=2,UP+1
          MT(TT+1,1)=TNODE(I-1,1)
```

```
      MT(TT+1,2)=TNODE(I,1)
      TT=TT+1
      DO 50 J=2,NT
       MT(TT+1,1)=TNODE(I,J-1)
       MT(TT+1,2)=TNODE(I,J)
       MT(TT+2,1)=TNODE(I-1,J)
       MT(TT+2,2)=TNODE(I,J)
       TT=TT+2
50     CONTINUE
40    CONTINUE
      TALLY=TT
      NT=TNT
C     GO TO DRAW THE NEW MEMBERS AND LABEL THE NODES AND MEMBERS
      DO 60 I=STORT,TALLY
       MT(I,5)=SQRT((NLOC(MT(I,1),1)-NLOC(MT(I,2),1))**2+
     *              (NLOC(MT(I,1),2)-NLOC(MT(I,2),2))**2)*12
60    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SUPP(NT)
      COMMON /RELEASE/ MBREL,SREL,STALLY
      INTEGER MBREL(40),SREL(40),STALLY
      INTEGER NT,TES,N,I,J,STAR,EN,INC,K
      CHARACTER*2 FY,FX,MZ,STRING
      PRINT*,'SUPPORT/SUPPORT RELEASE: TX,TY,RZ,TT,XR,YR,NO'
302   PRINT*,'JOINT NUMBER>>'
      READ *,N
      IF(N.GT.NT)THEN
       PRINT*,'www INVALID JOINT # www'
        GOTO 302
      END IF
        IF(N)300,301,310
300   N=N
      DO 333 I=1,NT
333   CONTINUE
      RETURN
301   PRINT*,'COPY SUPPORT CONDITIONS FROM JOINT #?'
      READ*,N
      IF(N.LE.0.OR.N.GT.NT) THEN
       PRINT*,'*** INVALID JOINT # ***'
       GOTO 301
      END IF
       IF(SREL(N).EQ.0)THEN
       PRINT*,'*** JOINT ',N,' NOT A SUPPORT ***'
       GOTO 301
       END IF
306    FX=' '
        FY= ' '
        MZ=' '
      TES=SREL(N)
      IF(TES.EQ.111) THEN  !NO RELEASES
       FX='TX'
       FY='TY'
       MZ='RZ'
              ELSE IF(TES.EQ.110) THEN
               FX='TX'
               FY='TY'
               ELSE IF(TES.EQ.100) THEN
                FX='TX'
                ELSE IF(TES.EQ.11) THEN
                       FY='TY'
                       MZ='RZ'
                        ELSE IF(TES.EQ.10) THEN
                         FY='TY'
                              ELSE
                               MZ='R'
      END IF
397   PRINT 398,'JOINT ',N,'    FIXED  ',FX,FY,MZ
398   FORMAT(' ',A6,I3,A10,A2,1X,A2,1X,A2)
305   PRINT*,'COPY TO JOINT>>> START,END,INC'
      READ*,STAR,EN,INC
      IF(EN.GT.NT) EN=NT
      IF(INC.LE.0) GOTO 350
      IF(STAR.EQ.0.OR.STAR.GT.NT.OR.STAR.GT.EN) THEN
C       CALL BEEP
350     PRINT*,'*** INVALID JOINT NUMBER ***'
        GOTO 305
        END IF
      IF(EN.GT.NT)EN=NT
```

```
        DO 303 J=STAR,EN,INC
              IF(SREL(J).GE.1) STALLY=STALLY-1
              SREL(J)=SREL(N)
      STALLY=STALLY+1
303   CONTINUE
       GOTO 302  !ANOTHER JOINT
310   PRINT*,'RELEASE DIRECTION>>>'
              IF(SREL(N).GE.1) STALLY=STALLY-1    !ALREADY COUNTED
      READ 399,STRING
399   FORMAT(A2)
      K=INDEX('TX TY RZ TT XR YR NO',STRING)
      IF(K.EQ.0) THEN
C       CALL BEEP
        PRINT*,'*** INVALID DIRECTION ***'
        GOTO 310
      END IF
      K=K/3+1
      GOTO(321,322,323,324,325,326,327),K
321   SREL(N)= 11
      GOTO 330
322   SREL(N)=101
      GOTO 330
323   SREL(N)=110
      GOTO 330
324   SREL(N)=  1
      GOTO 330
325   SREL(N)= 10
      GOTO 330
326   SREL(N)=100
      GOTO 330
327   SREL(N)=111
330   STALLY=STALLY+1
      GOTO 302   !ANOTHER JOINT
      END
```

