DEVELOPMENT OF A FINITE ELEMENT PROGRAM WITH A HIGHLY EFFICIENT BANDWIDTH REDUCER FOR SLUICE GATE ANALYSIS

BY DOUGLAS G. SCOTT

B.Sc.(M.E.), University of Manitoba 1990

A Thesis Submitted to the
Faculty of Graduate Studies in
Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Mechanical and Industrial Engineering
University of Manitoba
Winnipeg, Manitoba, Canada
March 24, 1993

DEVELOPMENT OF A FINITE ELEMENT PROGRAM WITH A HIGHLY EFFICIENT BANDWIDTH REDUCER FOR SLUICE GATE ANALYSIS

BY

DOUGLAS G. SCOTT

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

ABSTRACT

MHYFECS, an acronym for Manitoba HYdro Finite Element Computer Simulation, is a group of interactive computer programs that have been developed to model the loads, reactions, and stresses of a typical sluice gate. The programs have been developed to perform three distinct tasks. One, to better quantify the hydraulic forces acting on a typical sluice gate. Two, to examine the reaction forces and stresses resulting from these loads. Three, to design suitable reinforcements, if necessary, for any highly stressed regions.

Steady state flow model results were used to show that under the prescribed maximum closure rates, the total dynamic effects caused by the gate closure were always less than one percent of the total force acting on the gate. This justified modeling the sluice gate with a hydrostatic pressure as the maximum loading force to determine the roller misalignment angles caused by the deflection of the gate. These misalignments along with initial and journal bearing clearance misalignments create lateral forces which are of great interest. The initial design parameter for the lateral forces acting on older gates was less than thirty percent of the maximum normal force which is used for newer gates. The application of these lateral forces are the final modeling step to determine the structural integrity of the gates. The results of the application of the lateral forces show unacceptable stresses in the supporting structure of the bottom side rollers. To alleviate this problem, a reinforced design of the side roller supporting structure was modeled and shown to experience more acceptable stresses than the existing design. The side roller support integrity was the main premise of the work. The resulting finite element elastic stress program was developed to the level that it can be used for a quick and easy proof of concept design analysis. To make the PC based program more useful, an improved bandwidth reduction algorithm was developed. The new algorithm, tested on a wide range of test problems, outperformed the widely used GPS algorithm and produced some as of yet unfound reductions in bandwidth in the test problems.

It is recommended that Manitoba Hydro either reinforce the existing side roller support areas or design a roller path that includes constraining edges. The constraining edges, such as a heavy channel, would eliminate lateral forces and hence even the need for side rollers.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my advisor for his support, insight, and guidance in the preparation of this thesis, Dr. Ray P. S. Han. I would also like to thank the other members of my examining committee, Dr. John Shewchuk and Dr. John Glanville, for reading the thesis.

Financial support was provided by Manitoba Hydro. This assistance is gratefully acknowledged along with the continuous technical support provided by Ron Britner.

I would also like to extend my thanks to the other professors whose timely suggestions saved much time and effort in the preparation of this work. A special word of thanks to my parents for their never ending support and encouragement during my studies.

A word of thanks also goes to Dr. Gordon Everstine, at David W. Taylor Naval Ship R&D Center, Bethesda, Maryland, for providing his test set of 30 problems that came in very useful in the bandwidth reduction algorithm development.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 What is a Sluice Gate?

An integral part of any hydro generating station is the control of the reservoir water level. The reservoir water level, which must be maintained to ensure adequate head for the hydro generating turbines, is regulated by the opening and closing of sluice gates. These gates are large metal structures that are adjusted by cables or screw jacks. A picture of a typical sluice gate, with a person on it to indicate scale, is shown below in *Figure 1.1*.

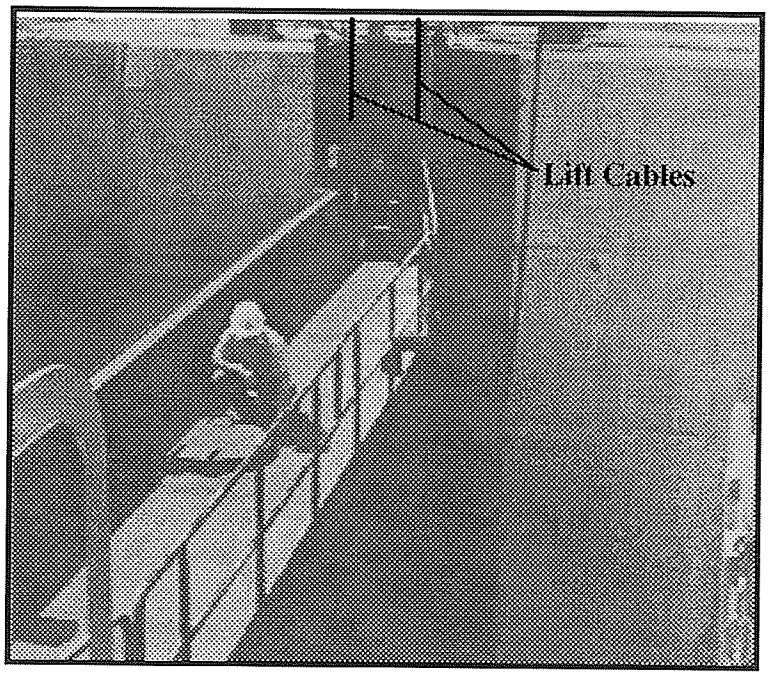


Figure 1.1 Typical Sluice Gate

We can see the relatively large size of these gates, about 45 feet wide by 50 feet high, and can immediately understand the large forces generated by the water on the upstream side of the gate. We can also see in *Figure 1.1* the cables typically used to lift a sluice gate. To facilitate motion, 2 sets of rollers are mounted along the edges of the gate as depicted in

Figure 1.2. The main rollers transfer the water forces from the gate to the embedded supports. Notice the spacing of the rollers decreases near the bottom of the gate. The main rollers are complemented with smaller side rollers used to ensure the gate tracks vertically. *Figure 1.2* shows the typical positions of the main and side rollers on a spillway gate.

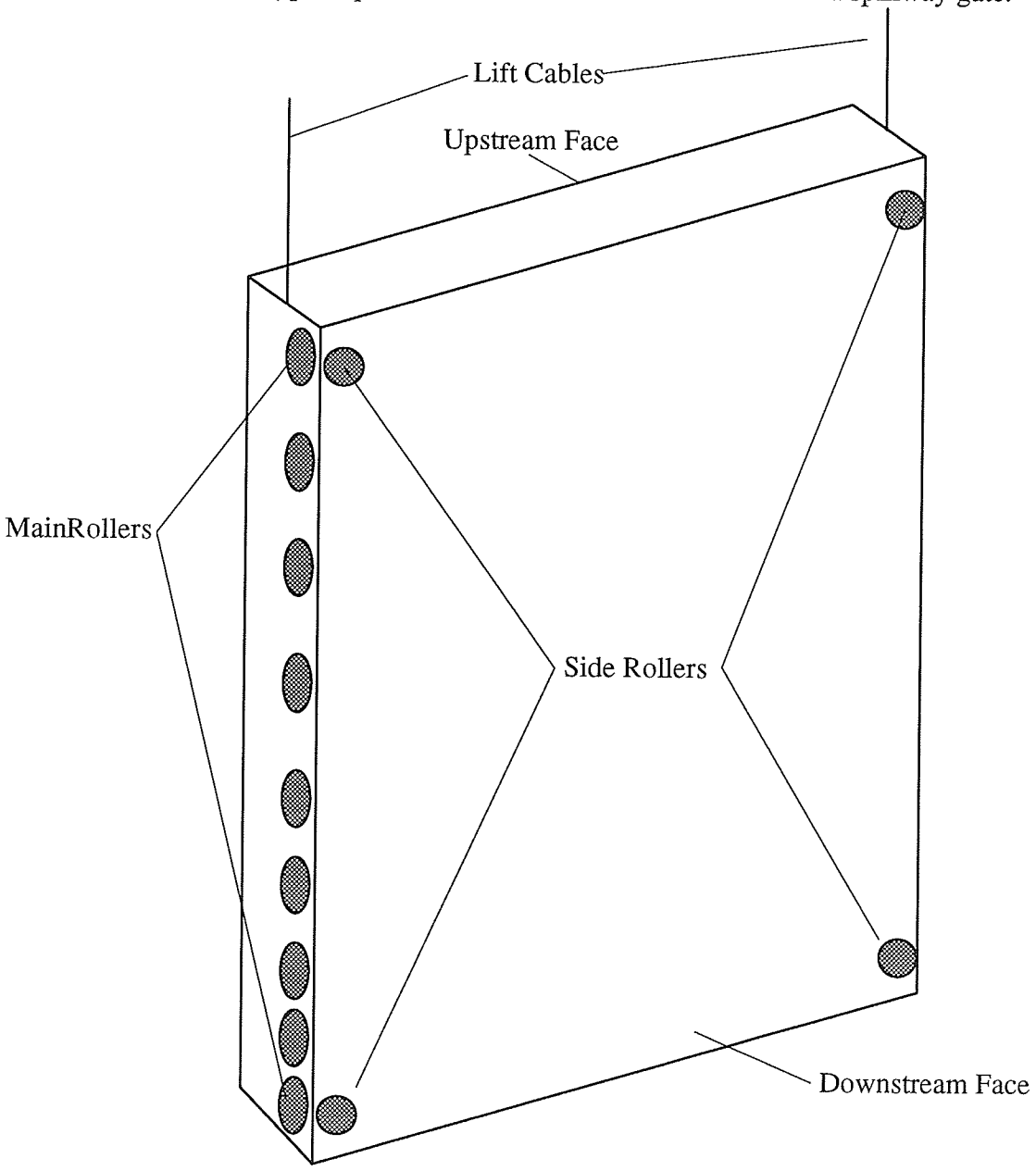


Figure 1.2 Rollers on a Sluice Gate

1.2 Need for Study

One problem in an existing sluice gate design is the control of its vertical motion. The two lifting cables are normally employed to raise or lower the gate, but if the main rollers do not track parallel to the vertical motion, the gate will not move in the desired manner. This method of lifting requires the use of the side rollers to guide the gates in their vertical motion. The side rollers are much smaller than the main rollers as the force thought to be acting on them is relatively small. It is this original assumption of only a small force acting on the side rollers that has caused the most concern in existing sluice gate installations.

Since 1968, all gates have been designed to withstand a side force of at least 30% that of the normal force applied from water pressure. Some specific concerns relate primarily to sluice gates designed and manufactured before 1968. The reason for these concerns is that the older gates may not have had such a large design factor to account for lateral loading. Whether or not a 30% figure is necessarily accurate is not yet known so the structural integrity of some older gates may be in doubt. To address these concerns, three specific questions must be answered and applied to the older gate designs.

1. What is the maximum normal force acting on a sluice gate? The normal force created by the water on the upstream face of the gate must be accurately quantified so that the reactions of the main rollers, and then the side rollers, can be determined. Presently, the maximum dynamic force acting on the gate during closure is not known, and neither is the question of how the magnitude of the dynamic forces compares with that of the static forces.

2. How does the deflection of the gate affect the misalignment of the main roller axles and hence, the side roller load? The main rollers, if misaligned, will cause the gate to want to track at an angle off vertical. The side rollers will want to resist the sideways motion by creating a reaction force equal to the main roller lateral force. The factors affecting the magnitude of the main roller lateral force are the normal force, the angle of misalignment, and the coefficient of friction. The 30% figure now used is a rough estimate of the coefficient of friction of 0.3.

3. Once a reasonable estimate of the acting lateral force is found, what is the resulting stress distribution in the existing gate structure? It should be noted here that onsite inspections of some gates have found localized permanent deformations around some side roller supports. This would indicate that indeed, some older gates were under-designed for the actual side roller force. If the estimated side roller forces are shown to cause comparable results in an analysis of the side roller support area, then a reinforcement design must be done to strengthen appropriate areas of the gate.

1.3 Scope of Investigation

The scope of this thesis will entail the review of the three basic questions outlined above and make recommendations based on those findings. The study will concentrate on the Kelsey generating station spillway gates since this station has been the focus of a past investigation and as a result there is considerable existing information that can be used for comparative purposes. The bulk of the work revolves around the development of a group of user-friendly finite element computer programs that will provide Hydro the tools not only to investigate the Kelsey gates, but other gates at other stations as well.

At present, the peak forces acting on the gate are unknown. To better quantify the water pressures acting on the gate, a fluid model was developed in Chapter 2 to find the relationship between the hydrostatic pressures and the dynamics effects of gate closure.

To calculate the main roller reaction forces, determine the effects of roller misalignments, and find side roller reactions on the gate, an elastic solid finite element model program was developed in Chapter 3. This program emphasizes a graphical user interface for the development of structural models and the display of results. To make this program more useful, a wide variety of both two and three dimensional elements were developed. Also, to allow the program to be used on a PC, an improved bandwidth minimization algorithm was developed in Chapter 4 to save memory, computational time, and computational effort.

CHAPTER 2

FLUID LOAD MODELING

2.1 Literature Survey

A literature survey revealed a great many papers dealing with the topic of numerical modeling of fluid flow. Due to the abundance of literature on the subject, the scope of the search was narrowed to the specific topic area of potential flow modeling by the finite element method. This topic area was considered appropriate only after careful consideration of the physical problem to be modeled. Two journal papers with specific use of the potential flow finite element method applied to relevant flow situations were chosen. The first paper is by Chan, Larock, and Hermann[1] and the other is by Diersch, Schirmer, and Busch[2].

The first paper presents the problem of solving a free surface flow with an unknown final surface geometry. The authors' solution method sets out specific criteria that a solution for this type of problem must satisfy. These criteria dictate that the method should handle free surfaces with gravitational effects and be adaptable to complex geometries. The example problems in the papers, as well as in this study, all involve flow situations in which an ideal fluid is assumed valid. The use of an ideal fluid means the existence of the velocity potential, ϕ , and hence the variational principle to describe the flow field. The element derived in this paper is a 6-noded triangle that lends itself to descretize irregular geometric boundaries readily. To handle the free surface flows, two specific boundary conditions must exist on that surface. The conditions for free surface flow are that the normal flow across the surface must be zero and the pressure along the surface must be constant. This constant pressure surface will require the specification of velocity

potentials at all nodal points on the free surface. Since the fluid is assumed ideal, the Bernoulli equation can be used to calculate the difference in speed between any two points, and hence their potential, as shown by the derivations in the original paper and later in the next section. The use of the Bernoulli equation sets the constant surface pressure and takes care of gravitational effects, but the problem of the ensuring zero normal velocity still exists. This is taken care of by iteratively moving the surface nodes into a streamline representing the surface of the flow. The results of this first paper give a flexible method that solves free surface flows with velocity and pressure fields for the entire modeled area.

The second paper addresses all of the problems of the first paper as well as considering the problem of an initially unknown flow rate. In the first paper, flow rates were fixed, but in the second paper, the objective is not only to describe the flow, but also to find the unknown flow rate. By simply relating the inlet and outlet flows through the Bernoulli equation and the continuity condition, the flow rate can be iteratively found. The existing difference in flow depth and free surface height gives the flow rate, and after a solution for the new free surface flows, the values are recalculated. The calculations are repeated until convergence occurs. These derivations are only briefly shown in the next section but in much more detail in the original paper.

The flow problem to be solved is that of water flowing under the sluice gate during gate closure. The first of two major assumptions was that the speed of gate closure is sufficiently slow to result in no noticeable hydro dynamic effects on the pressure distributions in the flow and the second is that an ideal fluid model would provide acceptable results. The assumption of no noticeable hydro dynamic effects is justified by employing the solutions of several pseudo-steady-state solutions. Since the flow field is known at each point in time, the hydro dynamic forces necessary to affect such a flow

were estimated. If the dynamic forces caused by the closure of the gate are found to be in the range of one to two orders of magnitude lower than static forces, the assumption will be valid. Also, the ideal flow assumption are checked though comparison of calculated discharge rates versus Manitoba Hydro's own discharge surveys

2.2 Derivation of the Potential Flow Element

The element used in this study is a 6-noded triangle for two reasons, it is simple to derive and can be used to model unusually shaped geometry. The variational principle describing two-dimensional potential flow is:

$$I(\phi) = \iint_A \frac{\rho}{2}\left[(\phi_{,x})^2 + (\phi_{,y})^2\right] dxdy \; - \rho \int_c \phi(\phi_{,n})_c \, ds \qquad (1)$$

where $I(\phi)$, ϕ, $\phi_{,x}$, ρ, A, and c are the functional of velocity potential ϕ, the velocity potential, the x velocity component, the fluid density, the area of entire flow region, and the boundaries with specified normal velocities respectively. Derivatives of ϕ, say with respect to n, are denoted by $\phi_{,n}$.

The potential flow theory assumes an irrotational flow of homogeneous, inviscid, and incompressible fluid. This is a valid assumption for large bodies of water as is the case in this study. The variational approach requires the first variation of Equation (1) be set to zero. This yields the following equations:

$$\phi_{,xx} + \phi_{,yy} = 0 \qquad (2)$$

$$V_x = \phi_{,x} \text{ and } V_y = \phi_{,y} \qquad (3)$$

To discretize the continuous field problem, we must use a technique to reduce the problem to a descrete system. This is accomplished by employing shape functions with

the natural triangular coordinates. The natural coordinates used for the six noded triangular element are shown in *Figure 2.1*.

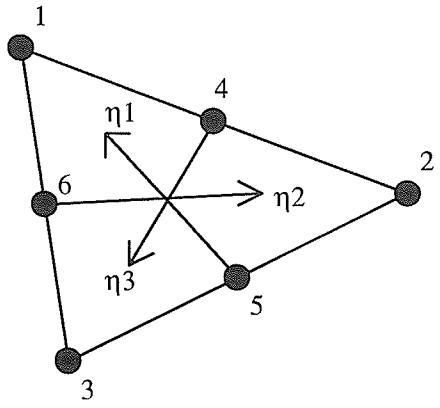


Figure 2.1 Potential Flow Element Nodes and Shape Functions

From this figure, the shape functions can be derived as follows:

$$\gamma_1 = \eta_1(2\eta_1 - 1) \qquad (4a)$$
$$\gamma_2 = \eta_2(2\eta_2 - 1) \qquad (4b)$$
$$\gamma_3 = \eta_3(2\eta_3 - 1) \qquad (4c)$$
$$\gamma_4 = 4\eta_1\eta_2 \qquad (4d)$$
$$\gamma_5 = 4\eta_2\eta_3 \qquad (4e)$$
$$\gamma_6 = 4\eta_3\eta_1 \qquad (4f)$$

This allows the calculation of the value at any point in the element through the use of *Equations (5)* and *(6)*

where $$\phi = \phi_i\gamma_i \ (i = 1,6) \qquad (5)$$

ϕ_i is the nodal value of ϕ

in which $$I(\phi) = \sum_{e=1}^{N} I^e(\phi) \qquad (6)$$

I^e is the elemental functional of $I(\phi)$

where N is the number of elements. The components of *Equation (6)* are in the local coordinates of the element. Since *Equation (1)* is in global coordinates, a tranformation method must be applied. One such transformation method is a Lagrange transformation matrix $\langle\chi\rangle$. It is simply the shape function in a vector for:

$$\langle\chi\rangle^T = \langle \eta_1(2\eta_1 - 1),\ \eta_2(2\eta_2 - 1),\ \eta_3(2\eta_3 - 1),\ 4\eta_1\eta_2,\ 4\eta_2\eta_3,\ 4\eta_3\eta_1,\ \rangle \qquad (7)$$

$$\phi = \langle\chi\rangle^T \{\phi\} \qquad (8)$$

This now allows the components of the *Equation (1)* to be found through the following equations:

$$\begin{Bmatrix} V_x \\ V_y \end{Bmatrix} = \begin{Bmatrix} \phi_{,x} \\ \phi_{,y} \end{Bmatrix} = [D_{xy}]\{\phi\} \qquad (9)$$

in which

$$[D_{xy}] = [L_R][D_L] \qquad (10)$$

where

$$[L_R] = \frac{1}{2A}\begin{bmatrix} b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \qquad (11)$$

where

$$b_i = y_j - y_k\ ;\ c_i = x_k - x_j\ ;\ 2A = b_j c_k - b_k c_j \qquad (12)$$
$$for\ i = (1,2,3)\ j = (2,3,1)\ k = (3,1,2)$$

and

$$[D_L] = \begin{bmatrix} 4\eta_1 - 1 & 0 & 0 & 4\eta_2 & 0 & 4\eta_3 \\ 0 & 4\eta_2 - 1 & 0 & 4\eta_1 & 4\eta_3 & 0 \\ 0 & 0 & 4\eta_3 - 1 & 0 & 4\eta_2 & 4\eta_1 \end{bmatrix} \qquad (13)$$

The x and y values used in *Equation (12)* can be seen in *Figure 2.2*.

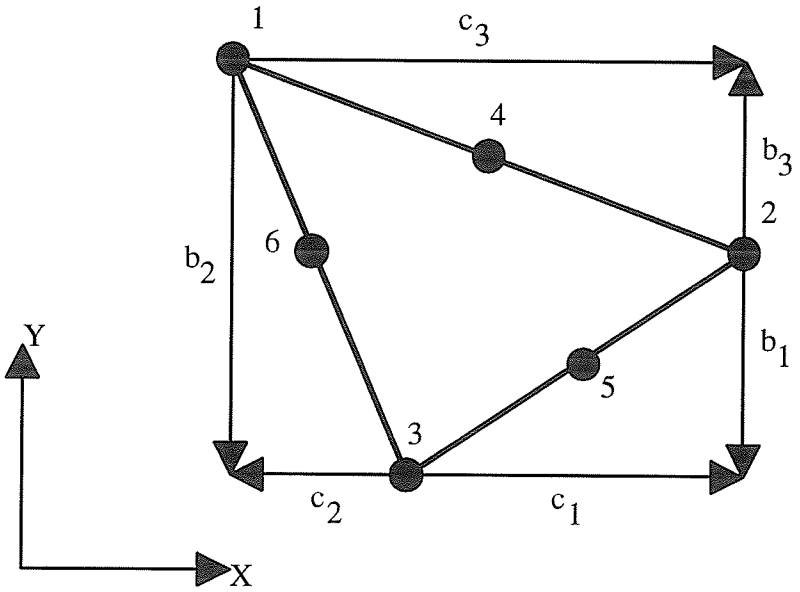


Figure 2.2 A 6-Node Triangular Element

Substituting *Equations (8)* and *(9)* into *Equation (1)* and taking the variation to minimize the function yields,

$$\frac{\partial I(\phi)}{\partial \phi_i} = S_{ij}\phi_j - SL_i \tag{14}$$

This is analogous to structure mechanics in that *[S]* is the element stiffness matrix and *{SL}* is the load vector. Since the variation is to be minimized, the result can be put into the more standard form:

$$[S]\{\phi\} = \{SL\} \tag{15}$$

The components of the *[S]* matrix and *{SL}* vector are described in the appendix in more detail. The calculation of the potential values on the free surface uses Bernoulli's Equation as shown in *Equation (16).*

$$\tfrac{1}{2}q_i^2 + \frac{p_i}{\rho} + gy_i = \tfrac{1}{2}q_d^2 + \frac{p_d}{\rho} + gy_d \tag{16}$$

where i is any downstream point on the free surface and d is the downstream reference point on the free surface. Since the pressures p are constant and the height y are known, the speed q_i can be found at all the free surface points. Assuming the velocity varies linearly between any two adjacent nodes, the potential at all the free surface points can be found from:

$$\phi_j = \phi_i - \frac{(q_i + q_j)}{2}\Delta s \qquad (17)$$

where Δs is the surface distance between the two nodes and ϕ_i is the known potential. This solves the constant pressure requirement on the free surface but, if the initial guess of the free surface position is incorrect, the resulting velocity vectors from *Equation (3)* have a non-zero normal component. To correct this problem, the nodes on the free surface are continually moved into the position dictated by the direction of the upstream velocity vector by solving the problem iteratively. If only the surface nodes were moved, the problem of an unacceptable element shape could arise. To alleviate this concern, a flexible network of nodes is used. That means that if a free surface node is moved up, then the nodes below that free surface node would move proportionally up relative to their distance between the surface and the fixed boundary of the flow. This is shown in *Figure 2.3* in which a flexible network of four nodes moves upwards. Notice how the top node moves the most and that the other nodes movement decreases linearly towards zero movement at the fixed boundary.

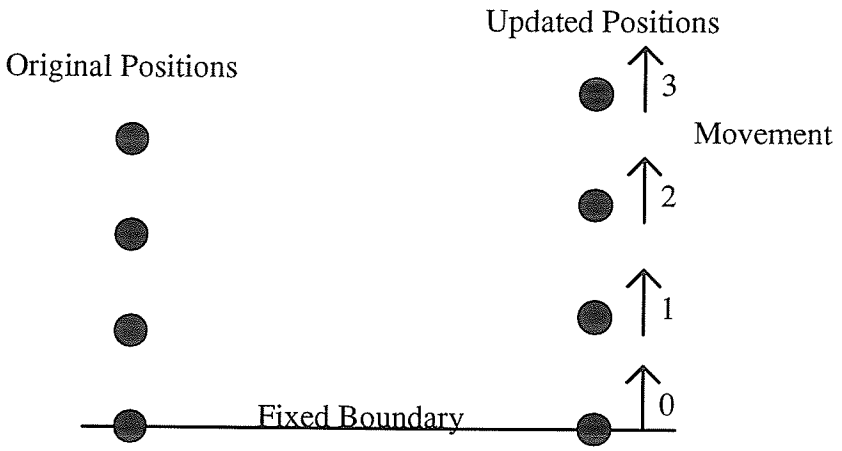


Figure 2.3 Flexible Network of Nodes

The last problem in the flow model is the unknown flow rate. This is solved from the combined use of the Bernoulli Equation, *(18)*, and the Continuity Equation, *(19)*.

$$\frac{q_o^2}{2g} + z_o = \frac{q_d^2}{2g} + z_d \qquad (18)$$

$$h_o q_o = h_d q_d \qquad (19)$$

where q_o is the approach speed, q_d is the downstream or exit speed, z is the elevation, and h is the depth of the flow at the approach and exit. Substituting *Equation (19)* into *Equation (18)* yields q_o:

$$q_o = \sqrt{\frac{2g(z_o - z_d)}{\left(\frac{h_o}{h_d}\right)^2 - 1}} \qquad (20)$$

The entrance flow is calculated and prescribed at each iteration of the solution until the changes in the flow rate and the free surface position are less than the convergence criteria value.

2.3 Potential Flow Program

The complete MHYFECS consists of two modules, the fluid modeling module, and the structural modeling module. The fluid modeling module is a potential flow finite element program described in this chapter. The structural modeling module will be presented in the next chapter.

2.3.1 Graphic Input Interface

To facilitate the development of flow models, a user-friendly graphical interface was developed to avoid the tedious editing of data files. All programming in this study was performed in FORTRAN using the GSS-GKS graphics library. This graphical interface allows the user not only to visually check the model but also modify all the element and node data without actually seeing the data file. The program consist of four menus of options. The first is the FILE menu which includes the options to OPEN, CHECK, and SAVE the data file. The CHECK function was added to insure the correctness of data files before the solution phase. The NODE menu contains the CREATE, DELETE and MOVE features necessary for manipulating the nodal data. The ELEMENT menu contains two element creation functions and a DELETE function to build and modify the model. The last menu is the VIEW menu. This menu allows the user to view the model with any center of view point and zoom factor. The graphical interface is shown in *Figure 2.4* where the VIEW menu containing an example data file is depicted.

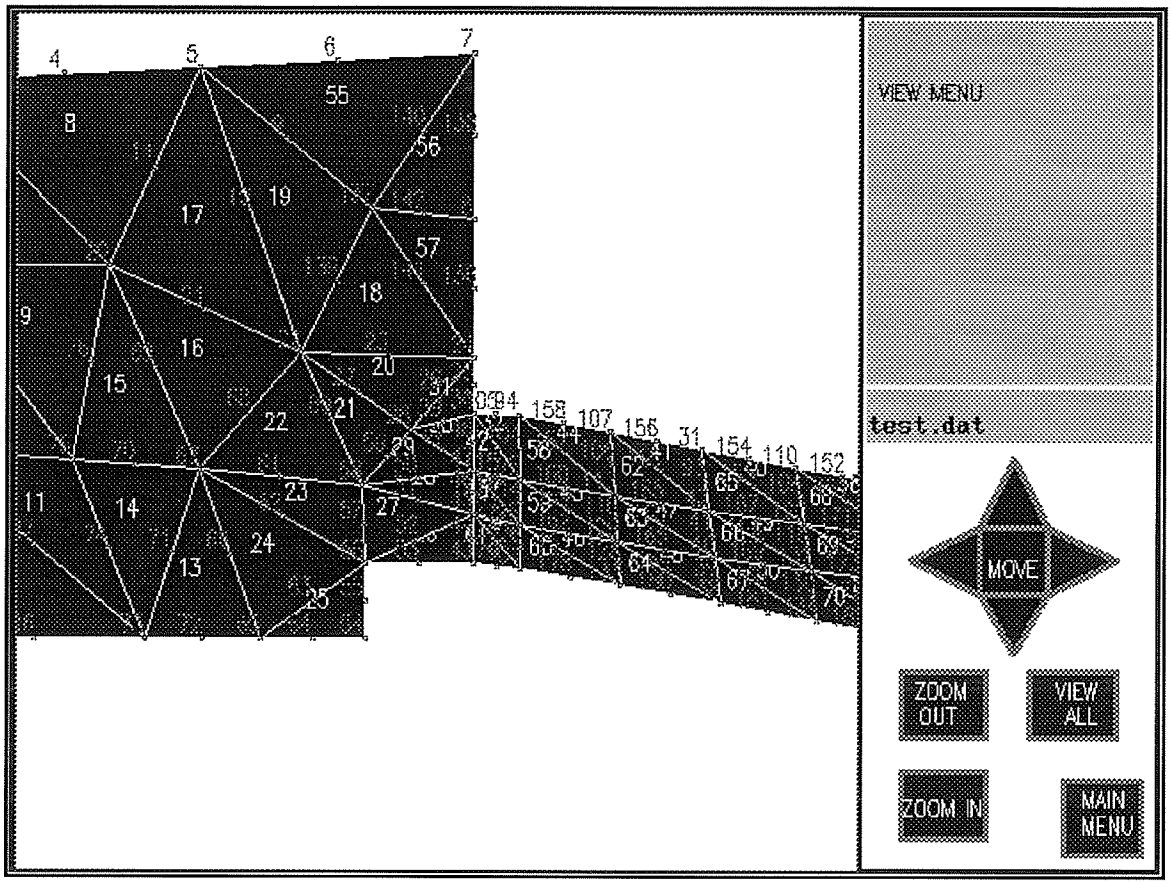


Figure 2.4 Graphical Input Interface for VIEW Menu

2.3.2 Solution Program

The solution program, as the name implies, contains not only the solution algorithm, but also includes some graphics. The programs' graphic abilities visually demonstrate the approach to convergence, and once converged show the velocity vectors and pressure contours. Each elemental matrix is calculated in the main solver of the fluid program and all the elemental matrices are assembled together to produce the global matrix, S. The load vector SL is then created from the specified velocities on the boundaries. The last step before solving the global system of equations is the setting of the potential value

conditions. The potential values are calculated as described earlier and applied to the matrix using the penalty method. The well-known penalty method works as follows. If a solution value for a specific node is to assume a value of say, 45, then the stiffness matrix diagonal for that node is set to a very high values, say 10^{20}, and the force vector is $45*10^{20}$. This forces the solution value for that node to be 45 or at least infinitesimally close to 45. The matrix vector equation, once assembled, is solved using Gauss-Jordan elimination to solve for the potential values. The program iteratively loops through the calculations until the change in the free surface is within the convergence criteria and then displays the final velocity and pressure fields. The operational flow chart of the program is shown in *Figure 2.4*. The program listing can be found in the appendix. Examples of a solved flow are shown in *Figures 2.6* and *2.7* which correspond respectively to a velocity vector plot and a pressure contour plot. The geometry depicted in *Figures 2.6* and *2.7* is that of a typical sluice way. Notice the gate rests on a small step at the bottom of the channel. Also notice how the flow model captures the rising water level in front of the gate as would be in reality, and that the magnitude of the velocity vectors is proportional to the length of the vectors. To facilitate printing of he results, the graphic output of the solved model can be saved in graphic metafiles.

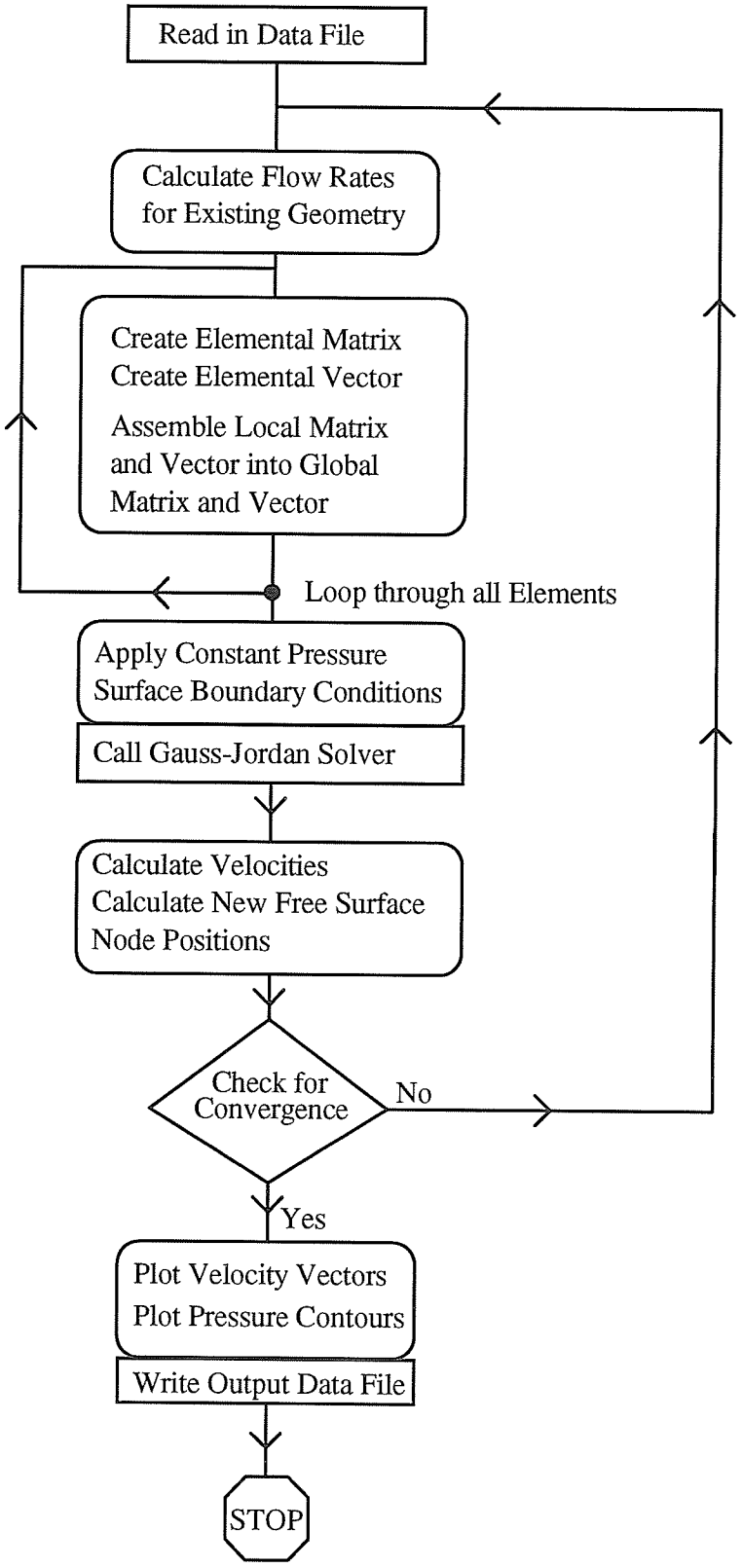


Figure 2.5 Solver Flow Chart

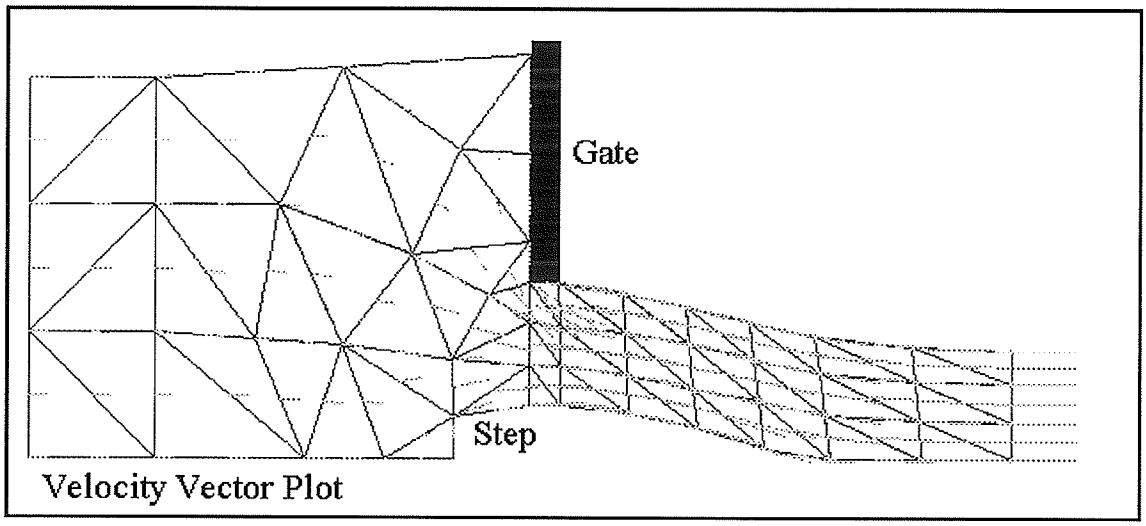


Figure 2.6 Output from Flow Module: Velocity Vector Example Plot

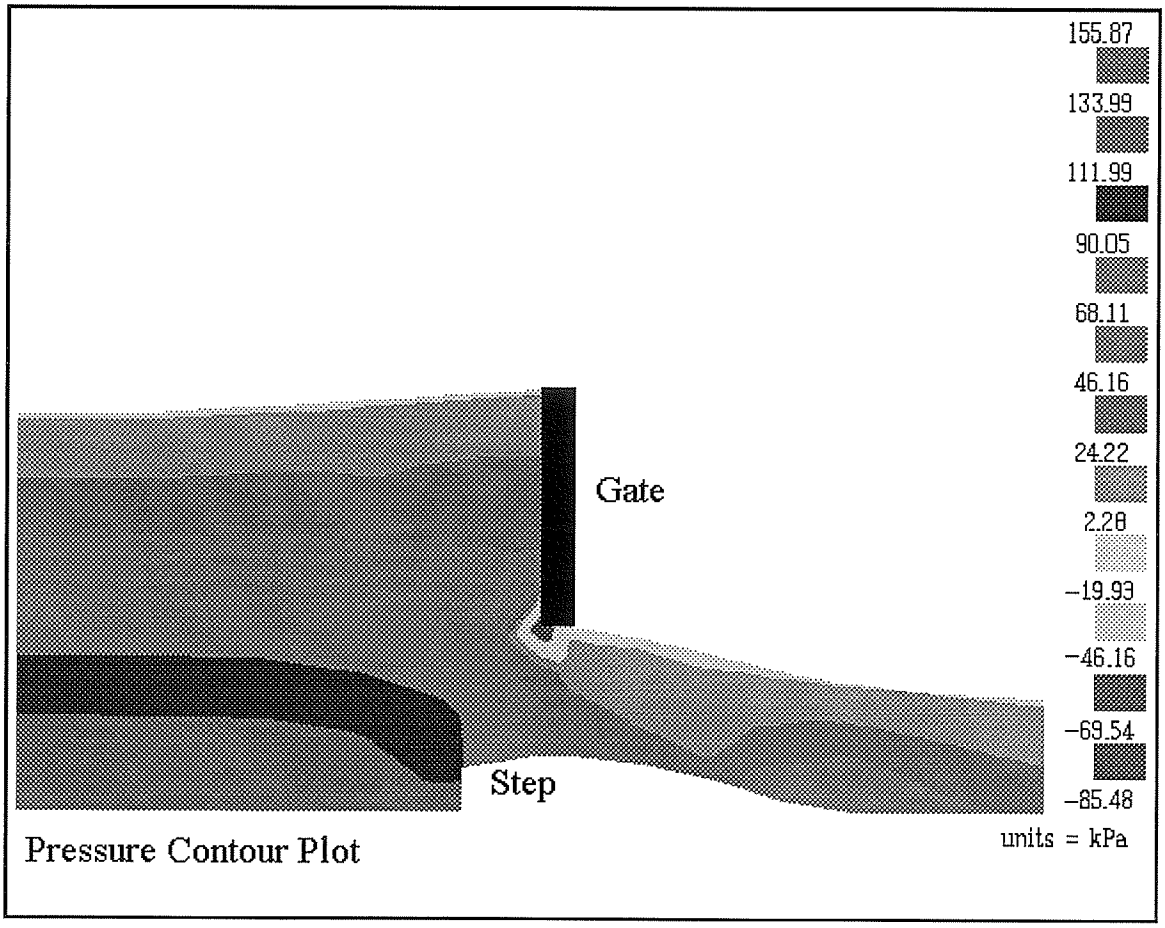


Figure 2.7 Output from Flow Module: Pressure Contour Example Plot

2.4 Results of Fluid Modeling

To verify the fluid modeling module and to assess its accuracy, comparison of flow rates predicted numerically with measured field data is next. This section also includes a discussion on the validity of the assumption of neglecting dynamic effects in computing the fluid pressure.

2.4.1 Comparison of Flow Rates

The sluice gate at Kelsey Generating Station normally operates with openings of twenty feet or less. Due to this operational procedure, the modeling completed in this study is the gate closing from an opening of 20 feet to a complete closure. The models were solved with the forebay elevation of 605 feet relative to sea level, the highest operational depth of the forebay. A demonstration of a sluice gate opening and closing showed that closing times are in the order of at least 5 to 10 minutes. The actual design speed for gate closure as set out by the designers, Canadian Vickers Limited, is approximately 2.5 feet per minute. This speed means that closure from a 20 foot opening takes about 8 minutes which is expected to be too slow to create any significant dynamic loads. A more detailed discussion will be presented in the next section.

To ease calculations and avoid errors, the metric system of units was used in the flow model computations. The flow was modeled with the gate in positions from an opening of 7 meters (20 feet) to closure in 1 meter increments. The flow rates for each opening were found to be relatively close to those found in Manitoba Hydro's 1971 *Report on Discharges* as well as the 1963 *Spillway Rating Curves*. The results are in *Figure 2.8*

which shows the two Hydro flow measurements versus the prediction of this study. It was expected that due to the use of an ideal fluid model, this study's predicted flow rates would be slightly higher than the actual flows. Note that the predicted flow rates are no more than 20% higher than the two Hydro sources. The slightly higher flow rates will cause a slightly higher estimate of gate forces which will result in a more conservative design. Observe that at small openings near the bottom of the gate closure, the numerical model is quite accurate. This accuracy is particularly important since the forces are maximum near closure.

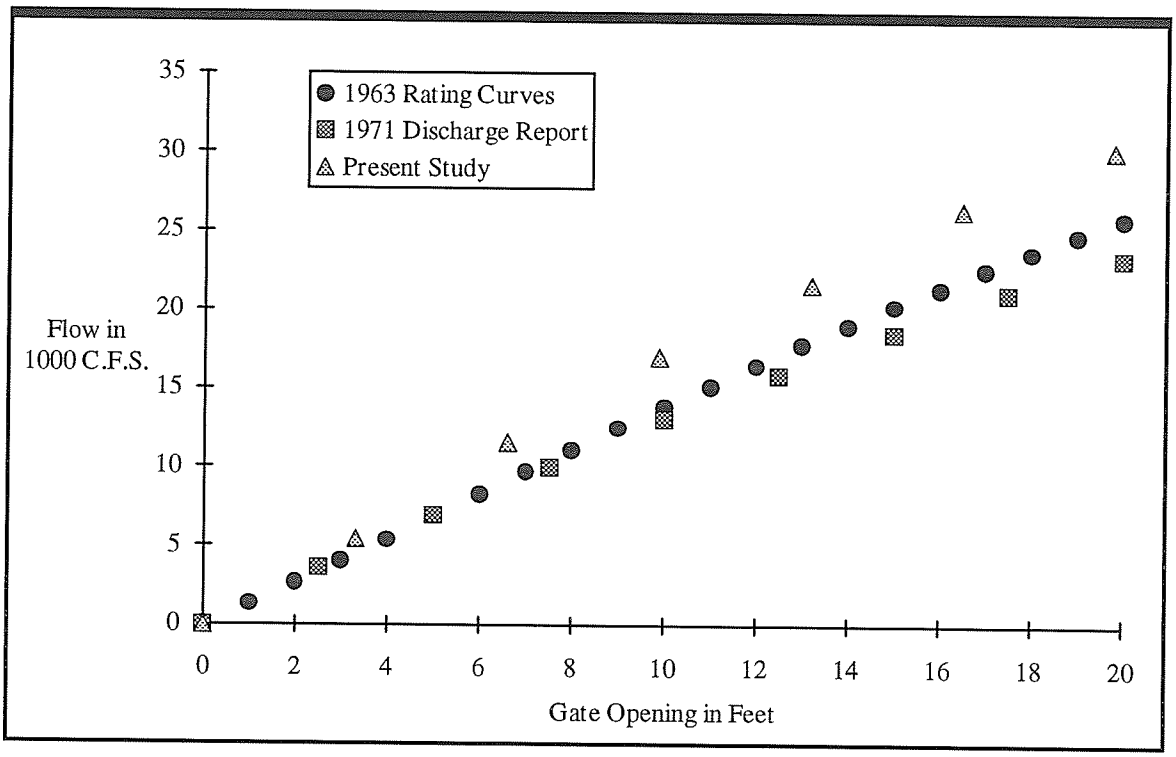


Figure 2.8 Comparison of Flow Rates

2.4.2 Dynamic Effects

The solutions of the flow models are for a series of steady state flows. To justify using a series of steady state flows to model a time dependent flow, the dynamic effects of the fluid flow must be shown to be negligible. To do this, the momentum equation for a control volume can be used to calculate the dynamic effects. The momentum equation in its standard form can be written as:

$$F = \frac{\partial}{\partial t} \int_{CV} V \rho dV + \int_{CS} V \rho V \, dA \qquad (21)$$

is where F is the reaction force, ρ the fluid density, V is velocity, CV is the control volume, and CS is the control surface. In a steady state flow the first part, the part with partial derivative with respect to time, is zero and the remainder of the reaction force is due to the constant momentum flux through the control surface. The first part of the equation is the dynamic part used to estimate the possible dynamic forces acting on the gate. Since the x axis forces are those of interest, *Equation 21* can be reduced to its x components:

$$F_x = \frac{\partial}{\partial t} \int_{CV} u \rho dV + \int_{CS} u \rho V \, dA \qquad (22)$$

where u is the x velocity component. The control volume used is the region in front of the gate contained by the concrete superstructure out to the forebay. The inlet velocity in the forebay is taken to be approximately zero and so is ignored. Thus, the u velocity is estimated to be, on average, the same as the entrance velocity since the entrance area is the whole upstream cross section. For each time interval between static gate positions, the u velocity is estimated to be the linear interpolation of the before and after speeds.

For example, for the forth time interval of approximately 68.6 seconds, the initial speed is 3.6 m/s and the end speed is 2.8 m/s. This would produce an estimation of u as:

$$u_t = 3.6 - 0.01167 \cdot t \qquad (23)$$

Using the above assumptions, the estimate of a dynamic force caused by the fluid deceleration becomes:

$$F_x = \frac{\partial}{\partial t}[(3.6 - 0.01167 \cdot t) \cdot 998 \cdot CV] \qquad (24a)$$

$$F_x = (0.01167) \cdot 998 \cdot 3175 = 37kN \qquad (24b)$$

This dynamic force of 37 kN is quite small compared to the maximum hydrostatic force of the water which is 12000 kN. The results for each time step are shown in *Table 1* and reveal that the dynamic portion of the flow never rise above 0.5% of the static pressure force.

Table 1 Estimated Gate Forces

Gate Positions (m)	Dynamic Force (kN)	Static Force (kN)	% Dyn./Stat
7-6	28	5751	0.48
6-5	30.5	7100	0.44
5-4	34	8591	0.34
4-3	37	10224	0.31
3-2	41.5	11999	0.30
2-1	46	13916	0.30
1-0	41.5	15975	0.26

To see how small and insignificant the estimated dynamic forces are, *Figure 2.9* shows their relative magnitudes. It is important to note that the dynamic forces can only be ignored for the slow closing situation assumed in this study. For higher closure speeds or a different control gate, the dynamic forces may be significant.

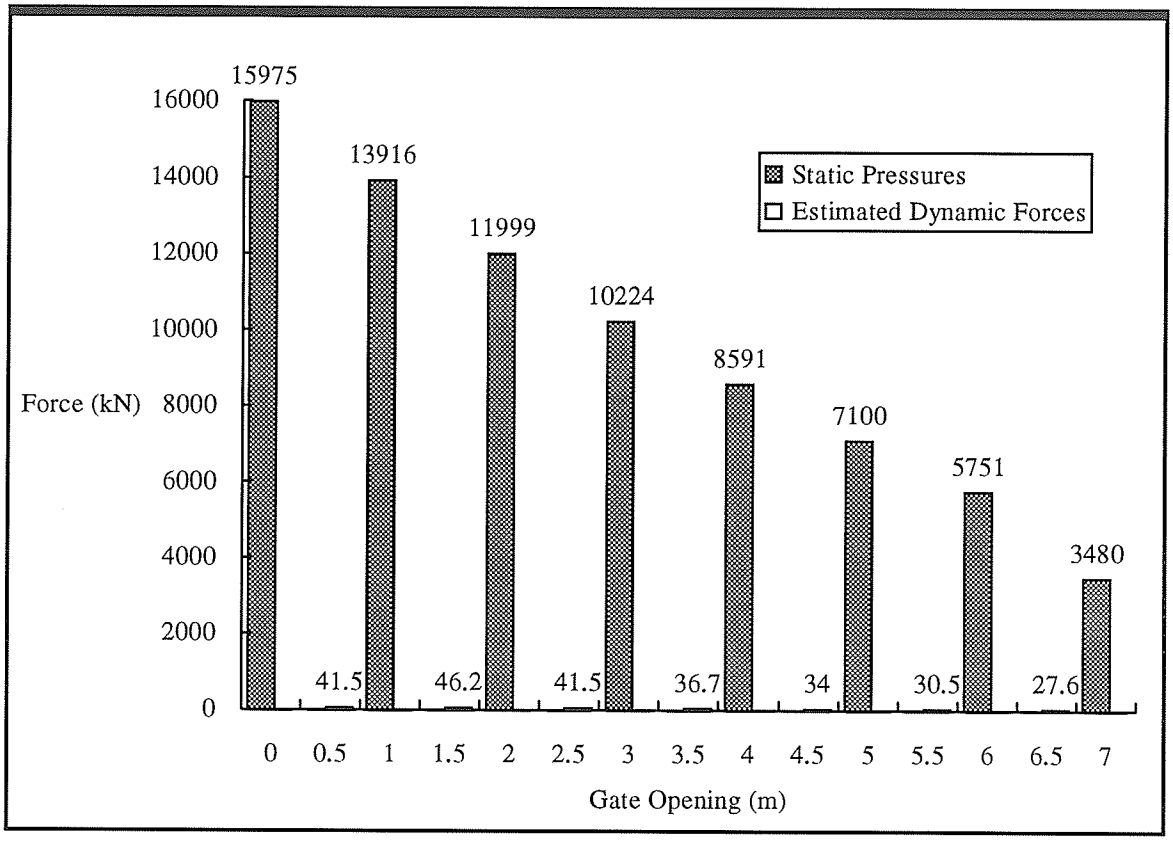


Figure 2.9 Estimated Gate Forces from Flow Model

2.5 Conclusions of Fluid Modeling

Two dimensional irrotational gravity flows of ideal fluids is studied with the use of finite elements. The computer code described herein allows the generation and analysis of flow models having unknown flow rates, complex geometry, and unknown free surface locations. Useful physical properties, such as velocities and pressures, can be computed

for the entire flow region. The development of a graphical input interface allows the user to visually modify and check the model prior to solving. The solution phase shows the user the progress of the solution and the final results in a graphical form.

The flow rates for the various gate positions agree quite closely with Manitoba Hydro's estimates and surveys. The resulting dynamic loads caused by the maximum rate of closure for normal operation are always less than half of a percent of the total pressure force on the gate and hence can be neglected. The maximum loading acting on the sluice gate is the hydrostatic pressure when the gate is fully closed with the reservoir at maximum forebay elevation.

CHAPTER 3

STRUCTURAL MODELING

3.1 Structural Modeling Programs

The structural modeling has been accomplished with the use of an elastic finite element package consisting of three independent programs. The first program is an interactive graphical pre-processor used to develop the structural models. The second program is the solution or solver program and the third an interactive graphical post-processor. This program group has been named MHYFECS (pronounced "miffecs"), an acronym for Manitoba HYdro Finite Element Computer Simulation.

3.1.1 Program Requirements

The initial scope of this study required the calculation of reaction forces, roller misalignments, and stresses in the solid modeling phase. The requirements immediately dictated the use of the well-known finite element method as the resulting complexities are most easily accounted for by this method. As the project proceeded, many structure design related questions were raised, and the need for a finite element method program that was more flexible than just a gate model became apparent. A discussion of the finite element method will not be given here as it can be found in many standard books and texts dedicated solely to this topic [3-7].