```
C   THIS SUBROUTINE WILL ZERO OUT THE REUSABLE VARIABLES
C   SO ANOTHER RUN CAN BE MADE
C
         SUBROUTINE ZERO
C
         COMMON /FORC1/ SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
         COMMON /ASSEMB/ BMAX,BASS
         REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3)
         REAL ACT(10,120),FEMDIS(5,40,6)
         INTEGER I,J,K,L,M
         REAL BASS(120,120)
         INTEGER BMAX
         DO 10 M=1,40
          DO 11 J=1,12
           DO 12 K=1,6
12          EMCASE(J,M,K)=0
           DO 13 K=1,3
            DO 14 L=1,21
14           SECTFORC(J,M,K,L)=0
13         CONTINUE
11        CONTINUE
          DO 15 J=1,5
           DO 16 K=1,6
16          FEMDIS(J,M,K)=0
15        CONTINUE
         DO 17 J=1,10
          DO 18 K=1,3
18         SUPCASE(J,M,K)=0
17       CONTINUE
10       CONTINUE
         DO 20 J=1,120
          DO 21 I=1,120
21         BASS(J,I)=0
          DO 22 I=1,10
22         ACT(I,J)=0
20       CONTINUE
         BMAX=0
         RETURN
         END
```

```
      SUBROUTINE CASEFORC(CASES)
C     THIS SUBROUTINE WILL TAKE THE DISPLAEMENTS FROM THE ANALYSIS
C      AND TURN THEN INTO FORCES AT THE MEMBER ENDS
C       ACT HOLDS THE ANALYSIS DISPLACEMENTS..LOCSTIF CALCULATES
C        THE LOCAL STIFFNESS (WILL MODIFY FOR RELEASED MEMBER END)
C       ...DISP HOLDS THE DISPLACEMENTS IN TH MEMBER LOCAL COORD
C   EMCASE HOLDS THE EQUAVELENT JOINT LOADS DUE TO MEMBER LOADS
C    THEN EMCASE HOLDS THE RESULTANT FORCES AT THE MEMBER ENDS
C   FEMDIS HOLDS THE FORCES AT THE MEMBER END DUE TO THE JOINT DISPLACEMENT
C     ONLY  .. THIS IS LATER USED IN CASEMOSH
C   SIGN CONVENTION-- CCW DISPLACEMENT AND MOMENT IS "+"
      COMMON /GEOM/   MT,TALLY,NLOC,NT
      COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
      COMMON /RELEASE/MBREL,SREL,STALLY
      REAL MT(40,12),NLOC(40,2)
      INTEGER TALLY,NT
      REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
      REAL  FEMDIS(5,40,6)
      INTEGER MBREL(40),SREL(40),STALLY
      REAL S,C,SM(6,6),DISP(6),FORC(6),IZ,L,E,A
      INTEGER NI,NJ,I,J,CASE,CASES
      DO 10 I=1,TALLY
        E=MT(I,6)
        A=MT(I,7)
       IZ=MT(I,8)
        L=MT(I,5)
       NI=MT(I,1)
       NJ=MT(I,2)
        CASE=MBREL(I)+1
      CALL LOCASE(L,E,IZ,A,CASE,SM)
       C=(NLOC(NJ,1)-NLOC(NI,1))/(L/12)
       S=(NLOC(NJ,2)-NLOC(NI,2))/(L/12)
      DO 12 J=1,CASES
      JJ=(NI-1)*3+1
      JK=(NJ-1)*3+1
      DISP(1)=C*(ACT(J,JJ))+S*(ACT(J,JJ+1))
      DISP(2)=-S*(ACT(J,JJ))+C*(ACT(J,JJ+1))
      DISP(3)=ACT(J,JJ+2)
      DISP(4)=C*(ACT(J,JK))+S*(ACT(J,JK+1))
      DISP(5)=-S*(ACT(J,JK))+C*(ACT(J,JK+1))
      DISP(6)=ACT(J,JK+2)
      CALL MULT6X1(SM,DISP,FORC)
      FORC(3)=FORC(3)/12.0    ! NOW IN F-K
      FORC(6)=FORC(6)/12.0    ! NOW IN F-K
      DO 21 K=1,6
21     FEMDIS(J,I,K)=FORC(K)
      DO 11 K=1,6
       EMCASE(J,I,K)=FORC(K)-EMCASE(J,I,K)
11    CONTINUE
12    CONTINUE
10    CONTINUE
      RETURN
      END
```