3.1.2 The PRE-PROCESSOR Program

The first program is the PRE-PROCESSOR program that is used to build the solid model. A great deal of time and effort were devoted to this program as the development of large models in the text data file is not only long and tedious, but also prone to errors. The PRE-PROCESSOR program's graphic interface was developed to also allow the user to visually check the model, to insure that the model is indeed coded. Also, the PRE-PROCESSOR program allows the user to develop and change the finite element model without actually seeing the data file. The features and use of the PRE-PROCESSOR program are described in detail in the users manual in the *Appendix*. *Figure 3.1* depicts an example of the PRE-PROCESSOR graphic interface showing a model in

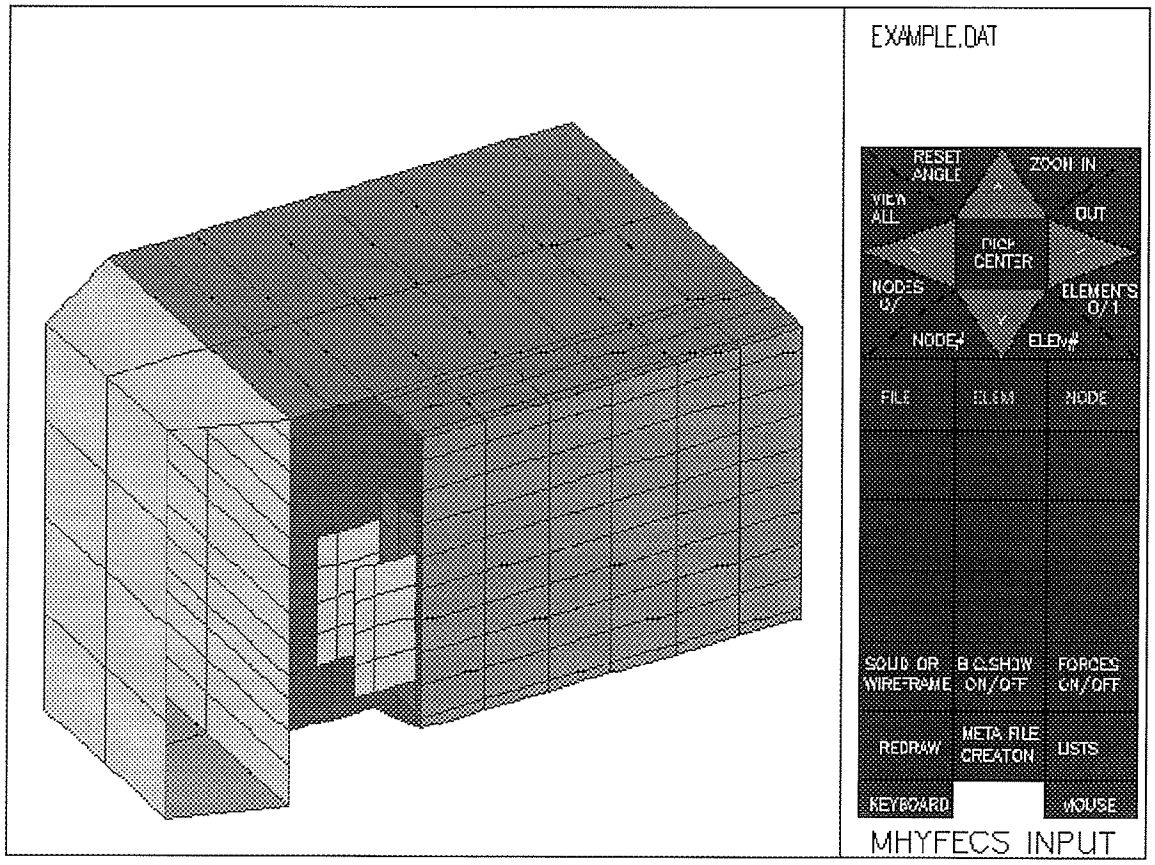


Figure 3.1 Pre-Processor Example in MHYFECS

development. The window on the left is the on screen display of the model being worked on. The function menu which allows mouse driven use of the program is on the right.

3.1.3 The SOLVER Program

The SOLVER program is the program that generates the elemental stiffnesses, assembles the global stiffness matrix, generates the load vector, solves the resulting system of equations, calculates the resulting stresses, and prints the results to an output file. It is modular in design so as to ease future program development and employs the use of a vector scheme for efficient storage. The listing of the SOLVER program can be seen in the *Appendix*. The SOLVER program contains no graphics as it is written to be portable between the PC and UNIX hardware. The program asks for an input data file but creates the output file automatically to which the solution is written. If any errors occur, command line messages are generated to describe the problem so that appropriate action can be taken. The program also continuously updates the user on the status of the solution progress with respect to the assembly progress, the matrix solution progress, and the stress calculations progress. A more detailed description of the SOLVER program is given in the *Appendix*.

3.1.4 The POST-PROCESSOR Program

The last program in the solid modeling group is the POST-PROCESSOR program. This program is a graphical interface based program designed to show the displacements and

stresses of the solved model graphically. The program reads in the output from the SOLVER and allows the user to view any one of the 6 primary stresses or the Von Mises equivalent stress in an undeformed or deformed configurations from any orientation in space. The data can also be reviewed in this program to allow the user to check model parameters to better interpret results. *Figure 3.2* shows an exaggerated plot of the deformed shape of an example model. *Figure 3.3* shows the equivalent stress contours in an undeformed shape of the same example. The POST-PROCESSOR program is listed in the *Appendix* where it is explained in greater detail in the manual.

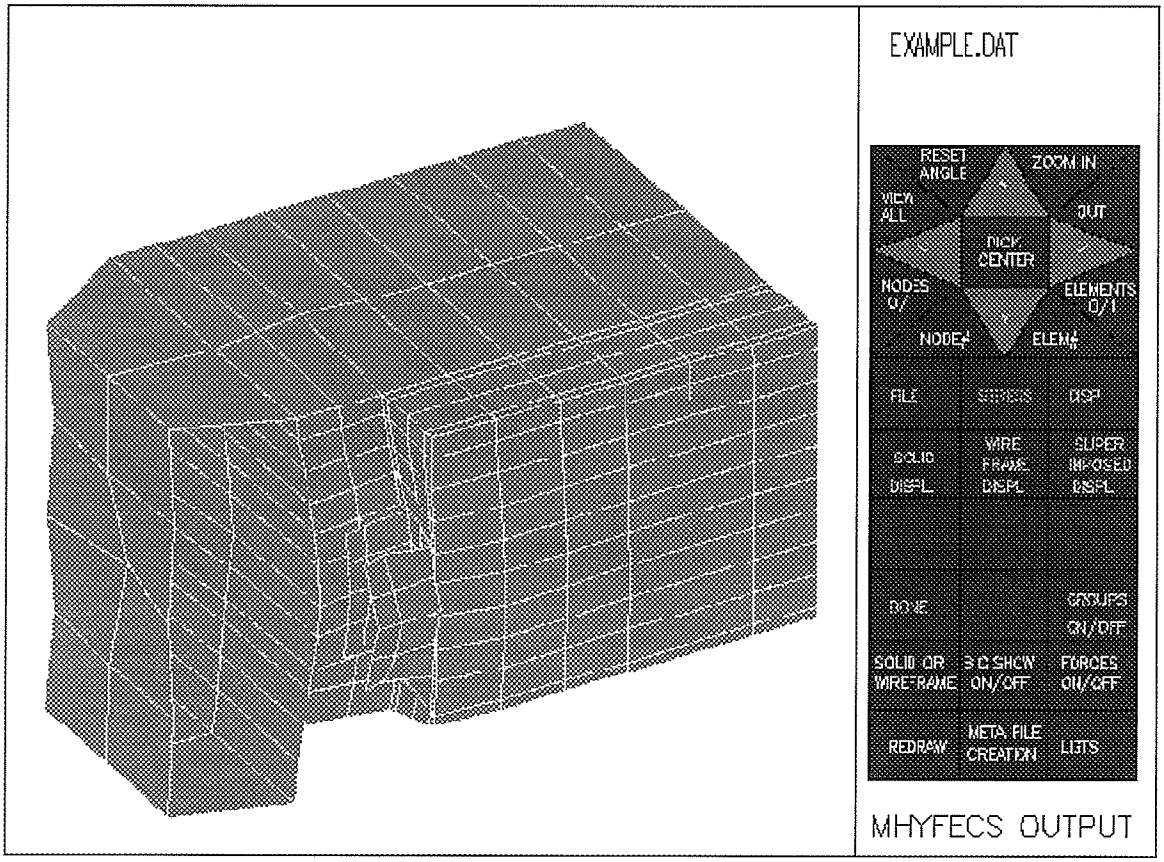


Figure 3.2 POST-PROCESSOR Displacement Example

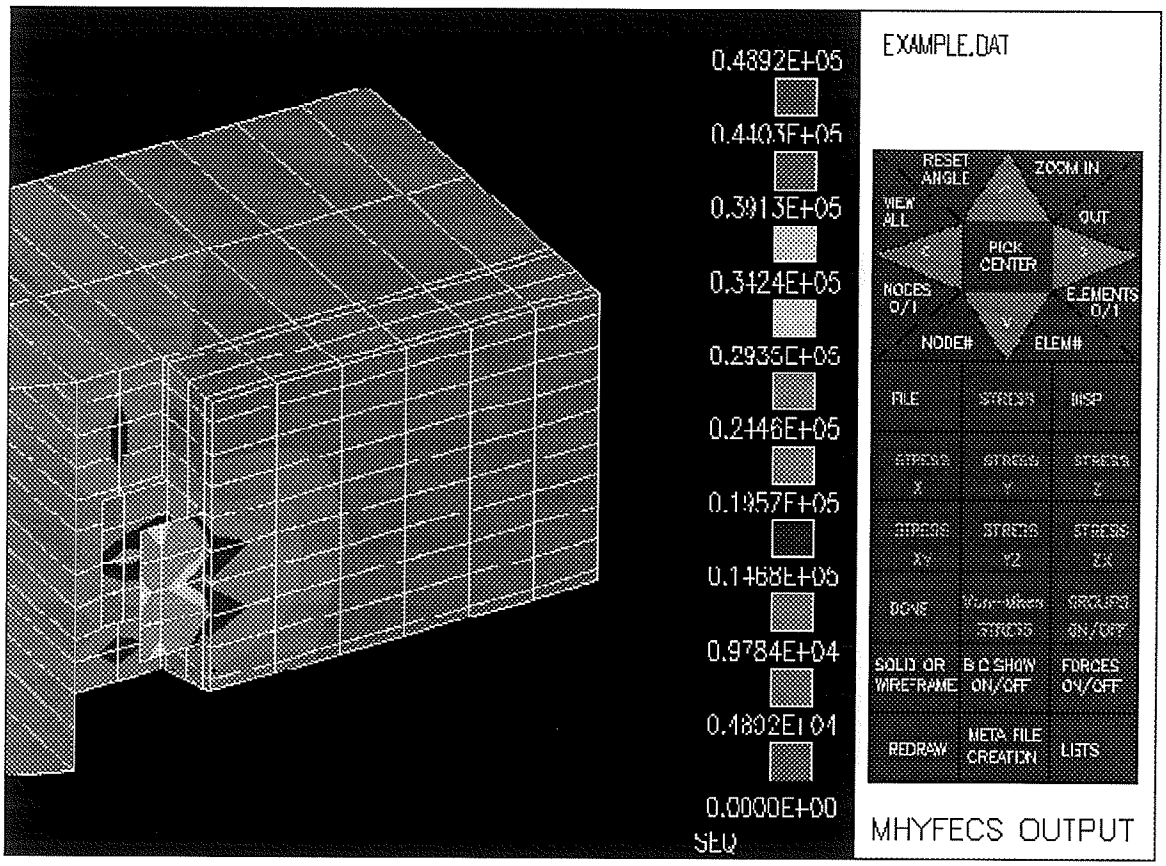


Figure 3.3 POST-PROCESSOR Stresses Example

3.2 Modeling Procedure

The modeling of a typical sluice gate is presented here. Two models are employed in this study, the first is a coarse mesh and the second is refinement around the edge beam and the roller support area. An existing Hydro gate model from a July 1991 report is also discussed here.

3.2.1 MHYFECS Models

The models in this study were developed and solved using the MHYFECS package. The MHYFECS models consist of a mixture of elements in three dimensions with the end plates included since it is the ends of the gate that are of most interest. The first MHYFECS model contains 590 nodes and 696 elements, while the second MHYFECS model which doubles the mesh density around the side rollers and the supporting area, contains 707 nodes and 774 elements. These discretizations are shown in *Figures 3.4* and *3.5*. Due to symmetry, only half of the gate is modeled. Notice the increased refinement of the mesh around the edge beam and roller supports. Both of these models are loaded by hydrostatic pressures. The nodal values of the forces are calculated in a separate spread sheet program to insure each node on the upstream surface received its correct load. The resulting loads for the second MHYFECS model are tabulated in *Table 2*.

3.2.2 The Existing Hydro Gate Model

The key to accurate assessment of the lateral forces acting on the side rollers is the accurate calculation of main roller reactions. An estimation of the main roller reactions was found in a study by Hydro in their July 1991 report[8] and will simply be referred to as the *Hydro Model* in this study. This Hydro Model which is also based on the finite element method, employed a hydrostatic pressure distribution on the gate to find its reaction forces. Recall that the results of the fluid modeling in this study confirmed the assumption of hydrostatic forces as the principle fluid loading on the gate. The main difference between the Hydro Model and this work is that, in the former, a 2 dimensional beam structure was used to simulate the actual three dimensional gate whereas, in the latter, a three dimensional model was used. Several additional beam elements in the

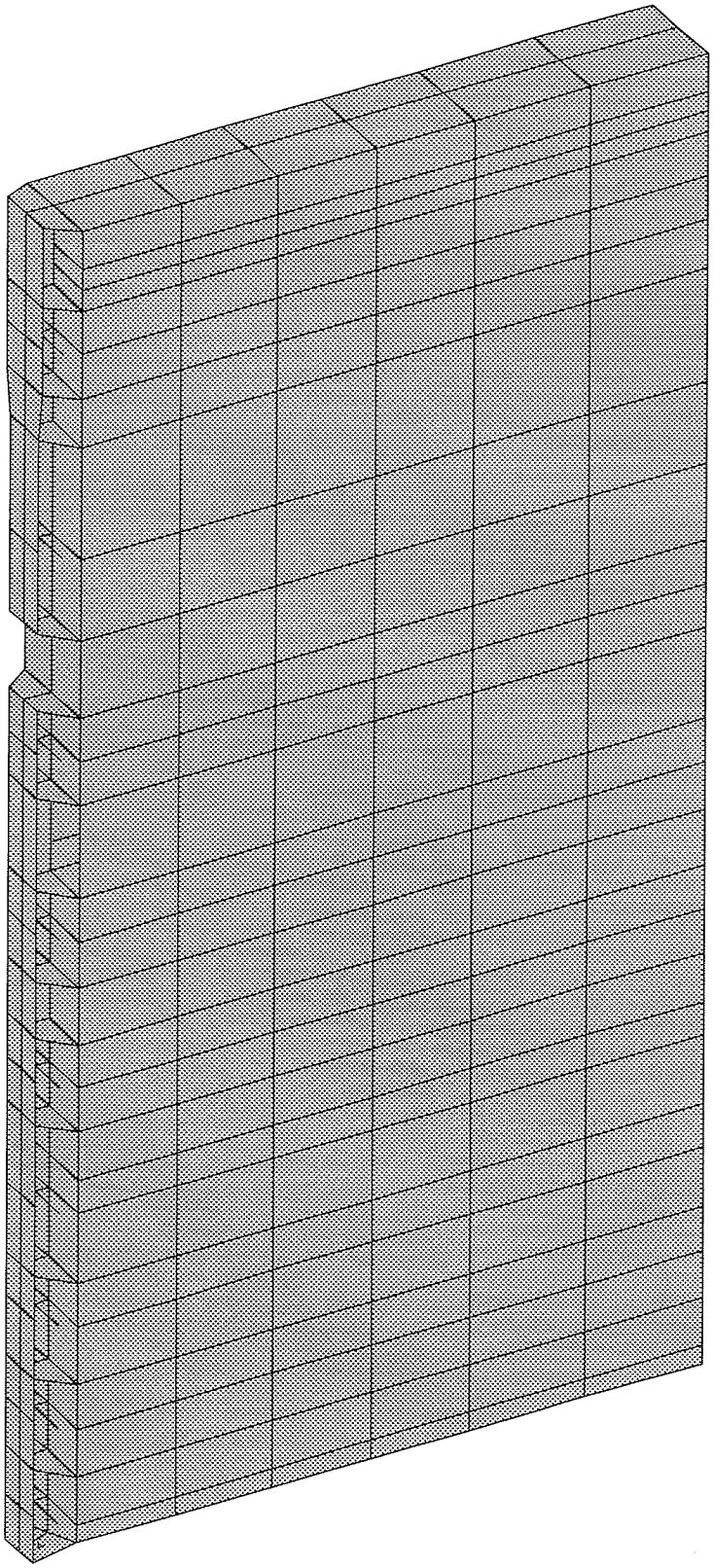


Figure 3.4 MHYFECS 1 Model (590 Nodes & 696 Elements)

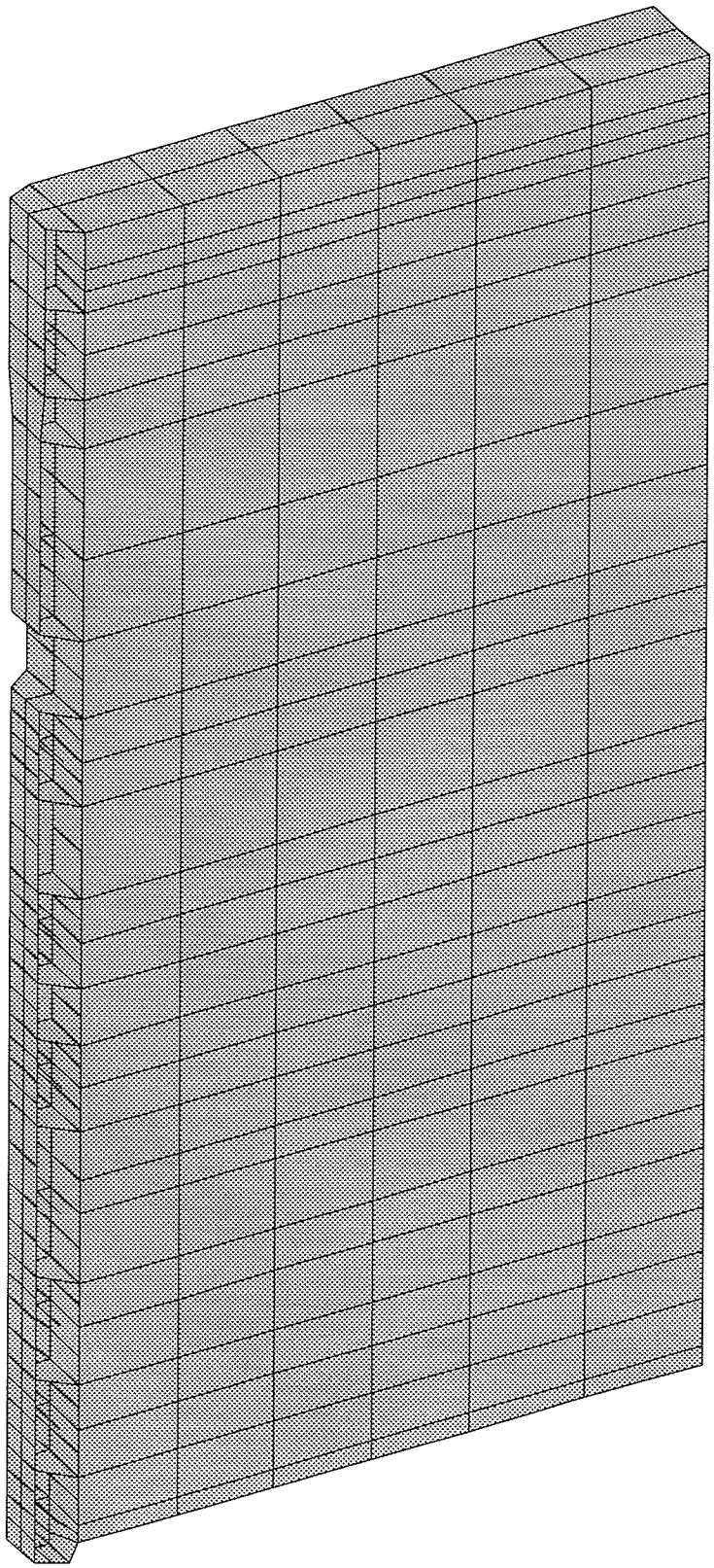


Figure 3.5 MHYFECS 2 Model (707 Nodes & 774 Elements)

Table 2 Nodal Loads for MHYFECS 2 Model

Nodal Loads	Horizontal Position *				
Vertical Position	Outer edge (lbs)	1st, 2nd, and 3rd column	4th column	5th column	Symmetry edge
Top Node	3205.540	6411.080	7052.188	7693.296	3846.648
.	6333.093	12666.18	13932.80	15199.42	7599.712
.	6187.233	12374.46	13611.91	14849.36	7424.680
I	5707.142	11414.28	12555.71	13697.14	6848.570
n	6213.321	12426.64	13669.30	14911.97	7455.986
t	5898.550	11797.10	12976.81	14156.52	7078.260
e	6247.536	12495.07	13744.57	14994.08	7497.043
r	5296.377	10592.75	11652.03	12711.30	6355.653
m	4072.873	8145.747	8960.321	9774.896	4887.448
e	4360.273	8720.547	9592.602	10464.65	5232.328
d	3914.461	7828.922	8611.814	9394.707	4697.353
i	4279.955	8559.911	9415.902	10271.89	5135.947
a	4092.033	8184.067	9002.474	9820.880	4910.440
t	3350.010	6700.020	7370.022	8040.024	4020.012
e	4846.202	9692.405	10661.64	11630.88	5815.443
	4193.393	8386.787	9225.465	10064.14	5032.072
n	2437.086	4874.173	5361.590	5849.007	2924.503
o	3147.803	6295.607	6925.168	7554.728	3777.364
d	3595.583	7191.167	7910.284	8629.400	4314.700
e	3666.407	7332.814	8066.096	8799.377	4399.688
s	2682.403	5364.806	5901.287	6437.767	3218.883
.	1810.622	3621.244	3983.368	4345.493	2172.746
.	98.53726	197.0745	216.7819	236.4894	118.2447
Bottom Node	91475.03	182940.0	201233.0	219526.0	109768.0
Total Applied Force = 1184 (kips)					

** Columns as numbered for left to right in Figure 3.5*

Hydro Model were specifically used to simulate the end plates and other stiffening members which could not be properly modeled in a two dimensional model. The results of the Hydro Model are discussed in the next section.

3.3 Numerical Results

Results of numerical simulation for main roller reaction forces are presented here. Comparison with the Hydro Model is also given. The main roller axle deflection is also discussed.

3.3.1 Main Roller Reaction Forces

The three gate models results are listed in *Table 3* and shown in *Figure 3.6*. Notice how the two MHYFECS results are nearly equal which indicate convergence of the solution. Note that the converged MHYFECS solutions are quite different from that of the Hydro Model results.

Table 3 Main Roller Reaction Forces (Kips)

Roller #	MHYFECS 1	MHYFECS 2	Hydro Model
Top 1	66	72	36
2	79	68	100
3	124	132	131
4	129	129	153
5	152	149	153
6	107	103	152
7	136	138	155
8	116	123	100
Bottom 9	275	271	225
TOTAL LOAD	1184	1185	1205

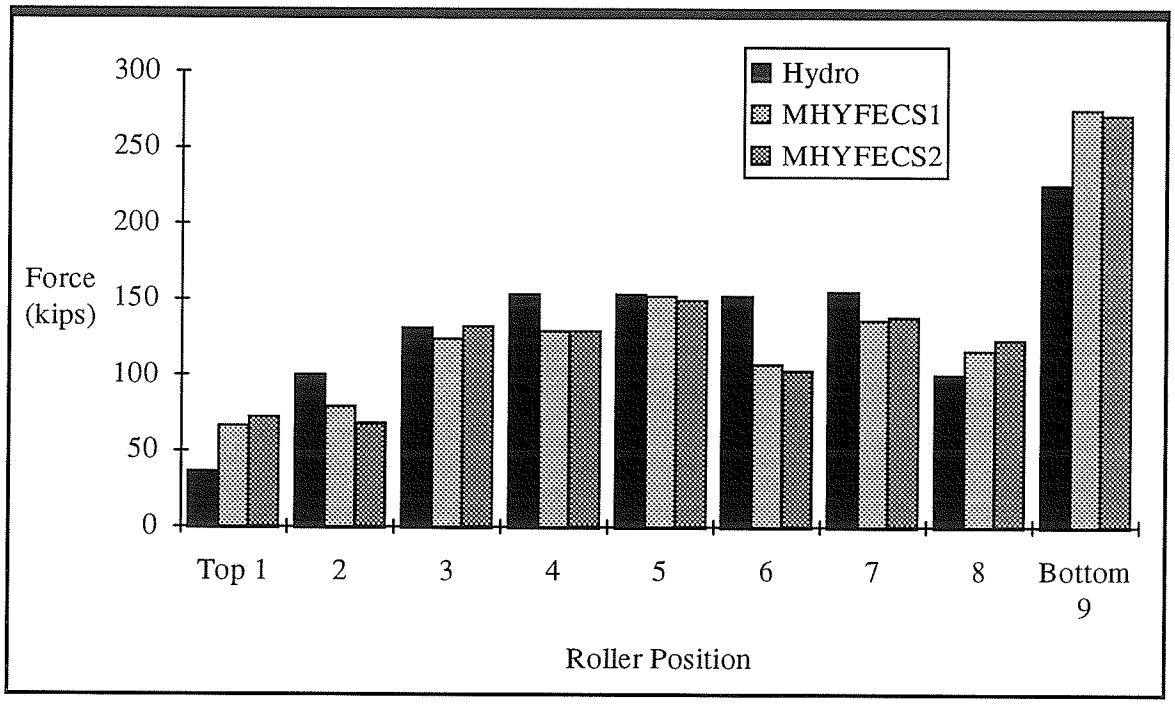
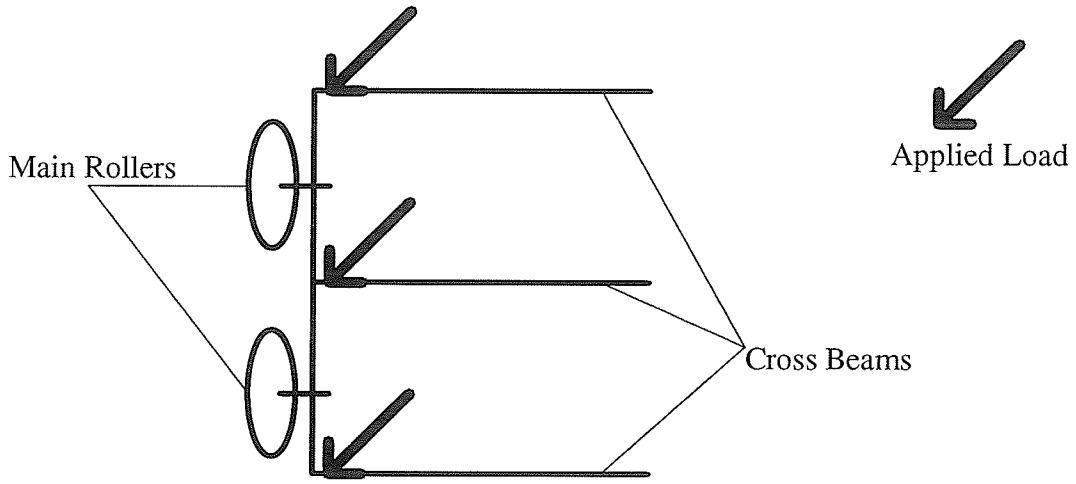


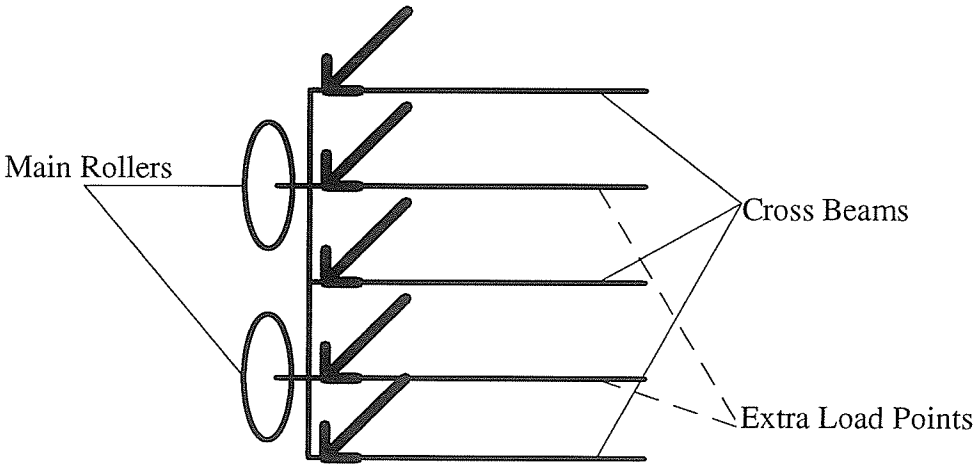
Figure 3.6 Comparison of Roller Reaction Forces

The difference in the total applied force of the two studies is less than 2%, even though the Hydro Model uses a deeper than normal head water depth. The largest difference is in the bottom roller reaction force which is quite important considering the bottom *side* roller is located beside the bottom *main* roller. The Hydro Model predicted this force to be 225 kips whereas the final MHYFECS value is 271 kips, a difference of 20%. This difference can be partially explained through the application of the load in the models. The Hydro Model has its loads applied at the 13 cross beams in the gate, which although gives the correct total force, does not represent the distribution very accurately. The MHYFECS model uses the redistributed pressure on the plates at 23 separate horizontal levels. Thus, the MHYFECS model is especially more detailed in the lower regions of the gate. The lowest forces, that act mostly on the last two rollers, are modeled with four instead of two loading positions. *Figure 3.7* shows the respective loadings in the two models, Hydro Model and MHYFECS model respectively. This combination of more

accurate loadings and the refined plate model instead of a simple beam model created the difference between the Hydro Model and MHYFECS models.



(a) Hydro Model



(b) MHYFECS Model

Figure 3.7 Comparison of load application in the Hydro Model and the MHYFECS model

3.3.2 Main Roller Axle Deflections

A major limitation of the Hydro Model is its inability to model the angular deflections of the roller axles. This is quite important as the angular deflection of the axles is believed to play a major role in the creation of lateral force. The two MHYFECS models, which do have this ability, had nearly identical roller center deflections and the results are shown in *Table 4*.

Table 4 Main Roller Center Deflections

Roller #	Disp. X	Y	Z	θx	θy	θz
Bottom 9	-1.39E-01	5.54E-02	0.00E+00	-0.1714	-.491	-0.005
8	-1.35E-01	2.79E-03	0.00E+00	-1.33E-9	-.449	-0.0122
7	-1.27E-01	2.11E-03	0.00E+00	1.03E-9	-.419	-0.0099
6	-1.28E-01	1.47E-03	0.00E+00	-5.73E-10	-.418	-0.00699
5	-1.13E-01	3.08E-04	0.00E+00	0.0021	-.370	-0.0033
4	-1.10E-01	7.16E-04	0.00E+00	-3.82E-10	-.351	-0.00101
3	-1.04E-01	9.06E-04	0.00E+00	-2.85E-10	-.328	0.0008
2	-9.06E-02	5.75E-03	0.00E+00	-0.013	-.292	-0.0031
Top 1	-5.43E-02	2.21E-03	0.00E+00	-5.54E-10	-.166	-0.0026

(displacement in inches, rotation in degrees)

The rollers themselves were modeled as very stiff beams fixed to the roller path, and thus resulting in zero deflection in the Z direction. The rollers were free to roll, slide, and rotate on their crown as they would in the real situation. The largest deflections seen are the X axis deflections of the roller centers. This is expected as the rollers are closer to the rear of the gate than the front and the resulting bending of the gate and axles will tend to move the contact side of the rollers out. The Y-axis deflections are insignificantly small. The angular deflection of most interest is the rotation about the Z-axis as this is the angle that determines the misalignment of the rollers. The maximum angular deflection found is 0.01° which is very small in consideration of the following two factors. The initial alignment of the axle holes that are bored into the supports is expected not to be exact. The angular deflection caused by the journal bearing clearance will also easily be greater than 0.01°. Due to these two factors, the lateral force is thought to be quite independent of the misalignment caused by the gate deflection. *Figure 3.8* shows a typical misalignments of a gate roller.

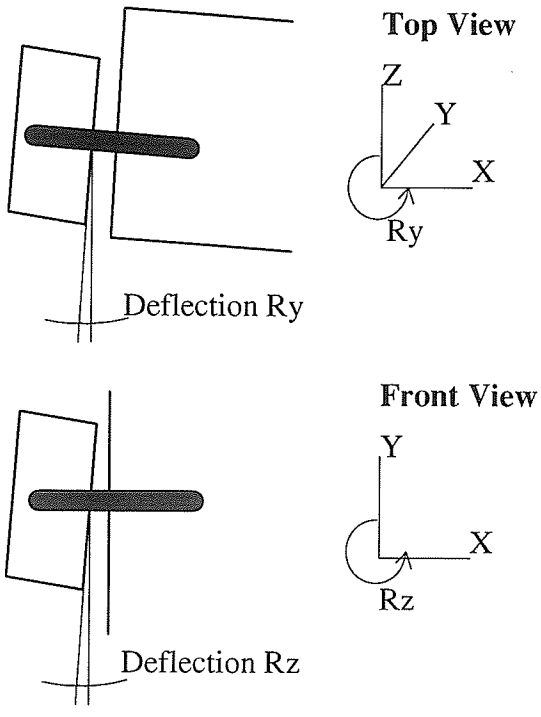


Figure 3.8 Misalignment Angles of Main Rollers

3.4 Lateral Force Estimation

This section presents an estimate of the lateral forces. It starts by discussing the worse case, which proves to be unrealistic. A more realistic model is developed through a more accurate lateral force coefficient and other refinements.

3.4.1 Worst Case Scenario

The minimal angular deflections of the rollers due to gate deflections creates the possibility of the following worst case situation. If, during the hoisting of the gate, one lift cable or screw jack lifts slightly more than the other, all of the main rollers may be misaligned in the same direction due to their journal clearances. This worse case scenario was modeled with a 30% lateral force coefficient as is presently used in the design of new gates. The MHYFECS model used to compute the main roller reaction forces is also used here. The lateral loads of 30% of the main roller reactions are applied at the main rollers and on the symmetry boundary to account for the roller forces from the other side of the gate. The resulting solution yields side roller reaction forces of 224 kips for the top side roller and 485 kips for the bottom roller. To gain more accurate results in the side roller support area, a refined submodel for the bottom side roller support area has been developed as shown in *Figure 3.9*. The model represents the bottom corner of the gate from the bottom skin up to the first cross beam. The model includes the mounting for the bottom main roller and the bottom side roller. The result of applying the 485 kips side roller reaction force to the submodel is shown in *Figure 3.10*. The resulting stresses are far past the yield point for the steel used in the construction of the gate. This implies that the situation of all main rollers producing lateral forces 30% of their normal load in the same direction would cause a catastrophic failure. Since no failures of this magnitude

have occurred, it is obviously quite an impossible situation. Also, the large platic regions predicted would probably never exist as the sluiceway structure would support the gate before such regions could develop. A more realistic model that considers some past and recent experiments must be developed to gain a better estimate of the actual acting lateral forces.

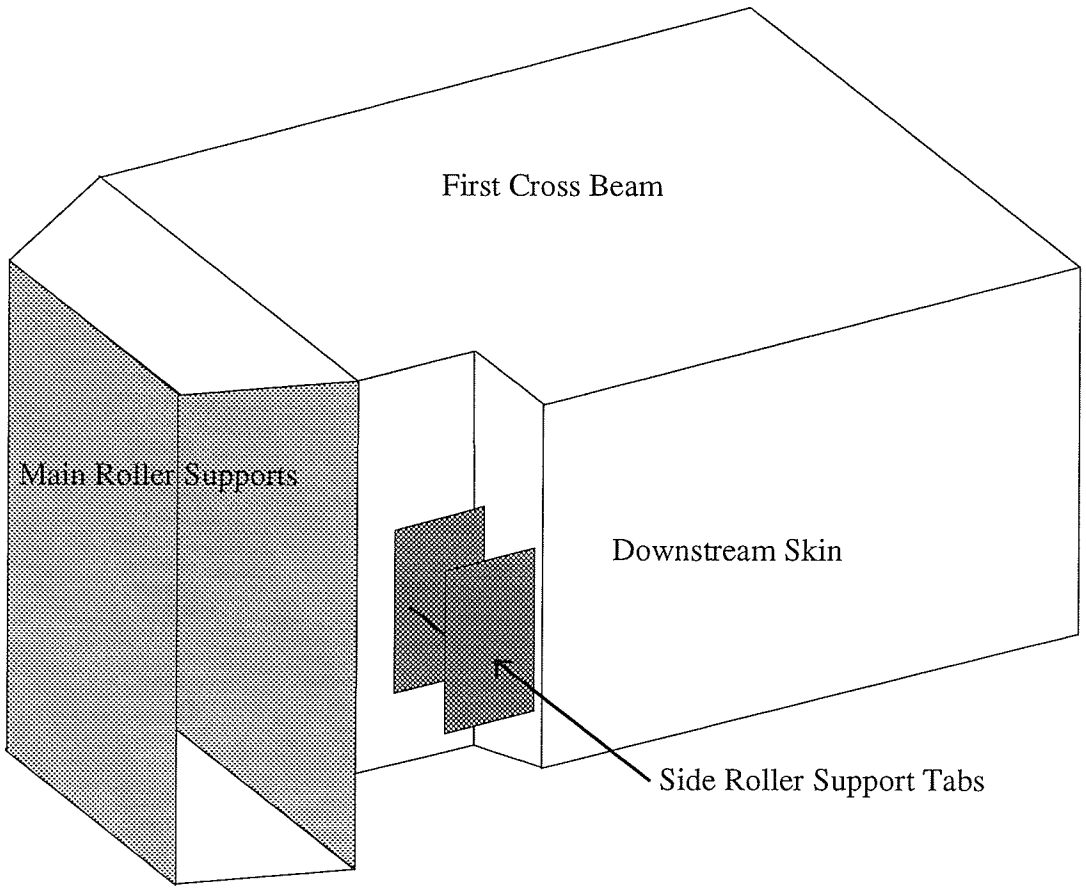


Figure 3.9 Section of Gate used in Submodel

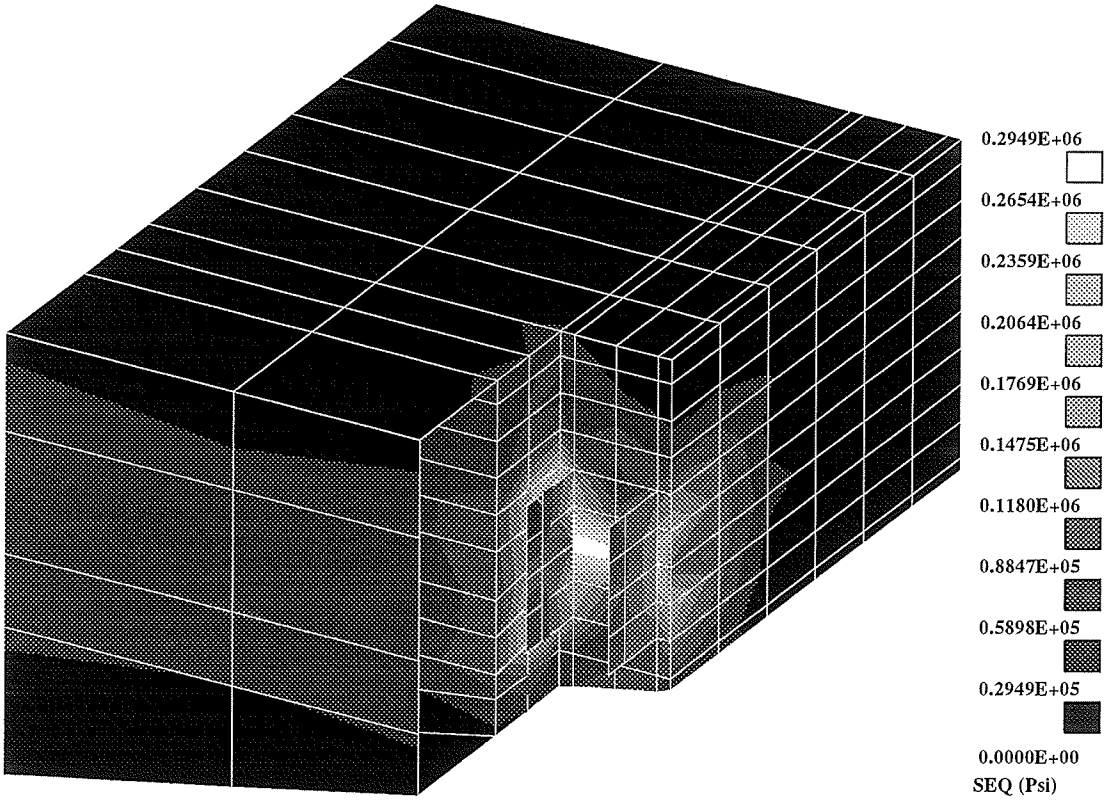


Figure 3.10 Effective Stress in the Worst Case Scenario of 485 Kips Side Roller Reaction Force

3.4.2 Determination of a Realistic Lateral Force Coefficient

Due to the obviously impossible situation in *Figure 3.10,* a more realistic model of the lateral loading is required. The only field data available is that of the Hydro's inspection of a Kelsey sluice gate. Their inspection reports only minor plastic deformations around the side roller supports without specific details. The only conclusion that can be drawn form the inspection report is that in an accurate lateral load model, only localized yield stresses would exist. This qualitative assessment could be used as a rough guideline for the acceptance of a model as accurate.

More recent data is from Manitoba Hydro's roller tests at the University of Manitoba's Civil Engineering Lab. The main purpose of these tests was to determine ultimate loads capable of being supported by the rollers. The test also provided the opportunity to measure the coefficient of friction of the rollers. The rollers used were in clean condition and a new piece of steel was used for the roller path. The main loading actuator, applying a normal load, and the side force actuator, applying a lateral, were simultaneously controlled by a computer. The resulting strains were also continuously recorded by the computer. The experimental setup is shown in *Figure 3.11*.

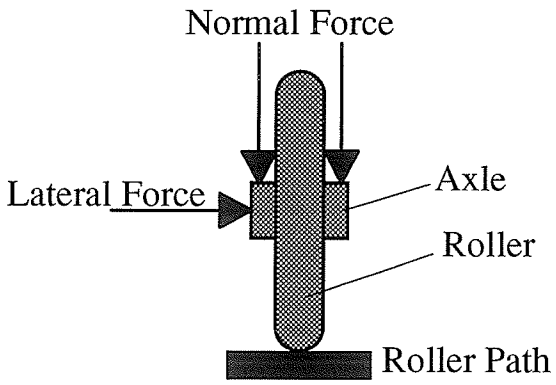


Figure 3.11 Friction Testing Setup

The results of the experiment showed that slippage between the roller and roller path occurred in the range of a 0.18 to 0.20 coefficient of friction. This however does not directly translate into a coefficient of lateral force of the same magnitude as there is a difference between the coefficients of friction and lateral force. This is demonstrated in the 1963 ASME paper 'Experimental Determination of Lateral Forces Developed by a Misaligned Steel Roller on a Steel Rail', by Cress and Talbert[9], which outlines their experimental lateral force determination method. Their experiment drives a misaligned roller down a roller path, with the roller rolling, while simultaneously measuring the lateral forces. The range of misalignment angles covered is from 0° to 3° with varying surface conditions of roughness. Hydro tested the coefficient of friction, μ, of the

different surface conditions in the same manner as the 1963 experiment. In the paper, a coefficient of friction test showed fine ground steel surfaces to have a μ of 0.16 and rusted surfaces a μ of 0.22. These compare favorably with results of the Hydro test. The paper's results show that the lateral force coefficient rise quickly with misalignment and level off at a value of 0.27 at about 1° of misalignment for all of the surface types. This is shown in *Figure 3.12*.

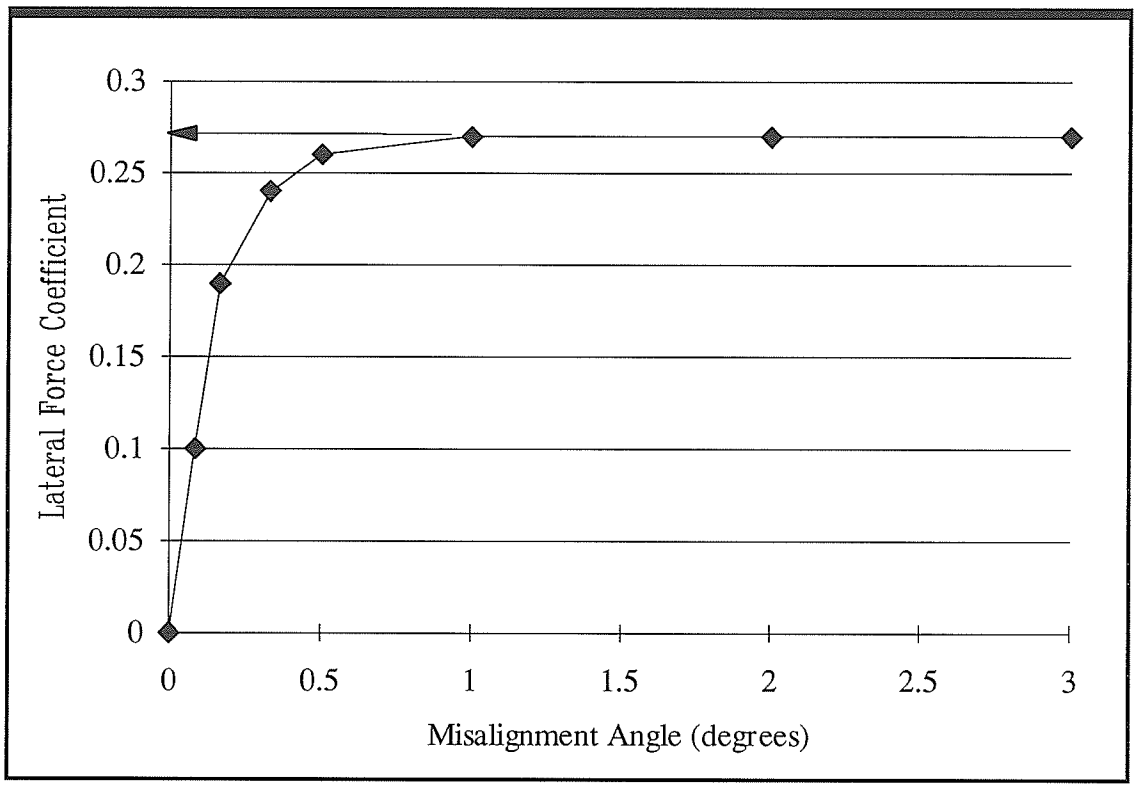


Figure 3.12 Lateral Force Determination Results[9]

From this graph it is obvious that the misalignments caused by the gate deflections would have a negligible effect. However, an initial misalignment of as little as one degree could cause the maximum lateral force of 27% of the normal force. The most important detail pointed out by this paper is the fact that the lateral force coefficient is not a function of the friction coefficient but the misalignment angle. Due to the agreement of the friction

coefficients between the paper and the Hydro tests giving credibility to the lateral coefficient of 0.27, this value will be used in creating a realistic lateral force model.

3.4.3 A Realistic Lateral Force Model

The use of a lateral force coefficient of 0.27 instead of 0.30 will still cause unrealistic stresses in the side roller support area. The situation where all the rollers are misaligned in the same direction is also thought to be rather unlikely as the initial misalignment from installation would be a random process causing misalignments in both directions. Due to the initial misalignments, the journal bearing clearances, the unevenness of the roller path, and the very small deflection caused misalignments, a less simplistic model must be used to predict a typical situation. This situation is that one half of the rollers have enough misalignment to create a lateral force coefficient of 0.27 either through deflection or initial misalignment. Some rollers on the other side of the gate also create lateral forces, but due to their opposite deflections, they are to be modeled as having lateral forces acting the other way. The main rollers with the largest deflection caused misalignments, 6, 7, and 8, are assumed to acting the other way. This combination results in all of the rollers on one side of the gate creating a lateral force and the other side rollers with reactions of 103, 138, and 123 kips acting the other way. This combination reduces the side roller load to 126 kips at the bottom roller. The realistic lateral force model is shown in *Figure 3.13*. This result has only a limited area in which yield stresses were exceeded which agrees with the findings by the Hydro inspection.

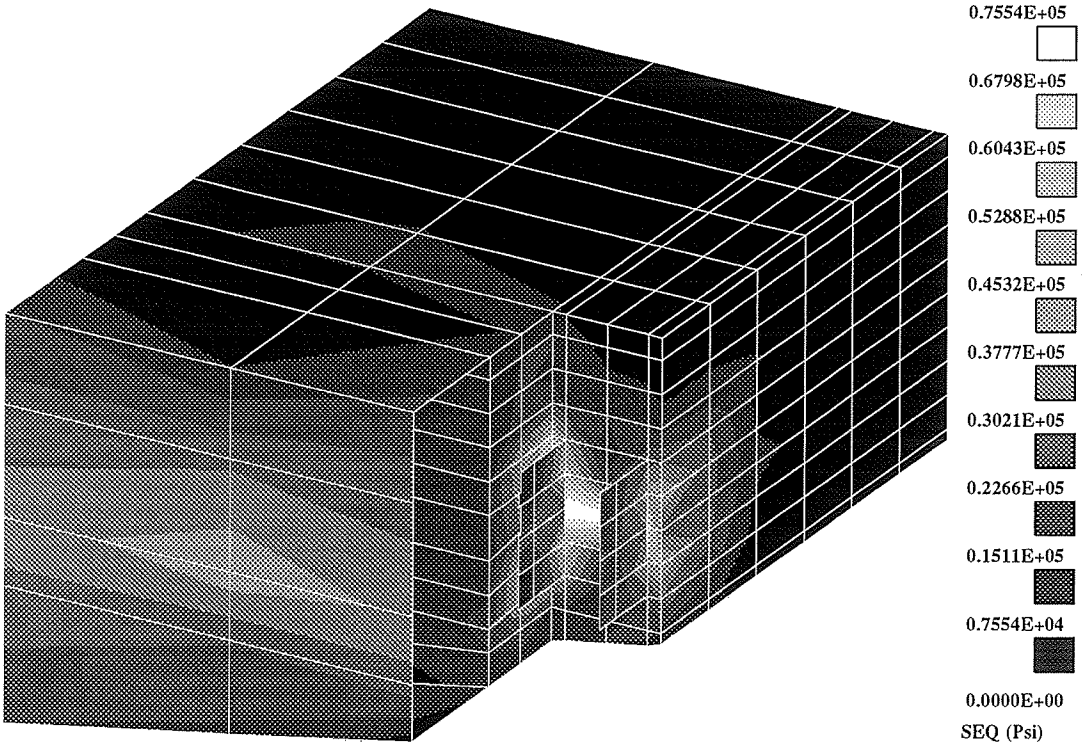


Figure 3.15 Effective Stresses in Realistic Lateral Force Model with 126 Kips Side Roller Reaction Force

3.5 Broken Lift Cable Situation

Another situation that can cause lateral loadings is that of a broken lift cable. If one of the two lift cables breaks, or is very slack, the resulting lifting force will be off center and will create other side roller forces. This is quite an easy situation to analyze as all the necessary data is known. The weight of the gate is 190 kips. The two lifting cables are 42 feet apart at the lifting lugs on the top corners of the gates and the two side rollers are 40 feet vertically apart. The resulting free body diagram is shown in *Figure 3.14*. In this figure fluid loadings and reaction forces are not included as this is strictly the case of supporting a gate with one broken lift cable. The gate would never be moved with a broken lift cable.

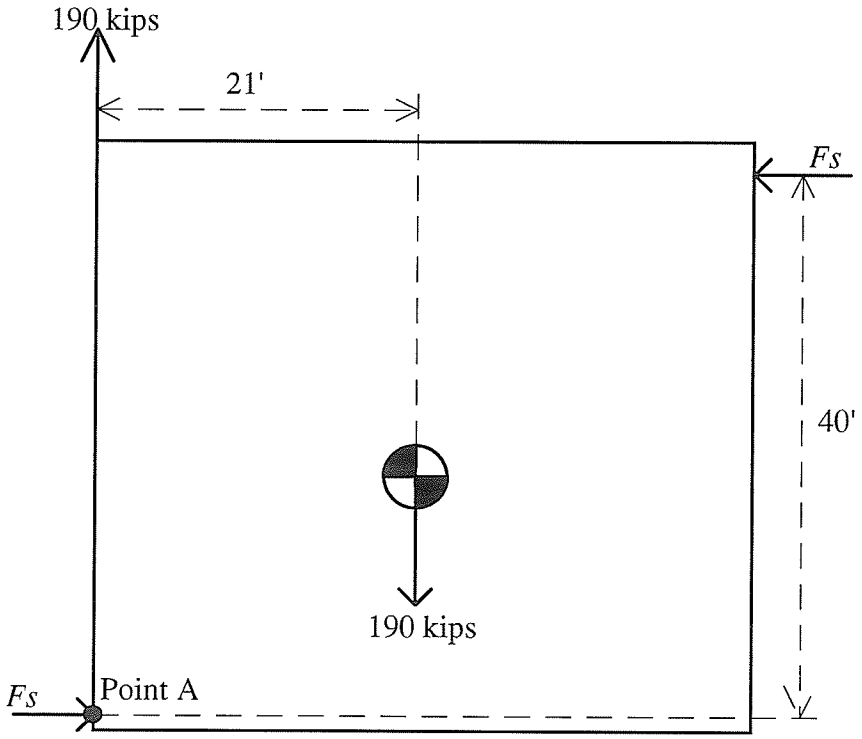


Figure 3.14 Free Body Diagram

The resulting side force, *Fs*, can now be easily found by taking moments about point A. This moment equation is as follows:

$$\sum M_A = Fs \cdot 40 - 190 \cdot 21 = 0$$
$$\therefore\ Fs = 99.75\ kips \qquad (23)$$

The 99.75 kip load created by a broken cable is less than the estimated side roller load and hence can safely be assumed as an acceptable load once the side roller support area is reinforced to withstand the estimated side roller loads from other sources.

3.6 Redesign of Side Roller Support Area

The results in *Figure 3.13* showed a limited area of yield stresses consistent with Hydro's observation of a limited area of yielding in the Kelsey gate. The situation of half of the main rollers lateral forces acting one way and three rollers on the other side of the gate acting in the opposite direction will be used to model the side roller support reinforcements since it closely matches the observed gate reactions. The existing structure must be reinforced not only to meet the structural requirements, but the cost factor and field operations such as cutting and welding must also be considered. The existing structural member behind the side roller support tabs is a 8" by 8" by 1/2" angle. To reinforce this member, a simple solution is to increase the thickness of the angle from the bottom skin to the first cross member. The first reinforcement design does just that with an 8" by 8" by 1.5" piece of angle. A schematic of the reinforcement is shown in *Figure 3.15*. The 1.5" thick angle in the reinforcement meets the needs of supporting the side roller support tabs as can be seen in the reduced peak effective stress in *Figure 3.16*.