```
C    THIS IS THE SUBROUTINE THAT WILL EXAMINE THE STRUCTURAL
C    INPUT PARAMETERS TO BE SURE THAT THERE ARE NO FATAL
C    ERRORS... "PASS" IS THE LOGICAL VARIABLE THAT IS
C    THE FLAG IF IT PASSES THE TEST
C
        SUBROUTINE CONSIS(PASS)
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        COMMON /LOADING/CASES,NMCASE,NJCASE,MCASE,JCASE
        COMMON /COMBINE/NCOMB,COMB,ACTLIST,ACASES
        COMMON /RELEASE/MBREL,SREL,STALLY
        COMMON /ASSEMB/ BMAX,BASS
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        REAL MCASE(5,40,6),JCASE(5,40,3)
        INTEGER CASES,NMCASE(5),NJCASE(5)
        REAL COMB(5,5)
        INTEGER NCOMB,ACTLIST(10),ACASES
        INTEGER MBREL(40),SREL(40),STALLY
        INTEGER BMAX
        REAL BASS(120,120)
        LOGICAL PASS(10)
C
        DO 1 I=1,10
1        PASS(I)=.FALSE.
        PRINT*,'--- PERFORMING CONSISTANCY CHECK ---'
        PRINT*,'--- STRUCTURAL DATA:'
        PRINT 100,'NUMBER OF JOINTS               =>',NT
        PRINT 100,'NUMBER OF MEMBERS              =>',TALLY
        PRINT 100,'NUMBER OF LOAD CASES           =>',CASES
        PRINT 100,'NUMBER OF LOAD COMBINATIONS    =>',NCOMB
        PRINT 100,'NUMBER OF ACTIVE LOAD CASES    =>',ACASES
100     FORMAT(' ',A31,I3)
        PRINT*,' '
        IF(ACASES.LT.NCOMB) THEN
         PRINT*,'*** ERROR - ACTIVATE LOAD CASE ***'
         PRINT*,'    NUMBER OF ACTIVE LOAC CASES TOO SMALL'
         PRINT*,' PLEASE GO TO -LCASE- AND ACTIVATE CASES'
         RETURN
        END IF
        IF(ACASES.EQ.0) THEN
         PRINT*,'*** ERROR - NO ACTIVE LOAD CASES ***'
         PRINT*,' PLEASE GO TO -LCASE- AND ACTIVATE CASES'
         RETURN
        END IF
        PASS(1)=.TRUE.
        DO 10 I=1,TALLY
         IF(MT(I,6).LE.0) GOTO 11
         IF(MT(I,7).LE.0) GOTO 12
         IF(MT(I,8).LE.0) GOTO 12
10      CONTINUE
        GOTO 19
11      PRINT 101,'*** ERROR - IN CONSTANTS IN MEMBER ',I,' ***'
101     FORMAT(' ',A35,I3,A4)
        RETURN
12      PRINT 101,'*** ERROR - IN PROPERTY  IN MEMBER ',I,' ***'
        RETURN
19      PASS(3)=.TRUE.
        PASS(2)=.TRUE.
        DO 18 I=1,NT
         IF(SREL(I).NE.0) THEN
```

```
            PASS(5)=.TRUE.
            GOTO 17
          END IF
18      CONTINUE
        PRINT*,'*** ERROR -- NO SUPPORTS ***'
        RETURN
17      I=I
        DO 20 J=1,NT
         DO 21 I=1,TALLY
21        IF(MT(I,1).EQ.J.OR.MT(I,2).EQ.J) GOTO 20
          PRINT 102,'*** ERROR - JOINT ',J,' HAS NO MEMBERS ***'
102     FORMAT(' ',A18,I3,A19)
20      CONTINUE
        PASS(4)=.TRUE.
        RETURN
        END
```