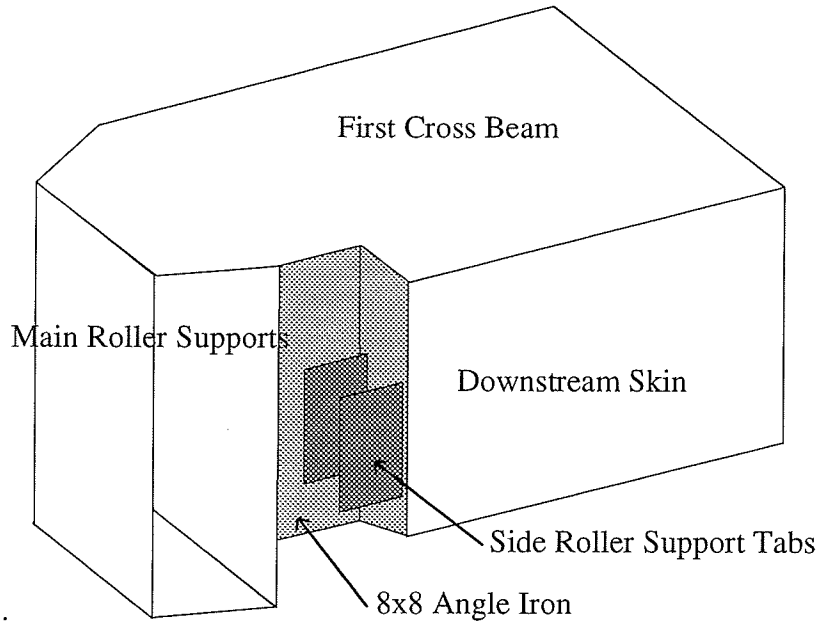


Figure 3.15 Schematic of Submodel

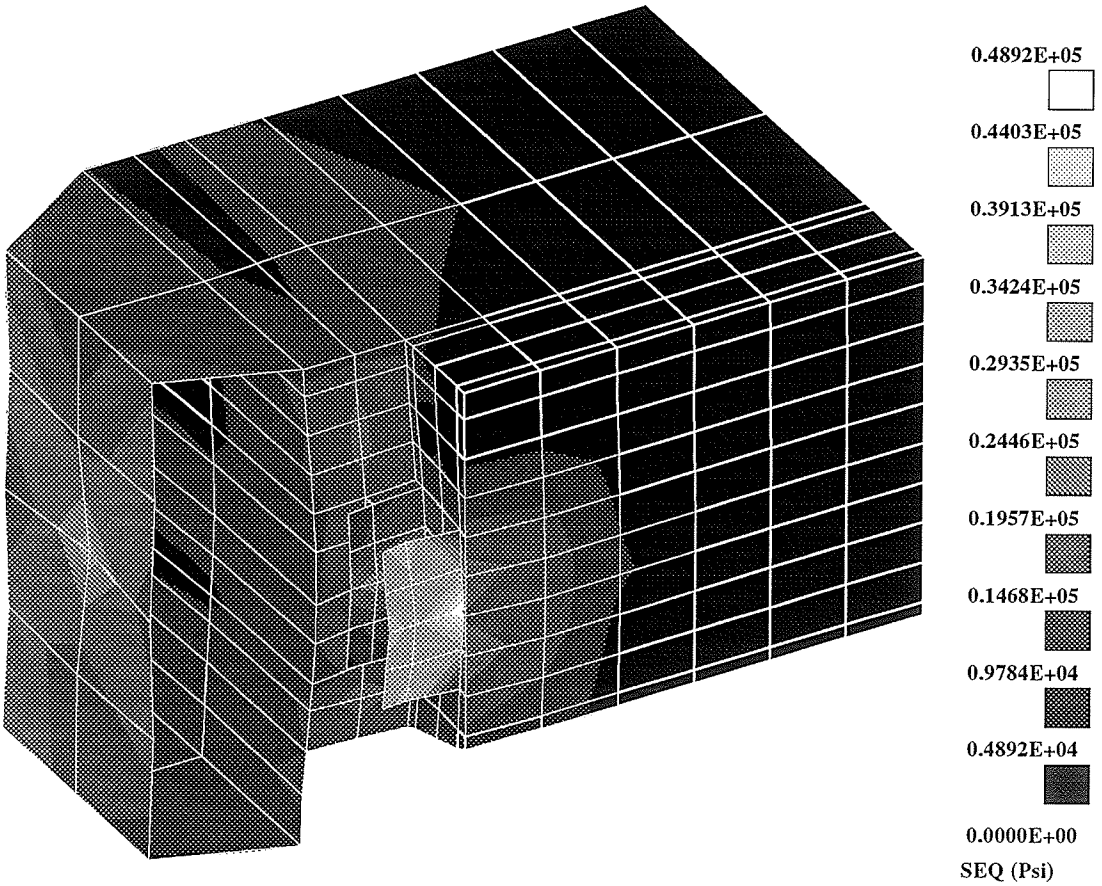


Figure 3.16 Effective Stresses in 1.5" Angle Redesign

The problem in the *Figure 3.16* design is that the actual side roller support tabs experience unacceptable stresses. The next logical design step is to use the same type of material to reinforce the support tabs as the use of common sized materials may possibly have a cost benefit. The stresses resulting in the new reinforcement design are shown in *Figure 3.17.* From this figure, not only is it evident that the side roller support area is now strong enough to withstand the estimated lateral loads, but that the low peak stresses show a conservative design with a large factor of safety.

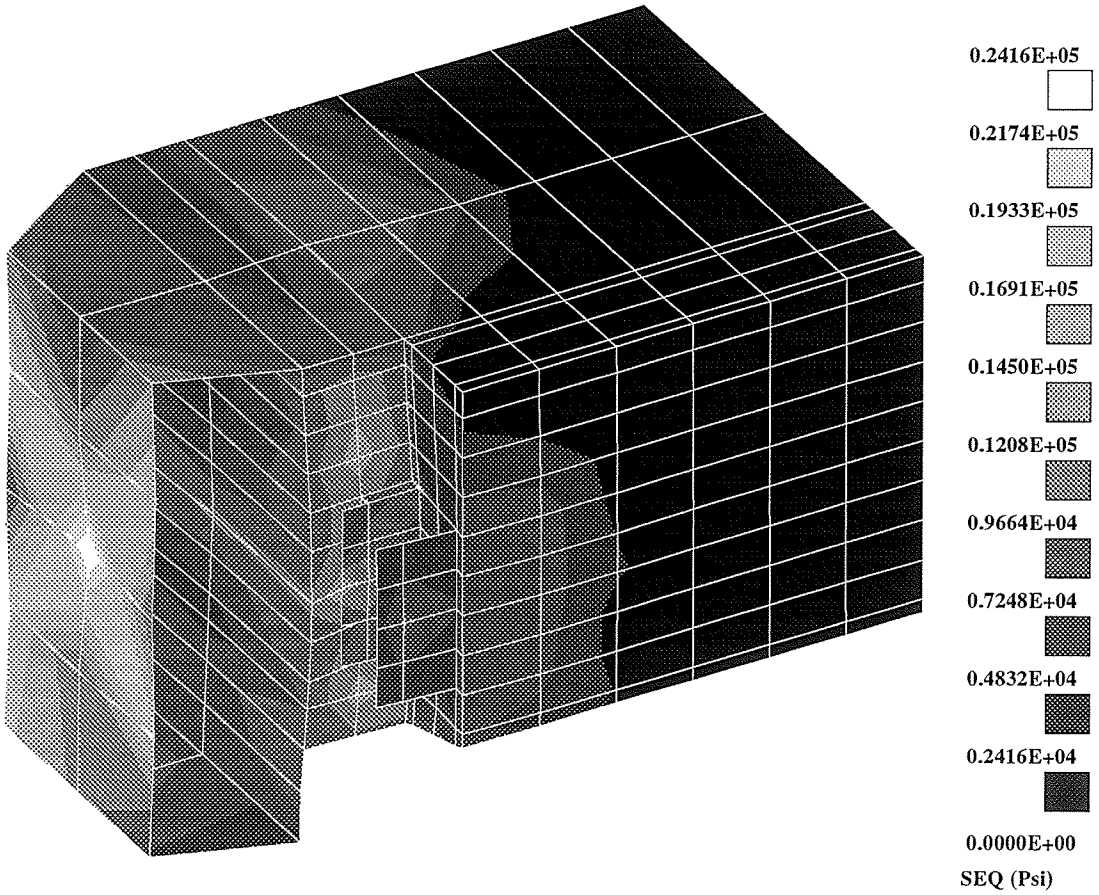


Figure 3.17 Effective Stresses in 1.5" Angle and Support Tabs Redesign

CHAPTER 4

BANDWIDTH REDUCTION ALGORITHM

4.1 Need for Bandwidth Minimization

Most realistic engineering analyses involve the solution of a large system of algebraic equations in matrix form on a computer. That is, solving the following set of equations,

$$[K]\{u\}=\{f\}$$

where **K** is assumed to be a symmetric matrix. The storage and solution of **K** are of great interest as the available storage, computational effort, and computational time are normally the limiting factors in an analysis. Since most modern storage schemes store only a specific portion of the matrix, it is desirable to minimize the size of the stored matrix. The amount stored is determined by the maximum distance from the major diagonal of the matrix to a non zero term as shown in *Figure 4.1*. Utilizing symmetry, only the part of the matrix N by the half bandwidth is stored versus the entire N by N matrix, where N is the order of the matrix.

$$\begin{bmatrix} D & X & 0 & 0 & 0 & 0 & 0 & 0 \\ X & D & X & X & 0 & 0 & 0 & 0 \\ 0 & X & D & X & X & X & 0 & 0 \\ 0 & X & X & D & 0 & 0 & 0 & 0 \\ 0 & 0 & X & 0 & D & X & 0 & 0 \\ 0 & 0 & X & 0 & X & D & X & X \\ 0 & 0 & 0 & 0 & 0 & X & D & X \\ 0 & 0 & 0 & 0 & 0 & X & X & D \end{bmatrix}$$

(a) Original Matrix

$$\begin{bmatrix} D & X & 0 & 0 \\ D & X & X & 0 \\ D & X & X & X \\ D & 0 & 0 & 0 \\ D & X & 0 & 0 \\ D & X & X & 0 \\ D & X & 0 & 0 \\ D & 0 & 0 & 0 \end{bmatrix}$$

(b) Halfbandwidth Stored

Figure 4.1 Example Matrix with Half bandwidth of 4

In *Figure 4.1 (a),* the diagonal terms are denoted by D, the off-diagonal non-zero terms with X, and the zero terms with 0. Notice how in the third row, and third column, the farthest non-zero term is 3 terms away from the diagonal. This results in a half-bandwidth of 4 which is the width of the storage matrix also depicted in *Figure 4.1(b).* Such a storage scheme can become very inefficient if the matrix to be stored is a very sparse matrix as many zero terms have to be stored. The execution time of a 'bandsolver' is $C*(N*B^2)$ where N is the size of the matrix, B is the bandwidth, and C a time constant for the computer used. Note that the B factor in the solution time of a matrix is related by a squared relationship. Both the storage requirements and solution time can be drastically reduced by lowering the bandwidth through a reordering of the equations in the matrix or as more commonly done, a renumbering of the nodes. The need for bandwidth reduction is especially acute when one considers that the vast majority of engineering computations, including the MHYFECS programs used in this study, are performed on PCs that have limited memory. Another reason to automate the bandwidth reduction process is that during the development of a finite element model, the mesh is usually built in sections with automatic meshing and mesh refinement procedures that are designed to be convienient for the user. These refinements and automatic procedures are indeed helpful, but do not generally produce the smallest bandwidths.

4.2 Existing Algorithms

The most important existing algorithms are outlined in brief in the following sections. More details can be found in the original papers listed in the references. Most existing algorithms can be grouped into two main categories, direct reduction algorithms, graph theory algorithms, or a hybrid of the two.

4.2.1 Direct Reduction Algorithms

These algorithms work by directly interchanging the rows and columns in the matrix for reduction of bandwidth. For example, let's say the maximum bandwidth of a problem was 11 due to a valency* between nodes 10 and 20. In this case nodes 11 to 19 would be iteratively renumbered above 20 or below 10 in hopes of bring the old nodes 10 and 20 closer together in the new numbering. Such algorithms have been presented by Grooms[10] and Puttonen[11]. They produce good results but take an inordinate amount time in calculation.

4.2.2 Graph Theory Algorithms

This second category of algorithms use concepts of graph theory to create level structures that are used in the renumbering process. The main algorithms based on these concepts have been presented by Cuthill and McKee (CM)[12], Gibbs, Poole and Stockmeyer (GPS)[13], and a relatively new algorithm by Burgess and Lai (BL)[14]. The graph theory method creates level structures in the following manner; first, a start node is selected and labeled as a level one node, then, all the nodes with connectivity to the start node are labeled level as nodes in level two. Next, the nodes connected to last calculated level are assigned the next level number if not yet assigned to a previous level. This is repeated until all the nodes are assigned to levels from 1 to n, the number of levels. This level creation is demonstrated in the simple example of *Figure 4.2*.

---

* term defined on next page

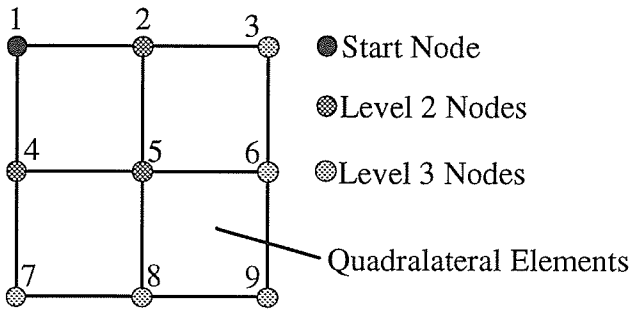


Figure 4.2 Level Structure Creation Example

Notice how nodes 2, 4, and 5, which are connected to the start node 1 due to the use of quadrilateral elements, are all in level 2 and the remaining nodes in level 3. The nodes are then renumbered from level 1 to n through each level by a variety of criteria outlined later. This description is obviously oversimplified but does allow us to break the graph theory method down into its three basic steps as follows:

i) start node selection

ii) level structure creation

iii) renumbering of the nodes

Many different algorithms do each step differently, so each step will be described in more detail below, but first we will define some relevant terms. The first term to define is *level structure width.* This term describes the maximum number of nodes in any level of the structure and hence, the width of the structure. In *Figure 4.2,* the level structure width for the second level is 3, for the 3rd level, the level structure width is 5. The next term, *level length,* is the number of levels in the structure, and hence the length of the structure. In *Figure 4.2,* since there are 3 levels, the level length is 3. Lastly, the term *valency* of a node is the number of other nodes connected to that node. For example, node 2 in *Figure 4.2* has a valency of 5 since it is connected to nodes 1, 3, 4, 5, and 6.

The *start node selection* is very important as a poor selection will result in an unnecessarily high level width which in turn results in large bandwidths. The selection of a starting node in the CM algorithm is as follows. First, the valency of all nodes is calculated and the level structures for all the nodes with the lowest valency are created. These level structures are renumbered and the one of minimum half bandwidth is selected. The great many level structure creations and respective renumberings make this method overly slow in most large problems. Another shortcoming of this is that the half bandwidth can never be less than the width of the level structure.

The GPS algorithm, an improvement over the CM algorithm, uses an improved method to find the start node. GPS starts with an arbitrary node v of minimum valency and creates a level structure starting at node v. Level structures are started from each node in the last level of the first structure. If any node u of the last level creates a longer level structure than the original level structure, the node v is set to u and the search process repeated. If no new start node is found, then the minimum width level structure starting at node v and ending at node u is used. These two nodes represent the start and end of a *pseudo-diameter* of the graph of the problem. The two level structures, starting at v and u respectively, are created and combined to produce a level structure possibly smaller than a simple level structure created from either nodes v or u or as a start node.

Level Structure Creation is the next step in the reduction process. The level structure creation process is, in most methods, the straight forward method described previously except for the GPS algorithm. The GPS algorithm comparing the position of each node in the two level structures created from nodes v and u. These nodes that are in overlapping levels in both structures are not moved. Those nodes that do not overlap are called *disjointed nodes*. The disjointed nodes are put in the levels that result in minimum

level widths. This final structure is renumbered for the reduced bandwidth. The BL algorithm also uses the GPS starting node search as presented. The GPS start node search is superior to the CM method in most cases and hence is more widely accepted. All the graph theory methods find the peripheral nodes of minimum degree and work from there to find a minimum width path.

Renumbering of the nodes in the level structure is now performed to create the new nodal ordering and reduced matrix bandwidth. This procedure differs between methods significantly as do the results. The CM algorithm numbers the nodes connected to the lowest numbered node in the previous level in order of increasing valency until all nodes in the present level are numbered. It is important to remember that the CM algorithm renumbers each of the level structures generated for comparison. This method has the obvious draw back of not recognizing that the nodes that do not connect to the next level should be numbered first to keep their numbers closer to those of the previous level and to assign the nodes connected to the next level higher numbers. The GPS renumbering is very similar to CM but with a few additional constraints since the level structures produced by the GPS method are not simple structures as in the CM method. First, the numbering begins at the node v or u, the one of lower valency. The nodes are then ordered in the next level starting with the nodes connected to the lowest numbered node in the previous level. When all the nodes connected to the previous level are numbered, number the nodes connected to the already renumbered nodes from lowest to highest number. Repeat this for all the levels in the structure. This method rectifies the short comings of the CM algorithm. The BL algorithm starts with the GPS start node search but then uses a *level smoothing* method to produce lower level structures and then possibly lower bandwidths. Note that a lower structure width does not always create a lower bandwidth if the renumbering is not done carefully. The BL algorithm takes advantage of the fact that many nodes in a level have no connection to the next level but

only to the previous and same level. These nodes can be moved back into lower levels to smooth the level structure. The renumbering of BL is done using the criteria set down by Cheng[18] and CM.

4.2.3 Hybrid Algorithms

A hybrid algorithm presented by Armstrong[15] uses the GPS algorithm to initially renumber the nodes and then uses row column permutations to improve the ordering. This is a very logical step in development as Armstrong observes that graph theory algorithms quickly give the gross pattern of an optimum numbering but fail to produce the finer details of a row column permutation algorithm. Armstrong's method is presently the only well tested algorithm that outperforms the GPS method. Well tested is defined here as algorithms that have been tested with Everstine's[16] collection of problems. The most important point to note about Armstrong's algorithm is the way in which the direct iterations are carried out in two distinct manners. The first iteration is a simple node switching that only proceeds if the bandwidth reduces. The second is a *jiggling* component that attempts to over come local minima that can not be directly reduced. The bandwidth is allowed to slightly rise in the hope that the new order can be reduced to a smaller bandwidth with more direct reductions. This jiggling does produce reductions, but at a cost. The time for jiggling reductions can be between 10 and 50 times as long as non-jiggling reductions which, unless used for repeatedly solved matrices such as eigenvalue problems, can be longer than the solution of the matrix itself.

4.2.4 Ponderation Algorithm

This algorithm is niether a direct reduction or graph theory but deserves mention as the results in some cases are quite good. The method presented by Akhras and Dhatt[17] uses the concept of *Ponderation and Span.* Ponderation is found such that if node 1 is connected to nodes 2, 3, and 4, then the sum of 1+2+3+4=10 divided by 4 equals the ponderation of 2.5. The span is the sum of the largest and smallest numbered node connected to a node including itself. For example, the above node connected to nodes 2, 3, and 4 would have a span of 1+4=5. The method works by alternately sorting the nodes by increasing ponderation and span. The nodes are renumbered from the sort and the new values for ponderation and span found. The process is repeated until no further reductions occur. The results are quite good but this method has the disadvantage of being slow compared to graph theory methods.

4.3 New Algorithm

Since the most successful and well tested algorithms to date are the GPS and Armstrong's, which actually uses GPS as its starting routine, an initial graph theory based renumbering is believed best to create the gross structure of near optimum numbering of nodes. A different approach will be presented here which attempts to further draw on graph theory and mimic the way a person can look at the whole mesh of a structure at once. This new method is similar in concept to GPS but fundamentally different in execution. Whereas the GPS algorithm finds a *pseudo-diameter*, the new method works at determining the *pseudo-radius* to find possibly the same start node as GPS but in a more efficient manner. The new algorithm will also limit the amount of iteration that is

done compared to Armstrong's method. The increased time used to make the finest refinements in Armstrong's method may not always be recovered in the static problems solved with the MHYFECS programs. Some iterations are to be used as Armstrong's results compared to GPS are impressive in just the non-jiggling reductions.

4.3.1 New Start Node Selection Method

The new start node search method works as outlined in *Figure 4.3*. The philosophy behind this methodology is that the *'center node'* found represents the *pseudo-center* of the graph such that the resulting level structure made from it will have one of the end nodes from the GPS method or even a superior choice in the last level. The resulting search for the minimum level structure will therefore not restart as is possible in the GPS algorithm saving execution time. The advantages and disadvantages of this new method are shown later in the test examples.

As an example, Figure 4.4 shows the same 9 node mesh as used previously. Now notice how node 5 has been selected as the center node since it has the highest connectivity. In this example, all the nodes except 5 are in the last level of the resulting level structure S starting from node 5. Since node 1 has its level structure created first and no other level structure width would be less, the final start node selected would be node 1.

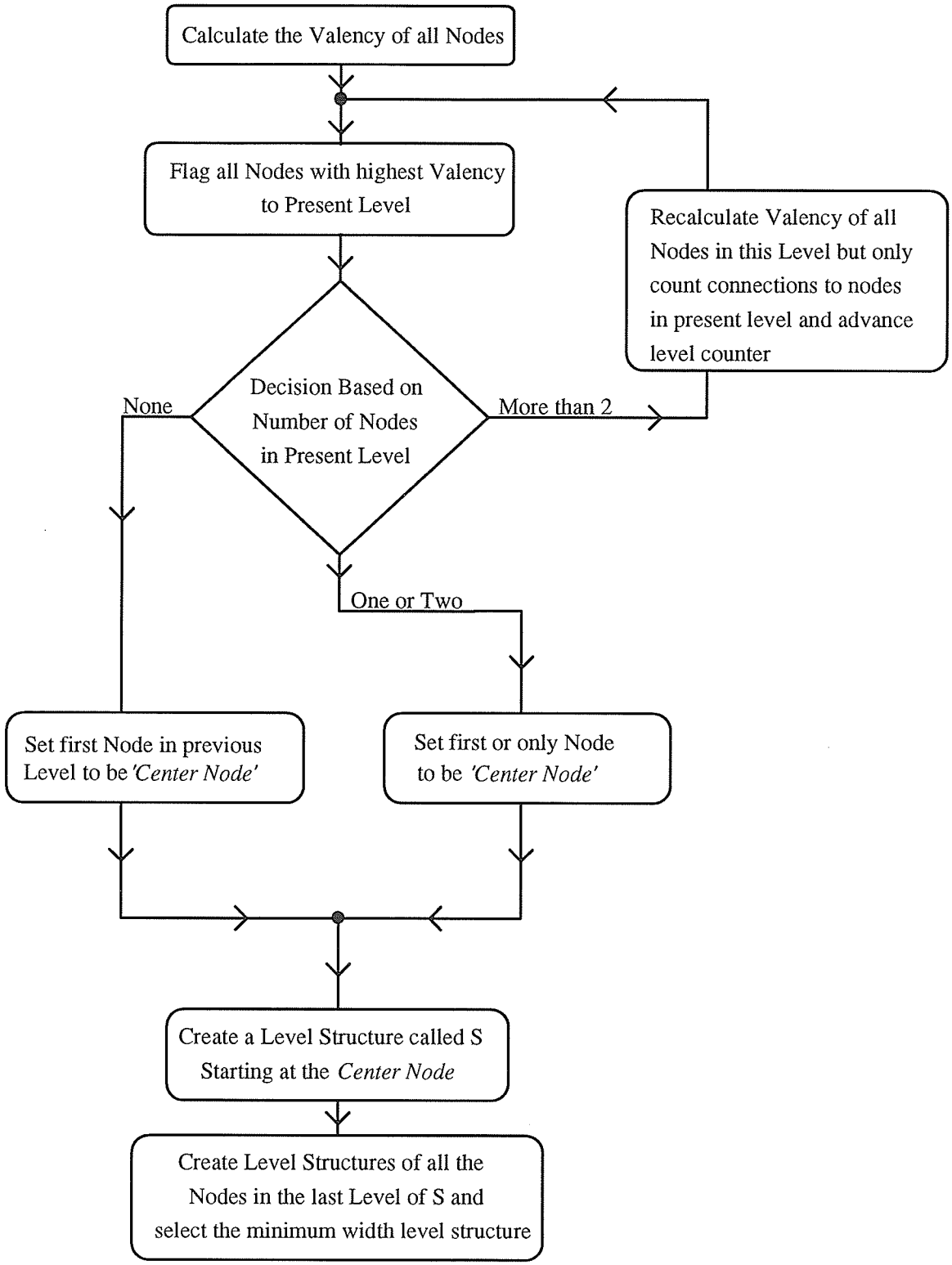


Figure 4.3 Start Node Selection and Level Structure Creation method for New Algorithm

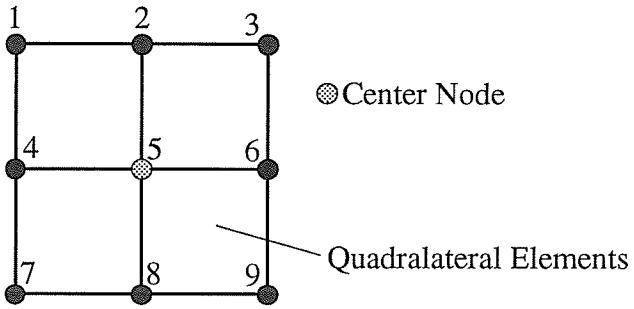
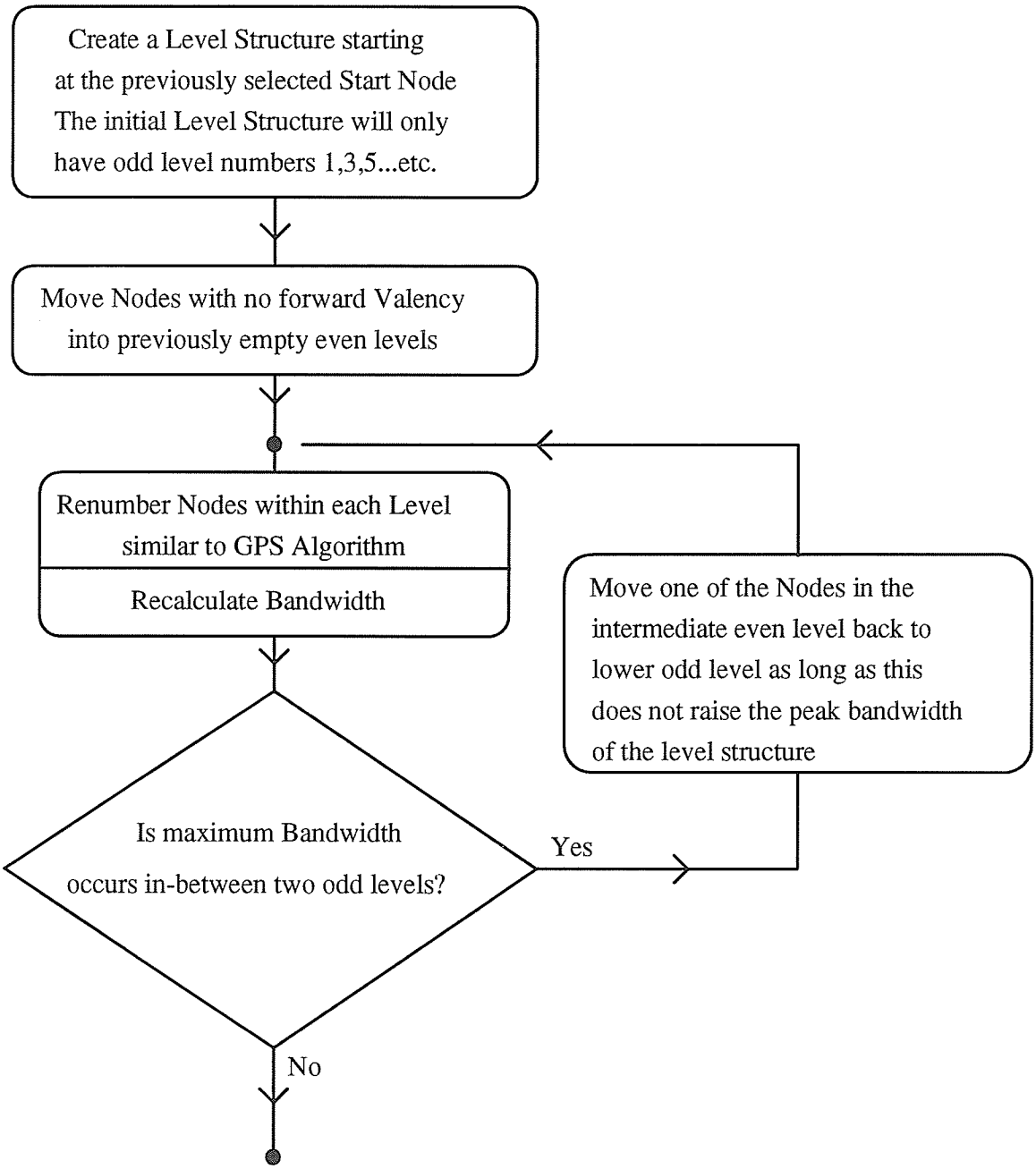


Figure 4.4 New Start Node selection method example

4.3.2 Hybrid Level Structure Creation and Node Renumbering

Hybrid level structure creation is done through a new level structure sorting method to simplify later operations. The new method draws on the merits of both the GPS and BL algorithms level structure and renumbering operations except for the part of GPS which is not possible due to only one start/end node selection. It can be seen that in both of the GPS and BL algorithms, much effort is expended in creating a minimum width path which is of course expected. After the level structure has been created, both algorithms renumber the nodes according to their own specific logic and then stop. Armstrong's algorithm continues at the end of GPS to use an iterative row column manipulation technique that generates the fine details of a good numbering order that graph theory methods just cannot produce. Due to Armstrong's success, an iterative reduction scheme will be added after initial number so the hybrid level structure creation and renumbering can be simpler than necessary in either GPS or BL. The hybrid method must still create a near optimum structure with good initial renumbering to insure that the iterative refinements do not stall at any local minima. In consideration of all the contributing factors, the new level structure creation and renumbering logic work as outline in *Figure 4.5*.



continued next page

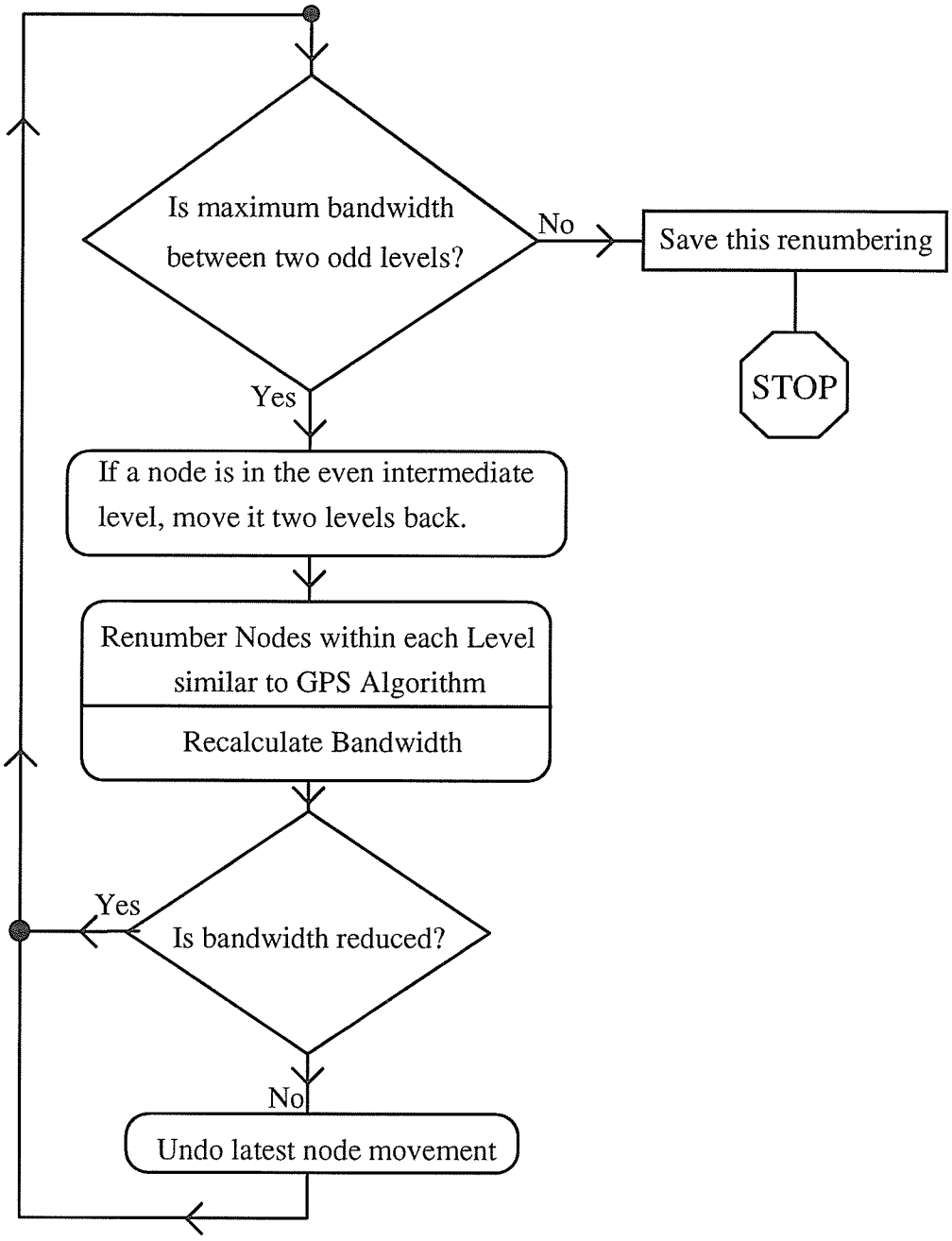


Figure 4.5 Level structure creation and node renumbering logic in LOOK algorithm

An advantage of the above method is that it simplifies the node movement logic compared to BL. The new method may also be easier to implement in an object oriented approach. Each level structure or node can be treated as an object and can be easily manipulated in this manner. This new method is not expected to be nearly as good as

Armstrong's results but should be superior to GPS if the starting node selected by this method is the same as that of the GPS method. The algorithm up to this point will be named LOOK as this represents an attempt to mimic the way in which a person looks at an entire structure or mesh. The LOOK algorithm attempts to mimic the way a person will determine the highest density area of a mesh, use that area as a center point, find a periphery, and then attempt to find the path of least level width through the structure starting from the periphery. The level smoothing also attempts to mimic the way in which a person has the ability to consider the effects of node movement on bandwidth in a level structure used for renumbering. It should be noted here that the LOOK algorithm is designed to work only with continuous structures. Five of the problems collected by Everstine are not continuous and could not be used for comparison.

4.3.3 Direct Reduction Component

The final stage of the reduction process will be a direct reduction component to fine tune the details necessary to approach the optimum node ordering. The direct reduction will be comparable to Armstrong's NSAS (node shuffling algorithm short) as it uses no jiggling components. The direct reduction component simply moves the nodes in-between the two nodes causing the bandwidth above or below these two nodes to create a reduction. The LOOK algorithm with the direct reduction component will be named the LOOK_DIRECT algorithm.

4.4. Numerical Comparison Results

These LOOK algorithm is a graph theory reduction method similar to GPS with level smoothing features similar to BL. A useful comparison to show the strengths and possible weaknesses of the LOOK algorithm would be a comparison of the LOOK results versus the GPS results for the *Everstine set.* The Everstine set is the group of 30 test problems collected from military and industrial sources used in his comparison paper[16]. The problems statistics are described in *Table 5*. Note that only the 25 problems used, as earlier explained, are listed.

Table 5 Test Problem Statistics

Number of nodes	File Number	Description	Original Bandwidth
59	6	2-D frame	26
66	15	Truss	45
72	12	Grillage	13
87	7	Tower	64
162	16	Plate with hole	157
193	17	Knee Prosthesis	63
209	10	Console	185
221	30	Hull-tank region	188
245	29	Tower with platform	49
307	20	Power supply housing	64
310	11	Hull with refinement	29
361	13	Cylinder with cap	51
419	24	Barge	357
492	26	Piston shaft	436
503	2	Baseplate	453
592	19	CVA bent	260
758	21	no description	201
869	14	no description	587
878	27	Plate with insert	520
918	4	Beam with cutouts	840
992	18	Mirror	514
1005	3	Baseplate	852
1007	22	no description	987
1242	9	Sea chest	937
2680	1	Destroyer	2500

The results of the comparison between LOOK and GPS are listed in *Table 6* below.

Table 6 Comparison of bandwidth results for GPS algorithm versus LOOK algorithm for Everstine test problem set

File Number	Halfbandwidth		Percentage Improvement
	GPS result	LOOK result	
6	9	8	11%
15	4	4	0%
12	7	7	0%
7	17	16	6%
16	14	19	-36%
17	43	47	-9%
10	43	32	34%
30	16	16	0%
29	40	59	-47%
20	41	40	2.5%
11	15	13	15%
13	15	15	0%
24	33	34	-3%
26	29	18	38%
2	66	50	25%
19	37	37	0%
21	27	26	4%
14	39	51	-31%
27	28	34	-21%
4	50	47	6%
18	36	49	-36%
3	108	77	29%
22	35	30	14%
9	100	87	13%
1	70	66	6%
Summation	922	882	4.3%

Figures 4.6 through 4.30 are reproductions of the plots from Everstines paper of the test problems.

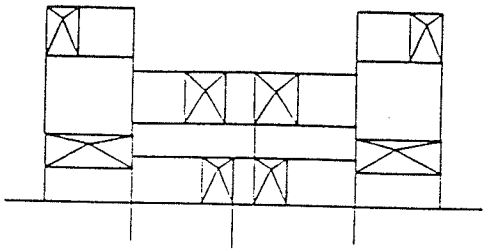


Figure 4.6 2-D frame

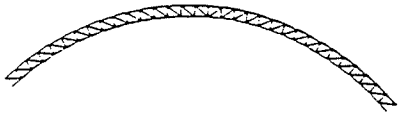


Figure 4.7 Truss

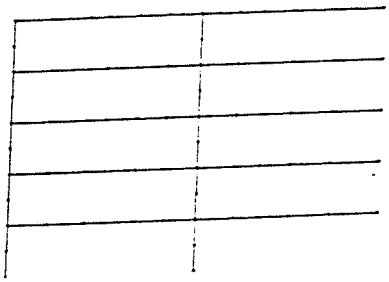


Figure 4.8 Grillage

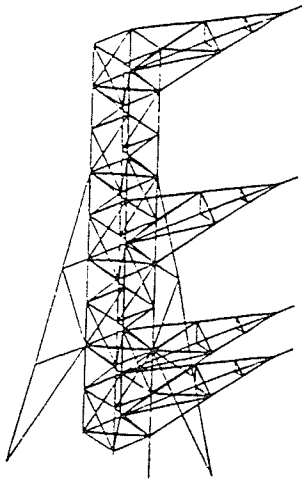


Figure 4.9 Tower

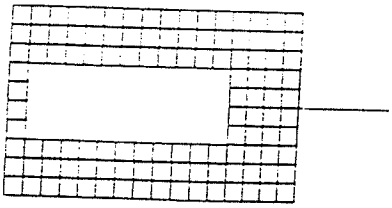


Figure 4.10 Plate with hole

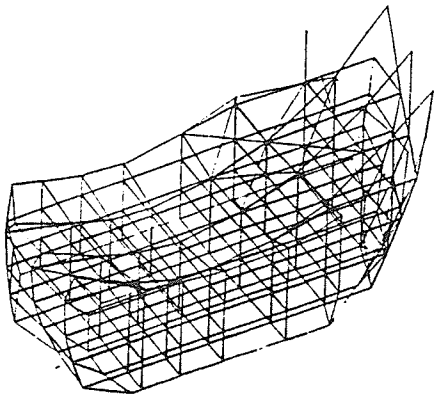


Figure 4.11 Knee prosthesis

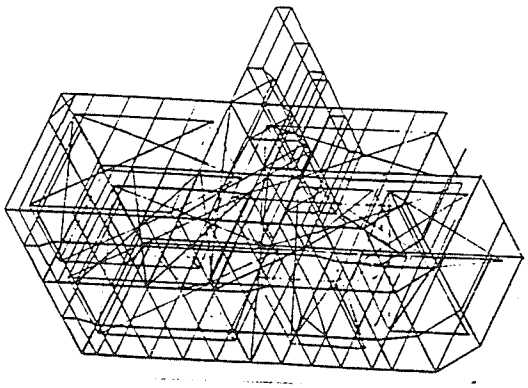


Figure 4.12 Console

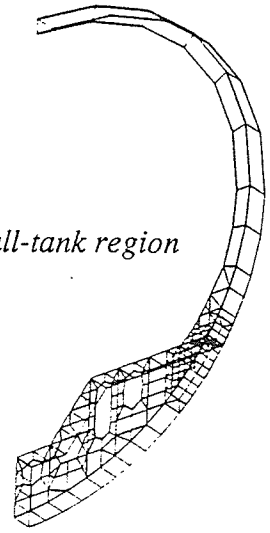


Figure 4.13 Hull-tank region

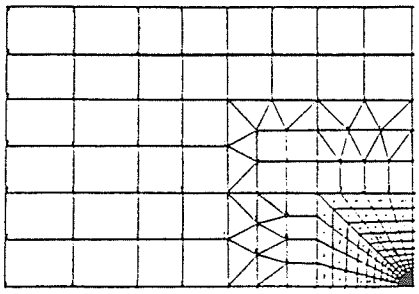


Figure 4.16 Hull with refinement

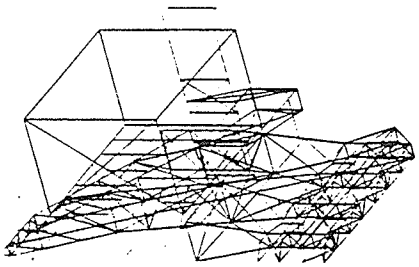


Figure 4.14 Tower with platform

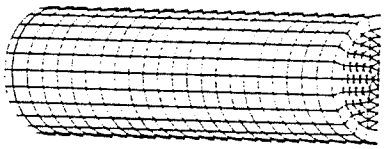


Figure 4.17 Cylinder with cap

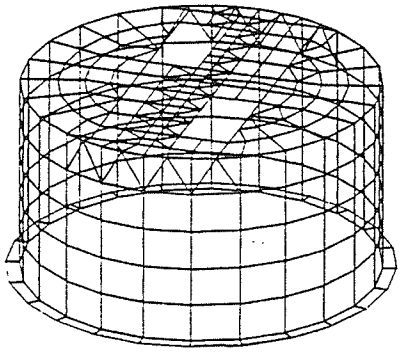


Figure 4.15 Power supply housing

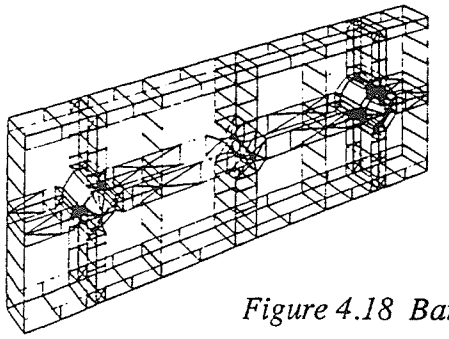


Figure 4.18 Barge

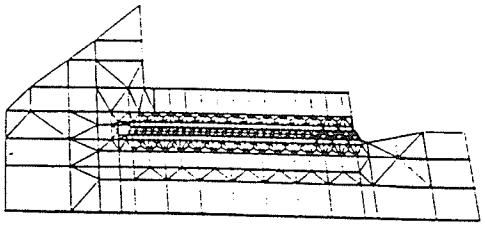


Figure 4.19 Piston shaft

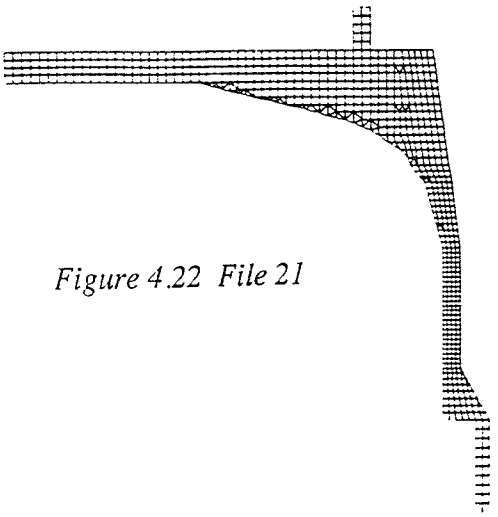


Figure 4.22 File 21

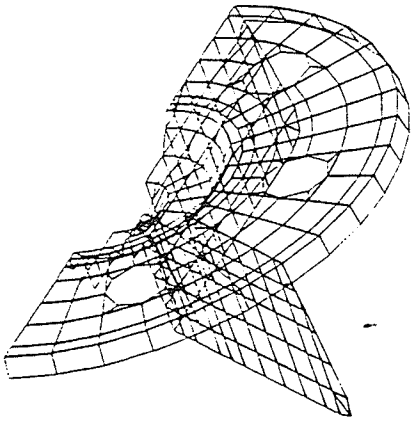


Figure 4.20 Baseplate

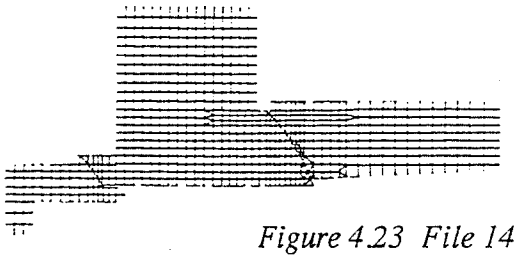


Figure 4.23 File 14

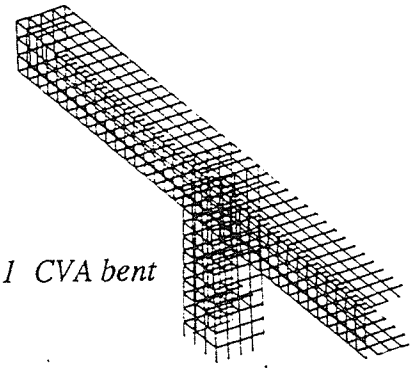


Figure 4.21 CVA bent

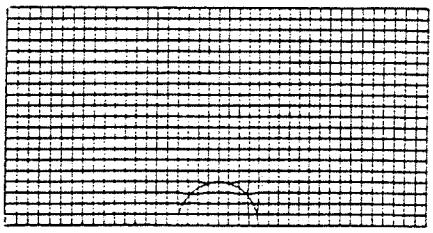


Figure 4.24 Plate with insert

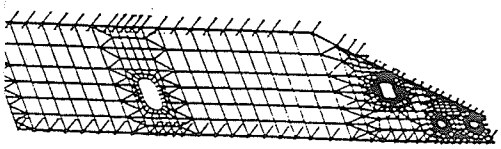


Figure 4.25 Beam with cutouts

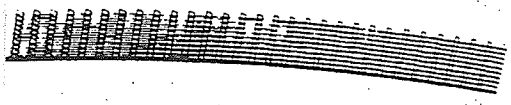


Figure 4.26 Mirror

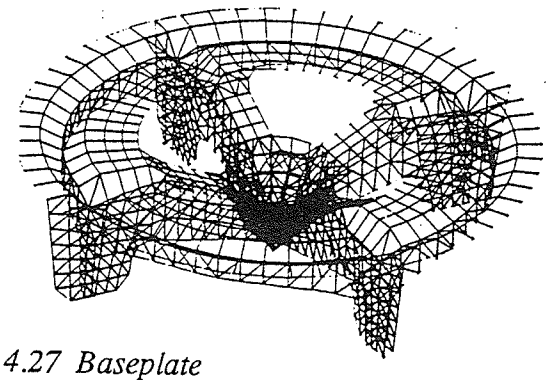


Figure 4.27 Baseplate

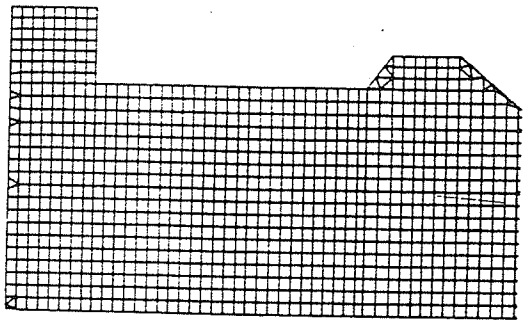


Figure 4.28 File 22

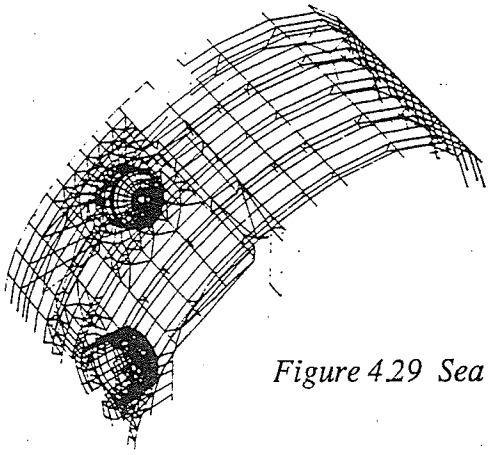


Figure 4.29 Sea chest

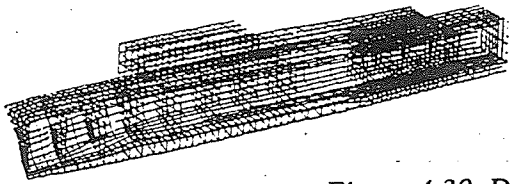


Figure 4.30 Destroyer

There are some problems that show much higher bandwidths than GPS and this, after some testing, was found not always to be a factor of the starting node selection, but the feature of GPS excluded in the LOOK algorithm. The GPS method, as discussed earlier, produces smaller level structures than possible with a single start level structure used in LOOK. This feature is felt to explain some of the higher bandwidths found in LOOK. Since some of the bandwidths are extremely higher in LOOK than GPS, the GPS starting node selection method was tested to see if that produced any further reductions to the LOOK results. The results, see *Table 7*, were a reduction in problem 29 from 59 to 34 and a reduction in problem 27 from 34 to 33. The remaining GPS advantages can be explain via the disjointed node positioning ability of GPS.

Table 7 Further Reductions in the LOOK algorithm using the GPS start node

File Number	LOOK start node	GPS start node
27	34	33
29	59	34

The results of the LOOK algorithm, even using both the start node selection methods, still lags behind even the direct reduction method of Armstong in most problems. It is important to note that 5 LOOK results tied the best yet found bandwidths and one case even produced a new reduction never before found. This can obviously be attributed to the new start node selection process. A useful comparison of the LOOK_DIRECT algorithm is the NSAS (Node Shuffling Algorithm Short) from Armstrong. In NSAS only a direct reduction scheme is used similar to LOOK_DIRECT. The Results are shown in *Table 8*.

Table 8 LOOK DIRECT vs. NSAS Bandwidth Results

File Number	Halfbandwidth		Percentage Improvement
	LOOK_DIRECT	NSAS	
6	7	8	12%
15	4	4	0%
12	8	7	-14%
7	15	14	-7%
16	17	14	-21%
17	43	40	-7%
10	31	40	23%
30	15	14	-7%
29	34	29	-17%
20	39	39	0%
11	13	14	7%
13	15	15	0%
24	33	31	-6%
26	18	25	28%
2	49	56	13%
19	37	34	-10%
21	25	27	7%
14	44	38	-16%
27	33	27	-22%
4	47	41	-14%
18	49	36	-36%
3	69	93	26%
22	30	34	12%
9	81	88	8%
1	65	65	0%
Summation	819	833	2%

4.5 Bandwidth Reduction Algorithm Development Summary

The bandwidth reduction algorithm present here, although not always creating the best possible reductions, reveals four important findings that should be used in future reduction algorithms.

1. The new 'humanistic' starting node selection method presented here normally selects equivalent or superior starting nodes than the GPS selection method.
2. The use of single start node level structures is in many cases incapable of producing good reductions as compared to the level structures created by the GPS method.
3. The odd-even-odd hybrid level structure presented here simplifies the iterative level structure minimization process and is easily coded.
4. Armstrong's jiggler method for iterative reduction, not used in this study, appears useful to find the finer details of bandwidth reduction.

The four findings here are not all unique, but do set down the guidelines for the further development of a hybrid bandwidth reduction algorithm. An improvement over the LOOK_DIRECT algorithm would include; both the GPS and 'humanistic' starting nodes selection, the GPS method of level structure creation, an odd-even-odd level structure for iterative level structure reduction, and a jiggler reduction scheme similar to Armstrong's.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The fluid modeling showed that the difference between the dynamic effects of gate closure and the static pressures acting on the gate were negligible for the range of gate closure operating speeds employed by Hydro. Thus, the maximum force acting on the gate can be assumed to be the hydrostatic pressure distribution acting on the fully closed gate. The hydrostatic force is applied to a sluice gate model and the main roller normal reaction forces determined. The bottom main roller reaction force is found to be 271 kips which is larger than previously determined by Hydro. The main roller misalignments caused by the deflection of the gate are found to be negligibly small, less than 10% of one degree.

The loading situation in which all the main rollers creating a lateral force of 0.27 their normal force is modeled and found to be unrealistic. To better model a realistic loading, a comparison of the deflections reported by the Hydro onsite inspection and selected loadings is performed. The selected loadings model arrives at a maximum lateral loading on the bottom side roller of 126 kips. The case of a broken lift cable shows lower forces than the selected loadings model and is therefore not of concern. The resulting stress is confined to the immediate area around the side roller support structure. The redesign of the side roller support structure, using 1.5" thick reinforcement steel produces lower stresses with a generous factor of safety. It is recommended that Manitoba Hydro consider the reinforcement design shown in this study as it not only addresses the side roller support structural safety concerns, but is also a simple and probably cost effective design.