```
C       THIS SUBROUTINE WILL MULTIPLY A 6X6 TO A 6X1 MATRIX
        SUBROUTINE MULT6X1(MAT,ARR,RESUL)
        REAL MAT(6,6),ARR(6),RESUL(6)
        DO 10 I=1,6
         SUM=0
         DO 11 J=1,6
          SUM=SUM+(MAT(I,J)*ARR(J))
11      CONTINUE
         RESUL(I)=SUM
10      CONTINUE
        RETURN
        END


C    THIS IS A SUBROUTINE TO CALCULATE THE SUPPORT REACTIONS
C    DUE TO DIFFERENT LOAD CASES
        SUBROUTINE RECASE(CASES)
        COMMON /GEOM/   MT,TALLY,NLOC,NT
        COMMON /FORC1/   SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
        COMMON /RELEASE/ MBREL,SREL,STALLY
        REAL MT(40,12),NLOC(40,2)
        INTEGER TALLY,NT
        REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
        REAL  FEMDIS(5,40,6)
        INTEGER MBREL(40),SREL(40),STALLY
        REAL X,Y,Z,XX,YY
        INTEGER CASES
        DO 10 I=1,NT
         IF(SREL(I).EQ.0) GOTO 10
         DO 20 J=1,CASES
         DO 30 K=1,TALLY
          IF(MT(K,1).EQ.I) THEN
           X=EMCASE(J,K,1)
           Y=EMCASE(J,K,2)
           Z=EMCASE(J,K,3)
           GOTO 31
          ELSE IF(MT(K,2).EQ.I) THEN
           X=EMCASE(J,K,4)
           Y=EMCASE(J,K,5)
           Z=EMCASE(J,K,6)
           GOTO 31
          END IF
         GOTO 30
31      L=MT(K,5)/12    ! NOW IN FEET
        C=(NLOC(MT(K,2),1)-NLOC(MT(K,1),1))/L
        S=(NLOC(MT(K,2),2)-NLOC(MT(K,1),2))/L
        SUPCASE(J,I,3)=SUPCASE(J,I,3)+Z
        XX=X*C-Y*S
        YY=Y*C+X*S
        SUPCASE(J,I,2)=SUPCASE(J,I,2)+YY
        SUPCASE(J,I,1)=SUPCASE(J,I,1)+XX
30      CONTINUE
20      CONTINUE
10      CONTINUE
        RETURN
        END
```

```
      SUBROUTINE SOLVE(NT,CASES)
C
      COMMON /FORC1/  SECTFORC,EMCASE,SUPCASE,ACT,FEMDIS
      COMMON /RELEASE/MBREL,SREL,STALLY
      COMMON /ASSEMB/ BMAX,BASS
      REAL SECTFORC(12,40,3,21),EMCASE(12,40,6),SUPCASE(10,40,3),ACT(10,120)
      REAL  FEMDIS(5,40,6)
      INTEGER MBREL(40),SREL(40),STALLY
      REAL BASS(120,120)
      INTEGER BMAX
      INTEGER NT,I,J,K,L,N,M,MBAND,NSIZE,REL,CASES
      NSIZE=NT*3
      MBAND=BMAX
      SIZE=0
      DO 40 I=1,NSIZE
       IF(BASS(I,1).GT.SIZE)  SIZE=BASS(I,1)
40    CONTINUE
      BIG=SIZE*1.E20
C   APPLY JOINT CONSTRAINTS
      DO 50 I=1,NT
        REL=SREL(I)
       IF(REL.EQ.0)GOTO 50
        IR=3*I-3
       IF(REL.LT.100)GOTO 60
        BASS(IR+1,1)=BIG
        DO 59 M=1,CASES
59       ACT(M,IR+1)=0
       REL=REL-100
60     IF(REL.LT.10)GOTO 70
        BASS(IR+2,1)=BIG
        DO 69 M=1,CASES
          ACT(M,IR+2)=0
69      CONTINUE
       REL=REL-10
70    IF(REL.LT.1) GOTO 50
       BASS(IR+3,1)=BIG
        DO 79 M=1,CASES
          ACT(M,IR+3)=0
79      CONTINUE
50    CONTINUE
C    HERES THE SOLUTION
      DO 790 N=1,NSIZE
       DO 780 L=2,MBAND
        IF(BASS(N,L).EQ.0)  GOTO 780
         I=N+L-1
         C=BASS(N,L)/BASS(N,1)
         J=0
         DO 750 K=L,MBAND
          J=J+1
           BASS(I,J)=BASS(I,J)-C*BASS(N,K)
750    CONTINUE
          BASS(N,L)=C
780      CONTINUE
790    CONTINUE
C
800   DO 830 N=1,NSIZE
       DO 820 L=2,MBAND
        IF(BASS(N,L).EQ.0)GOTO 820
        I=N+L-1
        DO 809 M=1,CASES
```