A major limitation of the MHYFECS programs is the limited memory available on a PC for solving the large gate models as well as the inordinate amount of time it takes to solve a large problem. To address this concern, a hybrid bandwidth reduction algorithm is developed to minimize storage space and reduce solution times. The algorithm, named LOOK_DIRECT uses an improved start node selection method, and easier to implement level structure iterative reduction method, and a direct reduction component. The LOOK_DIRECT algorithm, on average, creates smaller final bandwidths than either the GPS or NSAS algorithms. The study performed here produced four recommendations on useful components of future hybrid reduction algorithms. They are; both the GPS and 'humanistic' starting nodes selection, the GPS method of level structure creation, an odd-even-odd level structure for iterative level structure reduction, and a jiggler reduction scheme similar to Armstrong's. The algorithm developed in this study shows the necessity of the second and fourth features, but also presents the original 'humanistic' starting node selection method and the method of odd-even-odd level structure for iteration.

5.2 Recommendations

The findings of this study reveal the possibility of unacceptable lateral forces developing in existing sluice gate installations. The reinforcement design presented herein addresses the existing situation by treating the symptom of the problem, lateral forces, not the cause of the problem, the development of lateral forces. It is recommended that future studies be undertaken to consider the redesign of the roller paths such that the main rollers are directly guided in their movement path. Such a roller path may be made of heavy channel or a combination of stock sections of steel. If the existing installations are to be maintained in their present configuration, the best use of resources may be to prevent the development of the lateral forces instead of reinforcing the gates to withstand high lateral loading. It can now be confidently stated that the lateral forces do not result from main roller misalignments caused from the bending of the gate. The major source of main roller misalignment is the initial misalignment from roller axle installation and the journal bearing clearance. If these can both maintained to a combined tolerance of 1 degree, the result would be a possible maximum of only 120 kips total lateral force. The ability to maintain this accuracy is doubtful so the original recommendation of a channel like roller path may be necessary.

REFERENCES

1. Chan, S. T. K., Larock, B. E., and Hermann, L. R., "Free-Surface Ideal Fluid Flows by Finite Elements", *Journal of the Hydralics Division,* ASCE, Vol. 99, No HY6, Proc. Paper 9787, June, 1973, pp. 959-974.
2. Diersch, H. J., Schirmer, A., Busch, K. F., "Analysis of Flows with Initially Unknown Discharge", *Journal of the Hydralics Division,* ASCE, Vol. 103, No. HY3, March, 1977, pp. 213-232.
3. Zienkiewicz, O. C., "The Finite Element Method in Engineering Science", McGraw-Hill, London, 1971.
4. Bathe, Klaus-Jürgen, "Finite Element Procedures in Engineering Analysis", Prentice-Hall, Englewood Cliffs, 1982.
5. Cook, R. D., "Concepts and Applications of Finite Element Analysis", John Wiley & Sons, Toronto, 1981.
6. Zienkiewicz, O. Co., and Taylor, R. L., "The Finite Element Method", McGraw-Hill, London, 1989.
7. Weaver, W., and Johnston, P. R., "Finite Elements for Structural Analysis", Prentice-Hall, Englewood Cliffs, 1984.
8. Manitoba Hydro Report (KGS) , File No. 89-038-01, July, 1991.
9. Cress, H. A., and Talbert, S. G., "Experimental Determination of Lateral Forces Developed by a Misaligned Steel Roller on a Steel Rail", *ASME Paper No. 63-WA-298*, January, 1964.
10. Grooms, H. R., "Algoithm for Matrix Bandwidth Reduction", *Journal of the Structural Division*, ASCE, Vol. 98, No. ST1, January, 1972, pp. 203-214.
11. Puttonen, J., "Simple and Effective Bandwidth Reduction Algorithm", *International Journal for Numerical Methods in Engineering*, Vol. 19, 1983, pp. 1139-1152.

12. Cuthill, E., and McKee, J. M., "Reduce the Bandwidth of Sparce Symmetric Matrices", *Proc. 24th National Conference*, Association for Computing Machinery, ACM Pub. P69, New York, 1969, pp. 157-172.

13. Gibbs, N. E., Poole, W. G., and Stockmeyer, P. K., "An Algorithm for Reducing the Bandwidth and Profile of a Sparce Matrix", *SIAM Journal of Numerical Analysis*,Vol. 13, 1976, pp. 236-250.

14. Burgess, I. W., and Lai, P. K. F., "A New Node Renumbering Algorithm for Bandwidth Reduction", *International Journal for Numerical Methods in Engineering*, Vol. 23, 1986, pp. 1693-1704.

15. Armstrong, B. A., "A Hybrid Algorithm for Reducing Matrix Bandwidth", *International Journal for Numerical Methods in Engineering*, Vol. 20, 1984, pp. 1929-1940.

16. Everstine, G. C., "A Cmparison of Three Resequencing Algorithms for the Reduction of Matrix Profile and Wavefront", *International Journal for Numerical Methods in Engineering*, Vol. 14, 1979, pp. 837-853.

17. Akhras, G., and Dhatt, G., "An Automatic Node Relabelling Scheme for Minimizing a Matrix or Network Bandwidth", *International Journal for Numerical Methods in Engineering*, Vol. 10, 1976, pp. 787-797.

18. Cheng, K. Y., "Minimizing the Bandwidth of a Sparce Symmetric Matrices", *Computing Journal*, Vol. 11, 1973, pp. 103-110.

Appendix A

MHYFECS USERS MANUAL

1. INTRODUCTION

1.1 What is MHYFECS ?

Welcome to MHYFECS, the Manitoba HYdro Finite Element Computer Simulation computer program. MHYFECS is a general purpose solid modelling stress analysis program written for linear elastic problems.

MHYFECS is used to model structures made of components such as plates, trusses, beams, and solid structures. The model of the structure is made of a finite number of components each known as an element, hence the name, finite element. Each element in the structure models the stiffness that its respective component adds to the total structural stiffness. When all of the structures stiffnesses are combined, the deflection behaviour of the whole structure can be modelled.

Since the finite element method is an approximate numerical method, the deflection behaviour is not calculated for every point in the structure. Each element is defined by a number of specific points in space known as nodes. The deflection is calculated at each node and any deflections between nodes is found from interpolation between the adjacent nodes. To link the elements together in the numerical model, the same node is used to define the end of one element and beginning of another. Since each node can only have one set of deflections, the elements that share this common node are said to be linked. In Figure One below, we can see two truss elements linked at node two but free at nodes one and three.

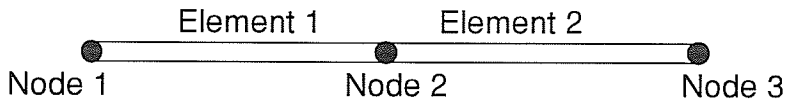


Figure One: Nodes and Elements

Once the total structural stiffness has been modelled, loads and restraints can be applied to the model. Loads are modelled simply by applying the desired load at a nodal location. All models must have some restraints or the structure would not resist the force and would have a rigid body motion. To restrain the model, we apply restraints known as boundary conditions. For example, if a beam was pinned at an end, the node at that end of the beam would not have a lateral or vertical deflection but can still exhibit a rotational deflection. More specific details are explained in the rest of the manual.

1.2 How do I use MHYFECS ?

MHYFECS is broken into three major sections known as the Input, Solver, and Output sections. Each section is a separate program that can be run independently or more easily by the main menu program included. This three section approach is useful in using all of the computers available memory in the DOS enviroment.

The Input section is used to generate the input data file for the solver section. In the Input section, the model is built from nodes, elements, then forces and boundary conditions.

The solver section is the program that solves the equations modelling the structure. This is the most complex part of the MHYFECS package and the most important. It calculates the displacements, element internal forces, reaction forces, and stresses.

The last main section of the program is the Output section. This section is used to graphically display the displacement of the structure, in reality or exaggeration, and display the stresses present in the structure. This section can also produce graphic outputs of the generated graphics.

The three main sections are described in detail in the following three sections of the manual. Also included in the manual is section six, a step by step example of how to generate a model, solve the problem, and display the output. After reading the manual and completing the example, you will be ready to use MHYFECS to solve your solid model stress analysis problems.

2. GETTING STARTED

This section is designed to show you how to load MHYFECS on your computer and run the Main Menu. After loading the program on your computer, it is suggested that you read the entire manual and do the example problem before attempting a real problem.

2.1 Loading MHYFECS on your machine.

To load MHYFECS onto your machine you must have a colour VGA monitor and graphics card, a Microsoft© compatible mouse, and at least 2.0 Mbytes free disk space for the executable codes. The program also requires more disk space for the virtual solver. If you have 8 Mbytes of memory, this can be overcome by using a ram disk.

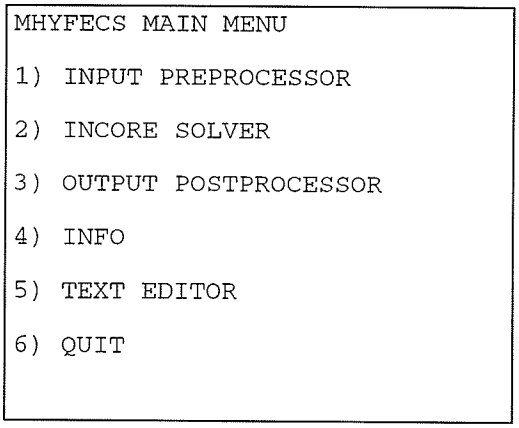
Start by putting the MHYFECS disk labelled INSTALL into your machine. At the floppy drive prompt, type install↵ (the symbol ↵ is used to denote the return or enter key in this manual). The installation procedure is automatic and needs no attentions. After about one minute all the files should be in place and ready to go.

The installed program has created a directory on your machine called MHYFECS. Three more subdirectories called INPUT, SOLVER, and OUTPUT have also been created. The information contained in the MHYFECS directory is the graphics drivers and the main menu program.

2.2 The Main Menu

To start the main menu, change to the MHYFECS directory and type MENU↵. The main menu will appear on the screen and your ready to try it out. The part of MHYFECS you wish to run is selected by using the ← and → keys to highlight the number in front of the program name and then pressing enter. The three main sections available are: 1) The INPUT program 2) The SOLVER program 3) The OUTPUT program. Also available is a call to the DOS Editor if on your machine. The use of each of these sections is shown in the example. Below is a simple drawing of the main menu.

Figure Two: Main Menu



3. Input

The INPUT routine is run either by the main menu or in the INPUT directory by typing INPUT↵ at the dos prompt. Once in Input, the program is menu driven and intuitive button names have been used.

The first part of the INPUT program is the setup variables. These are initial choices that tell the program what you want to do. The first choice is problem type, you can select 1 if your going to retrieve an existing data file, 2 if you wish to model a two dimensional problem, or 3 for a three dimensional model. Next, you can choose whether or not you wish to include the forces of gravity in your model, and lastly you can select the type of marker for nodes you wish to see. The program will ask if these choices you have made are OK, and if you responde y, you will begin the rest of the program. If you have made a mistake in selecting, respond n at the check request and you can enter all three choices again. Now you can start your modelling procedure.

Note: The gravity function has not yet been fully implemented. *Do not use!*

3.1 Nodes

Before creating any elements, applying any loads or forces, the nodes must be created. Nodes are the points in space that are used to define the physical geometry of the problem, and the points at which the deflections and stresses are calculated at. They are then used to define the elements that make up the model of the structure and used as places to apply loads and boundary conditions.

To create a node, select the button named NODE. The buttons below will now have functions printed on them which correspond to node functions. To create a node, select the button named CREATE NODE. The program will now prompt you for the coordinates of then node. If you didn't select 2 or 3 for the problem type in the setup routine, an error message will tell you to first select the problem type. This is done with the SELECT DIMENSIONS button in the FILE routine. Once the coordinate system is set,, a right handed rule X-Y-Z system is used. The created node will automatically be numbered as the number of nodes plus one.

If you typed in the wrong coordinates, it's ok, because the move node routine, selected by the MOVE NODE button allows you to reenter the coordinates.

If you have created too many nodes you may wish to delete some nodes. This is accomplished by using the DELETE NODE button. The prompt will now ask for the number of the node you wish to delete. You may also pick the node to delete with the mouse. This is done by first selecting the MOUSE button at the bottom of the menu before selecting DELETE NODE from the menu. To return to using keyboard input, select the KEYBOARD button instead.

To speed the tedius task of entering nodal coordinates, a special function, DUPLICATE NODES has been included. This feature allows existing node patterns to be duplicated at specific distance from the original. Select the DUPLICATE NODES button and you are prompted for the starting node, the end node, the increment, the X distance, the Y distance, and possibly the Z distance. The start node is the node number of the first node in the series you wish to duplicate and the end node is the last. You may sometimes wish to only duplicate every second node, so you can enter the increment as 2. Normally, you will want all the nodes, so just enter 1. The pattern is copied in space at the X,Y,Z distances away you now specify. The new nodes will be in same numeric order but will start at the former number of nodes plus one.

To view the nodes, exit the node routine by selecting DONE. Now select the VIEW ALL icon and all the nodes will appear in the graphic box. To turn the node numbers on or off, select the NODE# button and then REDRAW. If you wish to see one section of the graph in more detail, you can use the ZOOM IN button to select a small viewing area, or ZOOM OUT to see twice as much of the graph. To rotate your view in space, the < ^ v > rotation buttons can be selected. If you have rotated the view to a confusing viewing angle, you can select the RESET ANGLE button to return to the default XY plane view. Later on you may wish to not view the nodes so, you can select the NODE 0/1 button to either activate viewing of the nodes or hide them from sight in the next drawing.

3.2 Groups and Elements

Many elements may be used to model a structure and many elements may have the same properties such as, moments of inertia, elasticity, etc. To store the information about each element more efficiently, the elements are grouped together by type and material properties. Before creating an element, a group must be created to describe the elements in that group. To create elements and groups, select the ELEM button. The blank buttons below will now have functions printed on them for group and element functions. The CREATE GROUP button will let you create a group describing a type of element with specific properties. Before creating a group, first try the DEMO button. This button simply gives an online plot of all the available elements. Now select the CREATE GROUP button and the program will now prompt you for the element type number. The element details are listed in the appendix.

To create, for example, a 3D truss, select number 5. The program will now prompt you for the two material properties, elasticity and the cross section area of the member. If you have made a mistake in entering the material properties, it's ok. The MODIFY GROUP button can be used to modify the material properties of a group. The program will prompt you for the group number you wish to modify and then ask you to reenter the material properties. Note: Changing the element type after a group has been defined is *not* allowed. All other element groups are created this way and more detailed information on elements is given in the appendix.

Once groups are created, one specific group must be selected to create elements of that type. This is accomplished by selecting the SELECT GROUP buttons. Type in the group number from which you wish to create an element and then select CREATE ELEMENT. The program will now prompt you for the node numbers of the nodes used to create the element. Like the node functions, you could first select the MOUSE button so that instead of entering the node numbers to create the element, you would simply select them with the mouse. If you make a mistake, it is ok. You can modify the elements you have made by using the MODIFY ELEMENT button but, make sure you have redrawn the elements with the element numbering on so you know which element to modify. If you make too many elements or the wrong kind, you can always delete a wrong element by using the DELETE ELEMENT button. The delete element routine will ask you for the number of the element to be deleted.

The viewing of the elements is the same as the viewing of the nodes with a ELEM# button for turning the numbering on/off and an ELEM 0/1 for viewing the elements or not. Another feature is the REDRAW button. After using other view manipulation functions, you may just want to redraw the view without changing it. The REDRAW button does just that, it redraws the screen. Just like nodes and elements, groups viewing can also be turned on and off using the GROUP ON/OFF button to turn on or off the presently selected group. This is sometimes helpful to see the internal structures of a model by turning off the outer covering elements. Remember to select DONE to return to the main menu when finished in the element routine.

3.3 Loads

Once you have created the model of a structure, you will obviously want to apply some loading to it. This is accomplished by use of the APPLY FORCE button in the node routine. The program will prompt you for the number of the node on which you wish to apply a force, or if MOUSE was selected, you can pick the node to load with the mouse. The program then asks which degree of freedom you wish to load. You can responde X, Y, Z, MX, MY, or MZ. These represent linear forces in their respective directions and the moments about their axis. To view the forces, use the FORCES ON/OFF button. On the next redraw the forces will show as solid yellow lines and the moments as dotted purple lines in their respective directions.

If you notice an error, you can delete an applied force by using the DELETE FORCE button which prompts you for the node, or selection with the mouse, and dof to unload. If all you have done is type in the wrong value of force, you can redefine the force on a specific node and dof by simply using APPLY FORCE on the same node and dof again.

3.4 Boundary Conditions

Now that you have created and loaded your model, you must fix the model so as to represent its real life supports and restraints. Depending on what element you are

using, only certain degrees of freedom will exist and fixing nonexistent dofs has no effect on the solution. The dofs are defined as the linear displacements in the X, Y, and Z directions, and RX, RY, RZ rotations about the x, y, and z axis respectively.

To model a support under a structure, you would probably want to restrain the y movement, and the x and z axial rotations. To do this, select APPLY BC. The program will now prompt you for the node at which you wish to apply a boundary condition, or selection with the mouse. Select the node and then the computer will prompt you for the dof to fix, enter Y to fix the y displacement. Repeat this for dofs RX, the x rotation, and RZ, the z rotation.

After applying boundary conditions, you can view them by selecting the BC ON/OFF button and then REDRAW. The boundary conditions are graphically displayed as DX,DY,DZ,RX,RY,RZ for each fixed dof respectively beside their corresponding node.

If you notice an error, you can remove a boundary condition simply by selecting the DELETE BC button. The program will ask you for the number of the node, or selection with the mouse, and then the dof to free.

3.5 Lists

Sometime during your modelling procedure, you may want to review nodal locations, material properties, etc. To do this, you can use the LIST button. The program will show a menu containing NODES, ELEMENTS, GROUPS, MATERIALS, FORCES, and BC. By selecting the topic you wish to view, the information is displayed 15 lines at a time in the graphic box area. Select DONE when finished to return to the main menu. The last graphic plot will automatically be redrawn.

3.6 File Functions

Once models are created, they are saved as files using the functions in the FILE menu. The file that is saved is an ASCII file that is used in the solver section of MHYFECS and can be read back into INPUT. Before saving a file for the solver, you must check the file by selecting the CHECK button. The check routine will do a number of operations such as renumbering of nodes and elements, checking for repeated nodal positions, checking for repeated nodes in elements, missing forces and boundary conditions, etc. The renumbering of the nodes and elements is optional as you may wish to leave the node or element number pattern constant to ease understanding of results even though several intermediate nodes or elements may have been deleted. The errors, if any, will be written to the file WARNINGS.OUT. There is more details on errors in section 2.8. The gravity modelling, if you selected it at the start of INPUT, is done in check. *The gravity modelling must only be done once your model is completed and ready for solving!*

The files that INPUT generates and reads are by default in the directory in which INPUT resides. To change this, use the PATH function. This function prompts you for a new path for reading and writing files to. The default is . and this should be typed in to return to the default directory. An example would be **b:\data** to read and write from the data directory in b drive.

Once the file has been checked and found to have no errors, and you have set the desired path, you can save the file by using the SAVE AS button. The program will prompt you for a file name (8 characters max . 3 extension) and then save the file as that file name. If later on you change the model and wish to save your changes, you simply select SAVE and the program will save the model under the present file name. The program may sometimes ask you to reenter the path so as to ensure safe data storage.

The files that are saved are not only formatted for the solver section, but can be read back into INPUT for more model building by selecting OPEN. The program will then print a list of all the data files in the INPUT directory with the extension **.dat**. For this reason it is recommended that you save models as **.dat** files in the input directory. Next, type in the filename you wish to open.The file will be read in and will be drawn on the screen. If you are done with a file and wish to create a new file, select the NEW button. This will erase all of the data in memory and change the title to untitled.

Also in the FILE menu is the SELECT DIMENSIONS button. This lets you select the number of dimensions you wish to model. This is necessary if you selected NEW or did not enter dimensions in the starting of INPUT.

The last function in the FILE menu is the QUIT button. This is how you exit the program. If you have changed the file since you have last saved it, the program will prompt you whether you want to quit with out saving the new changes. If you do wish to quit without saving, respond y, else respond n, save the file, then quit.

Another file that can be created is a **Computer Graphics Metafile**. This allows the onscreen plot to be redirected to a file for later manipulation and/or printing. The desired plot must first generated on the screen and then the META FILE button is selected. The resulting data is sent to the file METAFILE.DAT in the INPUT directory. The file can then be renamed to an appropriate name (for example - girders.cgm) and used with most commercial drawing packages. The **.cgm** files are readily imported by both **Word for Windows**© and **CorelDraw!**©. One note, sometimes the data transfer isn't that smooth and the drawing programs initial imported picture is quite different from the onscreen look. It is suggested that you ungroup the drawing once imported and delete the background. After that, regroup the remaining drawing so you can scale the drawing as you wish.

3.7 Data File Structure

The files that are generated are standard ASCII text files so that they can be user manipulated. The data files also have end of line comments to ease understanding. The following page outlines their structure:

first line (control data)
- -number of nodes (user defined)
- -number of actual nodes (internal)
- -number of elements (user defined)
- -number of actual elements (internal)
- -number of dimensions
- -number of groups
- -number of nodes with boundary conditions
- -number of materials (same as number of groups)
- -1 (future solver control function)

list of nodal numbers and coordinates

_ element type, number of elements in group, group number
_ list of element numbers and node numbers in element ,in group
_repeated number of groups

list of nodes with b.c. and there fixivity (0=free , 1=fix, in dof order)

_ material number, number of properties
_ list of material properties
_repeated number of materials

elastic foundation constant (0.0 for not used)

number of loaded nodes (all nodes) 0 (future load functions)

list of all nodes with their force values (in dof order)

This file can be edit by the user provided they **knows what they are doing**. The file is documented with words at the end of each data line. An example of this file is given in the example section of the manual.

3.8 Error Control

One of the user friendly features of MHYFECS is the error control system. As stated earlier, the CHECK function checks your model for completeness and for repetition errors. The errors, if any are written to the DOS file, WARNINGS.OUT. Also written to WARNINGS.OUT is certain control data you can look at to ensure that the

model is consistent with what you wish to produce. After some practice you will probably never need to look at this file but it is of great help while learning. Let's take a look at the file from a model with no errors on the next page.

```
NUMBER ACTUAL NODES                21
NUMBER OF FORCES                    1
NUMBER OF BOUNDARY CONDITIONS       8
NUMBER ACTUAL ELEMS                23
```

The above data tells us that the total number of nodes in the model is 21. You may have created 30 nodes, then deleted 9. The check routine can renumber the nodes to make the storage of the data file as compact as possible. This compaction is also the case with the elements. The number of forces and boundary conditions are also printed out to ensure that your loading and restraining is as you planned it to be. Now let's look at some possible error messages from WARNINGS.OUT

```
************************************
******   WARNING!!  ERROR!!    ******
******                         ******
****** REPEATED POSITIONS FOR ******
****** NODES       1      5   ******
************************************
```

This is an example of an error message generated when two nodes have accidentally been assigned the same position in space. This error must usually be correct but sometimes two separate nodes at the same point in space. This could be used for modelling the very tip of a crack or another unjoined opening.

```
************************************
******   WARNING!!  ERROR!!    ******
******                         ******
******         NO FORCES       ******
************************************
```

This is the warning generated when there is no loading on the model. Since there is no use in a model without a physical load, this error must be corrected.

```
************************************
******   WARNING!!  ERROR!!    ******
******                         ******
****** NO BOUNDARY CONDITIONS ******
************************************
```

This is the warning generated when there are no restraints on the model. Since the model must be restrained, this error must be corrected.

```
**********************************
******    WARNING!!  ERROR!!    ******
******                          ******
******     REPEATED NODE IN     ******
******     ELEMENT         1    ******
**********************************
```

This is an example of an error message generated when an element has accidentally been assigned the same node twice. This error must usually be corrected as the element stiffness can not be properly generated with repeated node numbers. Some future elements may allow repeated nodes to reduce their shape from a quadrilateral to a triangle or the like.

There is also much online error control. For example, if at any time you enter a wrong value or impossible selection, the program will display a red error message and prompt you to continue.

Remember, the program does what it is told to do, if it appears to do otherwise, it is 99% likely to be something small that you have overlooked. If the program flat out crashes or hangs, check your data for anything that is not exactly standard. Although the program is designed to be robust and will normally display an error message when reading in bad data files, some data may be read into memory and do weird "*mystical*" things. Try NEW or rerunning the INPUT program to correct these "*mystical*" occurrences. If the graphicly written words in the menus appear to be "*disintigrating*", that is a sign that saving twice and rerunning the program might be very wise. (DO IT!)

3.9 Units

One important point to remember when using finite element modelling is to be consistent in your use of units. You must be very careful to define all of the nodal positions in the same units. Once this is done you must also insure the element properties are entered in the same units for such details as elasticity, moments of inertia, etc. The forces and moments applied to the model must also be kept consistent with respect to the rest of the model. On the next page is Table One which demonstrates consistent units in both the english and metric systems.

Table One. Unit Consistency

QUANTITY	ENGLISH		METRIC	
	UNITS	NAME	UNITS	NAME
FORCE	lb	POUND	N	NEWTON
MOMENT	lb·in	POUND INCH	N·m	NEWTON METER
LENGTH	in	INCHES	m	METER
MOMENT OF INERTIA	in^4	$INCH^4$	m^4	$METER^4$
MODULUS OF ELASTICITY	lb/in^2	POUNDS PER SQUARE INCH	N/m^2 or Pascal	NEWTON PER SQUARE METER
STRESS	lb/in^2	POUNDS PER SQUARE INCH	N/m^2 or Pascal	NEWTON PER SQUARE METER

4. SOLVER

4.1 Theory

The MHYFECS SOLVER is based on the stiffness-displacement method better described by the equation [Kg]{u}={f}. This is the basis of all structural finite element programs in that the stiffness · displacement = force. The stiffness is the combined stiffness of all the elements which is known as the global stiffness matrix [Kg], the displacements are the nodal displacement vector {u}, and the force is the global force vector {f}.

The global stiffness matrix is made by combining the local stiffness matrices together. For example, a 3D truss has 3 degrees of freedom at each end. If two trusses are joined at a common node, the 3 dofs at the first truss end are concurrent with the 3 dofs at the beginning of the other truss. This would result in the stiffnesses for the first truss and second truss being added together at the concurrent node in the global stiffness matrix. A problem occurs when the two elements are not inline with the coordinate axis. The stiffness of the trusses, or beams, or plates must be rotated to the global coordinate system. This is accomplished through the use of rotation matrices calculated for each element. Once an element is rotated to the global coordinate system, it can be added to the global stiffness matrix. This process of rotating the local stiffness matrix to global coordinates and adding it to the global stiffness is known as '**assembling the global stiffness matrix**'. The generation of the local stiffnesses and assembling of them is a computationally expensive process which on a large problem can takes a few minutes on a workstation and much longer on a PC.

The force vector that causes the structure to displace is also in global coordinates since the global stiffness matrix is. It is also easier for a user to apply global coordinate forces to the structure since the global coordinates are thr real world coordinates. The forces are either linear forces at nodes for linear dofs or moments for rotational dofs.

The solution for the {u} vector is done by Gauss-Jordan elimination in which all the elements of the matrix below the major diagonal are set to zero after which back-substitution is used to find the solution.

Once the global displacements are found, the global displacements can be rotated into local coordinates to find the elemental internal forces, the reaction forces at fixed nodes, and finally the stresses. The internal forces are found simply by:

$$[K_{local}] \cdot \{ U_{local} \} = \{ F_{internal} \}$$

If one of the internal forces is in a fixed dof, it is then a reaction force on the supports of the structure.