```
          ACT(M,I)=ACT(M,I)-BASS(N,L)*ACT(M,N)
809         CONTINUE
820       CONTINUE
          DO 830 M=1,CASES
            ACT(M,N)=ACT(M,N)/BASS(N,1)
829       CONTINUE
830       CONTINUE
C
          DO 860 M=2,NSIZE
           N=NSIZE+1-M
           DO 850 L=2,MBAND
           IF(BASS(N,L).EQ.0) GOTO 850
            K=N+L-1
            DO 849 I=1,CASES
             ACT(I,N)=ACT(I,N)-BASS(N,L)*ACT(I,K)
849         CONTINUE
850       CONTINUE
860       CONTINUE
C     ACT HOLDS THE SOLUTION
          RETURN
          END
```

REFERENCES

1. S. J. Fenves. "Future Directions of Structural Engineering Applications." Computers and Structures, 10, No. 1/2 (Jan. 1979), pp 3-6.

2. S. J. Fenves, R. D. Logcher and S. P. Mauch. Stress: A User's Manual. Cambridge, Mass. The M.I.T. Press, 1964.

3. E. P. Forster. "Structural Analysis on a Minicomputer." Engineering Software. Proceedings of the First International Conference. Edited by R.A. Adey. London, England: Pentech Press, 1979.

4. G. A. Hartley and D. J. Carson "Computer-Aided Structural Design of Buildings: State of a Canadian Industry. Engineering Software. Proceedings of the First International Conference. Edited by R.A. Adey. London, England: Pentech Press, 1979 pp 443-455.

5. E. Litton. Automatic Computational Techniques in Civil and Structural Engineering. London, England: Crosby Lockwood, 1972.

6. R. D. Logcher, B. B. Flachbart, E. J. Hall, M. C. Conner, R. A. Wells, Jr., and A. J. Ferrante. ICES STRUDL II, Structural Design Language, Engineering User's Manual. 1st Edition. Vol. 1. Cambridge, Mass. M.I.T. Press, 1968.

7. D. T. Pyle. "Guideline for the Successful Use of the Computer." The Paper Plane, 3, No. 5 (April 1980).

8. G. Rzevski. "On the Design of Engineering Software. Engineering Software. Proceedings of the First International Conference. Edited by R.A. Adey. London, England: Pentech Press, 1979.

9. D. A. Spencer. Computers and Programming Guide for Engineers. Indianapolis, Ind. Howard W. Sams and Co., Inc., 1973.

10. J. A. Swanson, G. J. DeSalvo. ANSYS Engineering Analysis Systems, User's Manual. Houston, PA: Swanson Analysis Systems, Inc., 1982.

11. J. A. Swanson. "Present Trends in Computerized Structural Analysis. Computers and Structures, 10, No. 1/2 (Jan. 1979), 33-37.

12. R. N. White. "Computer Graphics Will Do More Than Anything Else to Boost Design Productivity. Civil Engineering, (Oct. 1980).

13. E. L. Wilson, K. Bath, F. E. Peterson. A Structural Analysis Program For Static & Dynamic Response of Linear Systems, SAP IV User's Manual. National Technical Information Service, Report # EERC 73-11, June 1973.

14. E. L. Wilson. "The Use of Minicomputers in Structural Analysis". _Computers and Structures_, 12, No. 5 (Nov. 1980), 695-698.

15. A. S. Villandeva. "Computer Application in Structural Engineering". _Computers and Structures_, 9, No. 1 (Jan 1978), 65-68.