The stresses are found from the relationship:

$$\{ \sigma \} = [E] \cdot \{ \varepsilon \} \text{ where } \sigma\text{=stress vector and } \varepsilon\text{=strain vector}$$
$$[E] = \text{stress strain relationship matrix}$$
$$\text{where the strains come from: } \{ \varepsilon \} = [B] \cdot \{ u \}$$
$$\text{where } \{ u \} = \text{local nodal displacements}$$
$$\text{and } [B] = \text{part of the local stiffness}$$

For some simpler elements, the stresses can be found from internal forces. For example, the axial force F in a truss gives the stress from the equation:

$$\sigma = F / A \text{ where } A = \text{member cross sectional area}$$

In the above equations, the stresses are found at each node and stresses in between nodes are found using the elemental shape functions as interpolation functions.

4.2 Program Structure

The program is a modular multi-subroutine program using dynamic memory allocation. The modular format of the program allows new subroutines for new functions or elements to be added with only the addition of the subroutine code and a calling statement where necessary. The use of dynamic memory allocation allows previously used but no longer necessary data to be overwritten with new data without using any more memory. These features are only of interest to people modifying the main code, most people will be interested in the use of the code as outlined in the next section. As of June 1992, the virtual solver has been developed for RAMdisk solutions of problems upto 4 Megs of memory or even larger hard-disk solutions. The only problem with virtual solutions is the very slow solution process that can stretch from hours to over a day. Expected date of completion is September 1992 along with the bandwidth minimization code. The bandwidth minimization code will allow minimum memory usage so that a smaller matrix can be solved by the fastest solution method available.

People interested in modifying the main code should look at Klaus-Jurgen's base program STAP in his book on Finite Elements as its skeleton structure is almost identical to MHYFECS.

4.3 Program Usage

The SOLVER program is run either through the main menu or by typing solver↵ at the dos prompt. The program will open up a message box on the bottom of the screen. In this box will appear all the prompts and messages generated in the solver

process. The first thing the program does is prompt you for the name of the data file you wish to solve. An example is DEMO.DAT, the program will then automatically name the detailed output file to DEMO.OUT, and name the graphic output file to DEMO.STS. The detailed output is used by people programming and is not generally of interest to normal users unless errors occur. The graphic output file is used as input for the OUTPUT program to show displacements and stresses.

The first message is 'READING IN DATA'. If there are any errors in the data, the program will print an error message describing the problem in the data. An example of a data error message is ' ERROR OCCURRED WHILE READING COORDINATES ' which will be printed in yellow with a red background. This means that your nodal coordinates are incorrect in the data file or you have inconsistent control data (the first line of the data file) .

The program then writes the maximum dynamic memory allocated to the SOLVER program. If the end of the memory used is greater than that allocated, the program prints in yellow on red 'PROBLEM TOO LARGE FOR MEMORY' and stops. The allocation must be increased in the code and recompiled if possible. If the problem does not fit into the maximum possible sized PC based version, either a UNIX workstation version or the future virtual version must be used. If the problem does fit, a green message, 'PROBLEM FITS IN MEMORY' is printed.

The program then writes 'CREATING LOCAL STIFFNESSES' , '<== PRESENT ELEMENT' and echoes the present element number. If an error occurs during local stiffness creation it most likely not caused by the code but by the data being used for the element creation. The most probable causes are repeated node numbers in connectivity or zero property values. This kind of error, and NDP math or FORTRAN error writes an unformatted error message to the screen. The program has failed and you will return to the main menu upon the next keystroke.

After the local stiffness creation, the program begins to assemble the global stiffness matrix. This involves rotating the local matrices into global coordinates and adding the resultants into the global matrix if that global dof is not restrained. Any errors here should be reported to the software contact person as this is an extremely unusual case and should be looked into.

The next step is the loading of the model. The loads are now read from the data file. Errors here are normally an error in the input data file and not in the code. An onscreen message, 'LOADING' will appear.

After the loading, the 'INCORE SOLVER WORKING' message is displayed as the Gauss-Jordan elimination and back substitution takes place. The normal error message displayed here is 'NON-POSITIVE PIVOT' which means an ill-conditioned global stiffness matrix exists. This is a sign of poor local stiffness matrices and hence the reasons outlined in that section above are suspected. This could also be a sign of a

lack of proper boundary conditions which is easily solved by adding more restraints to the model.

Once the global matrix is solved, the displacements are written, the internal forces and stresses are calculated, and lastly the reaction forces found. All of the output is written to the detailed output file which can be viewed in any text editor. The graphic output file contains the model information, the displacements, and the stresses to be displayed graphically in the OUTPUT program.

5. OUTPUT

5.1 Introduction

The OUTPUT program can be run either from the main menu or from the DOS prompt by output↵. The main menu and the command line both execute the batch file output.bat which loads the graphics drivers, list the .STS files to a data file, then actually runs the output program OUTSHOW.EXE.

5.2 File Functions

The OUTPUT program only has two file functions. The functions are OPEN and PATH. The PATH function is the same as the INPUT program's PATH command. Since the SOLVER generates the .STS in its own directory, you must either copy them to the OUTPUT directory or enter the path as **..\solver**. The other function is OPEN that is used to read in the .STS files. Only .STS files can be read into the OUTPUT program.

5.3 Displacement Functions

The model, once read in, can be displayed four different ways. The view manipulation buttons at the top of the menu work with the undisplaced model in either wire-frame or solid shading. To view a displaced shape, you must select the DISP button.

The WIRE FRAME button will prompt you for an exaggeration factor. This factor multiplies the displacement for easier viewing. For example, 1.0 will show the true displacement, 100.0 will show 100 times the displacement, and 0 will show the undisplaced shape. The WIRE FRAME displacement will be a color wire frame drawing.

The SOLID DISP button does the same as above but this time the drawing is a color solid model.

The SUPER IMPOSED button is the most useful of the viewing functions. It allows you to create a color solid drawing of the undisplaced shape with the exaggerated displaced shape superimposed in black and white dotted lines.

You may want to turn certain groups on and off so that you can better see parts of your model. This can be done with the GROUP ON/OFF button. It will prompt you for the group number to show or hide.

A graphics meta file can be created by selecting the META button. You can enter a title for the plot.

5.4 Stress Functions

To view the stresses present in the structure, select the STRESS button. The stress menu contains the six primary stresses, Sx Sy Sz Sxy Syz Szx, and the criteria by Von-Mises for equivalent stress Seq. Selection of these buttons causes the stress contours to be plotted on the model in the last displaced shape viewed.

After a contour plot has been created, you can save it to a graphics meta file with the META button. You will be prompted for a title, and then either color or grey scale. Grey scale is normally preferable unless you are going to use a color printer or create slides from your drawing program. Remember that only on meta file can be generate per execution of the program and you must rename the metafile.dat file to something appropriate like STRESSES.CGM before rerunning OUTPUT or it will be lost.

Examples of all the above sections are in the following section.

6. EXAMPLE PROBLEM

6.1 Introduction

The problem we will model here is relatively simple but will show all the aspects of the programs. The problem is that of a cantilever beam with a vertical tip load. We will show how through mesh refinement progressively better answers can be found. Remember, all finite element meshes should converge to the theoretical answers with mesh refinement. Below is a drawing of the real life physical problem.

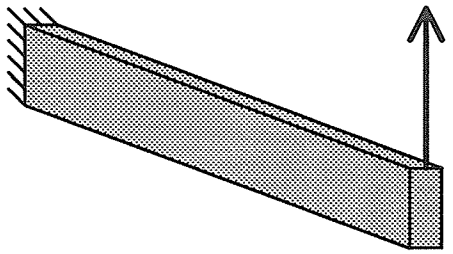


Figure Three: Physical Problem

We will use a length of 120, height of 10, width of 4, an elasticity of 1,000,000, and a vertical tip load of 100. Notice that this problem could be in any units as long as we keep consistency intact. Next we will develop this model in the INPUT program. You should now run the INPUT from either the main menu or the DOS prompt.

6.2 Input

The first thing to do is enter the start up information. We will enter 3 for a three dimensional problem, N to gravity, 1 to 4 for your choice of nodes, and Y if these choices are acceptable.

Next we will enter the nodal positions. Enter the nodal menu by selecting NODE. We will choose x=0 for the fixed end so, enter:

node 1 at 0, 0, 0
node 2 at 0, 5, 0
node 3 at 0, 10, 0

To view these nodes select NODE# then VIEW ALL. For the first model we can use just one element so we must generate the nodes at the end of the beam. To save work, we will use the DUPLICATE NODES button. We will enter:

1, 3, 1, 120.0, 0, 0

We entered nodes 1 to 3 incrementing by 1 to get all of 1,2,3. Then we entered the increments for x,y,z. They are 120, 0, 0 which is simply down the length of the beam.

We must now create an element to join the nodes. To create an element, we must first create a group. The element we wish to use is #6, 3D BEAM, so we will exit the node menu by selecting DONE, enter the element menu with ELEM, then select CREATE GROUP. The program will prompt you for the element type of this group, enter 6. You must now enter the material properties. It is a good idea to have all of these ready ahead of time. We will enter:

elasticity, 1000000.0
Bulk Modulus, 384600.0
cross section area, 40.
Cxx maximum, 5.
Cyy maximum, 2.
Ix (twisting), 1.0 (not important so why bother calculate ? , **but don't enter 0!**)
Iy (minor bending), 53.33
Iz (major bending), 333.33
ρ (density), 1.0

After creating the group, you must then select that group to tell the program from which group you wish to create elements. So, select SELECT GROUP and enter 1. You can now create the element. If you wish to simply enter the node numbers, select KEYBOARD or else select MOUSE. KEYBOARD is the default. If you are going to use MOUSE, insure that all the nodes you require are visible on the screen. Now select CREATE ELEMENT and enter:

1 2 3 4 5 6

Yikes! The element is wrong! Notice that the element is twisted. This is because the natural ordering for the element is a top node, middle node, bottom node ---> other end ---> bottom node, middle node, top node. We must correct this error by selecting MODIFY ELEMENT and then entering:

3 2 1 4 5 6

The element is now correct. Note again the correct connectivity. This is **very** important.

Next is the application of the load and boundary conditions. Select DONE to leave the element menu and NODE to return to the NODE menu. Now select APPLY FORCE and enter:

5 y 100.0

We entered 5 for node # 5, y for the dof to load, and 100.0 for the force value. As you can see, this would get rather tedious for large models. In the future, a loading feature that can distribute pressures and such may be developed. This is also similar to the gravity function presently being developed.

Now select APPLY BC and enter:

1 a

again APPLY BC

2 a

and again APPLY BC

3 a

We entered all three fixed node #'s and fixed "a" for all dofs. This applying of boundary conditions could also be tedious so a fixed or symmetric plane function is also envisioned for a future version.

Out model is now complete and ready for testing. Select DONE to exit the nodal menu. You might now want to turn on the force and bc viewing to look at the complete model. Rotated it around a bit and look at it. To check the model, select FILE, and then CHECK. Since no nodes or elements have been deleted, it is not necessary to compress either but we can do it anyway, enter:

y y

We have compressed to nodes and elements (well not really since they are already consecutively numbered from 1) and check the rest of the data. You should see the message in green "NO ERRORS FOUND". Now before saving the file, let's

select the path one more time as . using PATH just to be sure. Now use SAVE AS to save the file as, for example, DEMO.DAT. "save" should appear on the screen right of DEMO.DAT to indicate it was saved. We could also set the path to "..\solver" to save it in the solver directory.

We are now ready to solve the problem modeled here with the SOLVER program. Now select QUIT.

6.3 Datafile

The datafile is relatively simple for this problem, so let's take a look. The datafile is listed on the next page.

```
6      6      1      1      3      1      3      1      1  #NODES<U&A>#ELEM #DIM #GROUP #BC #MAT
       1    0.000      0.000     0.000      NODE#   POSITIONS X Y Z
       2    0.000      5.000     0.000      NODE#   POSITIONS X Y Z
       3    0.000     10.000     0.000      NODE#   POSITIONS X Y Z
       4  120.000      0.000     0.000      NODE#   POSITIONS X Y Z
       5  120.000      5.000     0.000      NODE#   POSITIONS X Y Z
       6  120.000     10.000     0.000      NODE#   POSITIONS X Y Z
       6       1      1     ELEM TYPE#    # IN GROUP     GROUP#
     1      3      2      1      4      5      6
       1       1       1       1       1       1       1      NODE# DOFS
       2       1       1       1       1       1       1      NODE# DOFS
       3       1       1       1       1       1       1      NODE# DOFS
            1           9       GROUP#   #OF MAT PROPS
 1000000.00 384600.00 40.00 5.00    2.00  1.00  53.33  333.33  1.00
   0.000000    ELASTIC FOUNDATION CONSTANT
       6       0       # OF FORCES    UNUSED #
       1    0.000      0.000     0.000     0.000     0.000     0.000
       2    0.000      0.000     0.000     0.000     0.000     0.000
       3    0.000      0.000     0.000     0.000     0.000     0.000
       4    0.000      0.000     0.000     0.000     0.000     0.000
       5    0.000    100.000     0.000     0.000     0.000     0.000
       6    0.000      0.000     0.000     0.000     0.000     0.000
```

The first line is the control data. It stands for:

6 nodes in total
6 used nodes
1 elements in total
1 used element
3 dimensional
1 group
3 boundary conditions
1 material
1 < future solver control function >

The next 6 lines are a list of the nodes and their coordinates.

After that comes the group and element connectivity lists. We only have one and it is:

6 3D beam element type
1 number of elements in group
1 group number

With each group line, comes the list of elements and their connectivity in that group.

1 3 2 1 4 5 6 element #1, and nodes 3,2,1,4,5,6 just like we entered.

After the elements, the boundary conditions are listed by node number and the dofs in order. 1 for fixed, 0 for free.

```
1 1 1 1 1 1 1
2 1 1 1 1 1 1
3 1 1 1 1 1 1
```

This is the list of the first three nodes with all of their dofs fixed, just like we entered.

After that is the material property listings. They are the group # and the # of material properties and then the property listings. This is repeated for the number of groups used.

```
1   9
1000000.0   384600.0   40.0   5.0   2.0   1.0   53.33   333.333  1.0
```

This is for group 1 with 9 properties and there they are!

Last is the forces. The control data here is the number of nodes and an unused number.

```
6     0
1   0.0     0.0   0.0   0.0   0.0   0.0
2   0.0     0.0   0.0   0.0   0.0   0.0
3   0.0     0.0   0.0   0.0   0.0   0.0
4   0.0     0.0   0.0   0.0   0.0   0.0
5   0.0   100.0   0.0   0.0   0.0   0.0
6   0.0     0.0   0.0   0.0   0.0   0.0
```

Here we have 6 nodes, the unused 0, and the list of forces.

We will now take this data file to the solver directory and run it.

6.4 Solver

If you did not save the DEMO.DAT file in the solver directory from INPUT, copy it over now. Then start SOLVER from either the main menu or from DOS. At the first prompt type:

DEMO.DAT

The SOLVER will respond with:

```
ENTER INPUT FILENAME...DEMO.DAT
OUTPUT FILENAME IS DEMO.OUT
STRESS FILENAME IS DEMO.STS

 READING IN DATA
  MAX-DYNAMIC MEMORY ALLOCATED=29000       PROBLEM FITS IN MEMORY

CREATING LOCAL STIFFNESSES
CREATING ELEMENT #  -->         1
ASSEMBLING GLOBAL STIFFNESS
ASSEMBLING ELEMENT # -->         1

 LOADING
 INCORE SOLVER WORKING
 WRITING DISPLACEMENTS
 CALCULATING STRESSES
 Stress for element # -->         1
 CALCULATING REACTION FORCES
Reaction for element # -->         1

SOLUTION FINISHED
```

The solution is now finished and the file DEMO.STS can be copied over to the output directory. We can now look at the results in the OUTPUT program.

6.5 Output

After running the OUTPUT program, select FILE then OPEN and type

DEMO.STS↵

Notice how the .sts files are listed when OPEN was selected.

The beam should now be drawn on the screen. Select DISP to enter the displacement menu. Select SUPER IMPOSE and enter an exaggeration factor of 100.0. You should see the following:

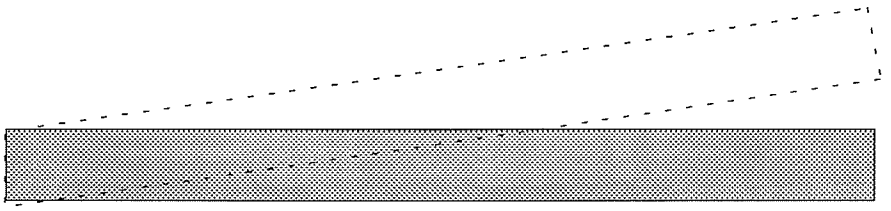


Figure Four: Displaced Shape

Now leave the displacement menu by selecting DONE and enter the stress menu by selecting STRESS. Now rotate the view so it is on an angle and then let's look at the major bending stress by selecting Sx. Theoretically the stress should be ±180.0 based on the equation $\sigma=Mc/I$. You should get the below drawing of the stress:

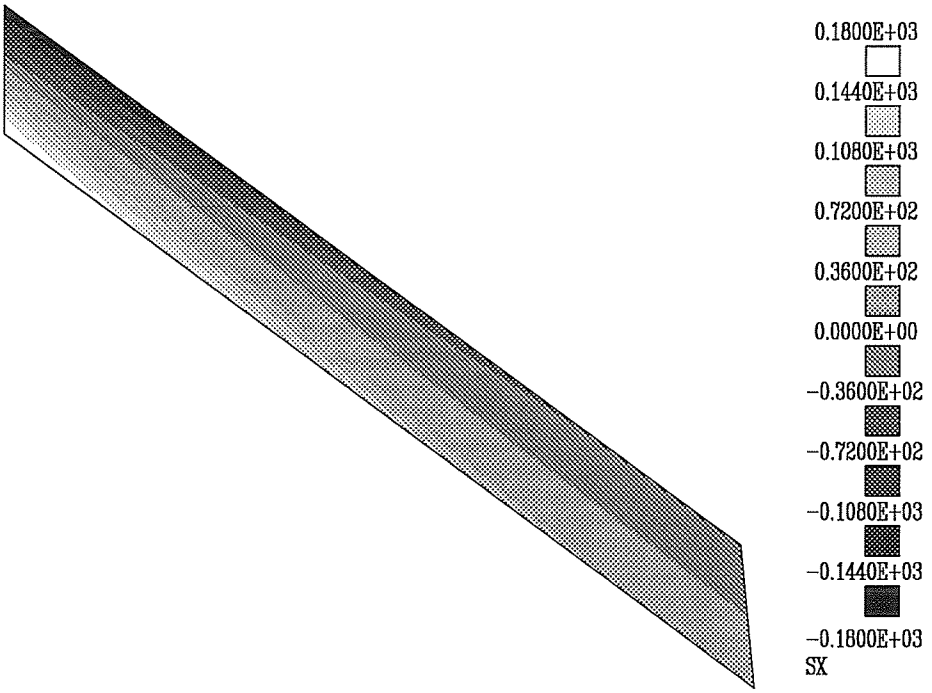


Figure Five: Beam Bending Stress

We can now refine our "mesh" to a two element model moving nodes 4,5,6 to x=60 and creating nodes 7,8,9 at x=120. After that add in a second element, remove the force on node 5 and add a force on node 8. This new model will produce the output shown below:

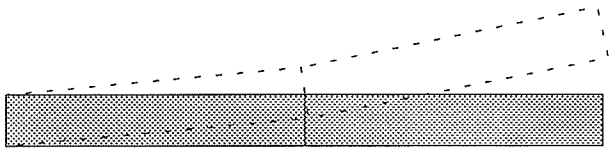


Figure Six: Two Element Mesh

Or even a 4 element mesh:

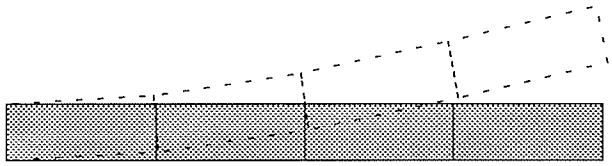


Figure Seven: Four Element Mesh

The tip displacement does not change with mesh refinement because the one element model has a shape function capable of capturing the linear bending moment. The only improvement in the two and four element meshes is that they show the curvature of the beam more accurately. The stress contours should also stay the same.

For other elements, a refined mesh will give markedly improved results and you must consider the physical problem carefully before developing your final model mesh.

7. APPENDIX

7.1 Element Description and Details

This section of the manual is designed to provide detailed information about the elements available in MHYFECS. There are elements for three types of problems and the problem types are:

- TWO DIMENSIONAL
- GRID
- THREE DIMENSIONAL

The rest of this section is divided into three subsections, each for a type of problem and the elements used for that type of problem. Section 7.1.2D is devoted to two dimensional elements, Section 7.1.G is on the grid elements, and 7.1.3D is on three dimensional problems. Remember, elements can only be mixed with other elements from the same type of problem.

7.1.2D Two Dimensional Problems

Two dimensional problems are problems that lie on a plane or can be modeled as a planer problem. A simple example is an "A" frame horizontally or vertically loaded but not loaded in or out of the plane on which it lies. The structures modeled in a two dimensional problem only have lateral displacements in the XY plane and possibly rotations about the Z axis. The following is a detailed description of the available two dimensional elements in MHYFECS.

2D Truss

The 2D truss element is used to model truss problems in which the structure of the problem lies in one plane or can be modeled as only one plane. A truss is defined as a member that has stiffness in axial compression and tension but is joined at its ends with pins so that there is no rotational stiffness. The loadings on a 2D truss are in plane forces with no moments. Typical problems modeled with 2D trusses are scaffolding and crane type structures.

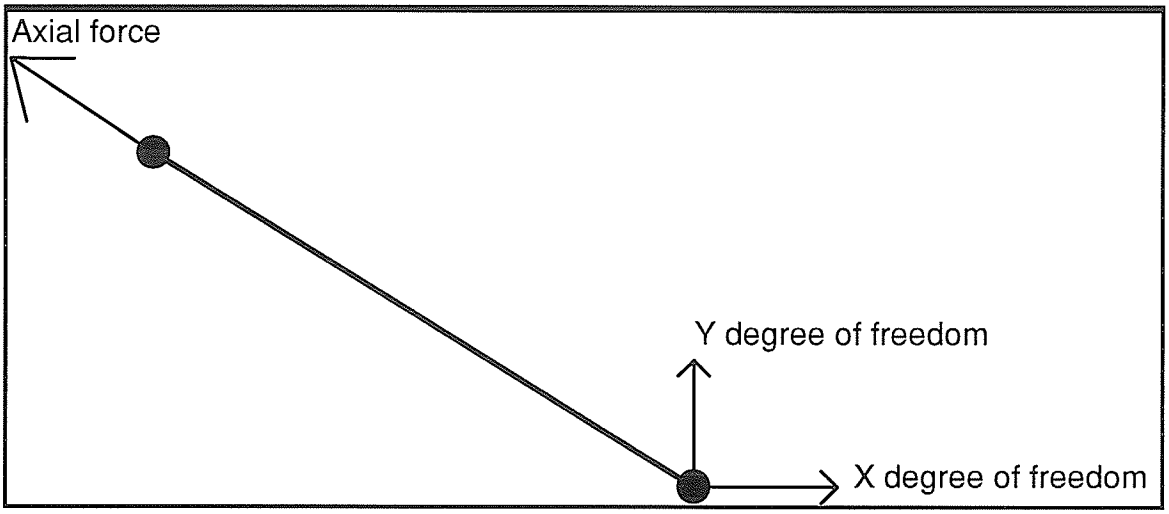
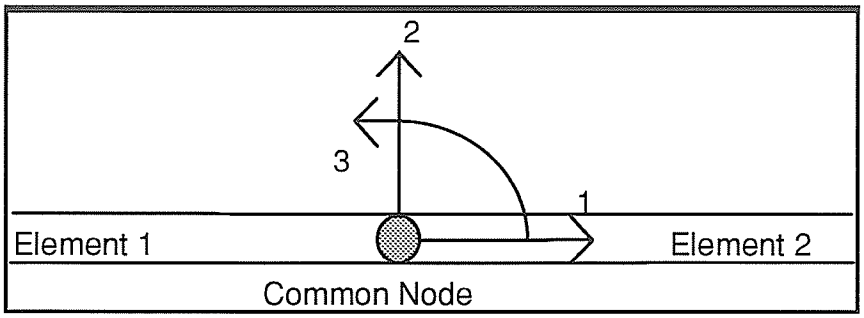


Figure Seven: 2D Truss Element

The element properties required to describe 2D truss element are the elasticity of the material, E, and the cross sectional area of the member, A, and the material density, ρ.

2D Beam



Like the 2D truss, the 2D beam is used in modeling structures that are all in one plane. The 2D beam has stiffness in three directions at each end. The first is the axial compression or tension, the second is the lateral deflection, and the third is the rotation at the end of the beam. When modeling a structure with 2D beam elements, the end of beams are joined with what is the equivalent of welding the two elements together. This means that the rotation of one beam member will rotate the connecting beam and deflect it in the other two degrees of freedom. See the below figure.

Figure Eight: 2D Beam Element Degrees of Freedom

The loading of the 2D beam can be in any of the three dofs. An axial load, a lateral load, or a moment applied to the end of the beam. The material properties required for the 2D beam element are the material elasticity, E, the cross sectional area of the member, A, the moment of inertia for bending, I, and the material density, ρ.

The 2D truss and 2D beam elements can be used in the same model if a mixture of beams and trusses is to be modeled. The joint between a beam and a truss will only join the first two (linear deflection) degrees of freedom and not the rotational dof of the beam since the truss has no rotational stiffness due to its pinned end.

Note: 2D Plane Strain and Plane Stress not documented and not yet implemented.

7.1.G Grid Problems

Grid structures are a special case of a two dimensional structure. They can not be mixed with normal two dimensional elements as they have different degrees of freedom. The grid is used to represent a roof like structure in which the loading is normal to the plane in which the structure lies. Loadings applied to grid structure generally represent distributed loads such as snow loads. The degrees of freedom of a grid are the deflection normal to the plane and the rotations along the axis in the plane. Fixing the normal deflection dof is like a rigid pillar under a node and fixing the rotational dofs represents clamped edges or symmetry conditions. To model a grid, two elements are provided, a grid beam and a grid plate.

Grid Beam

The grid beam is used to model a flat roof like structure that is made up of beams. The beams are joined together as if welded and are loaded with loads to represent snow or other weights. The grids have three degrees of freedom at each node. They are the axial twist, the length wise bending rotation, and the vertical deflection. These are better shown in the following figure.

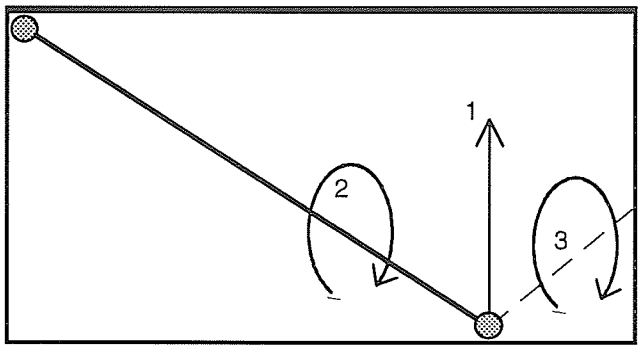


Figure Nine: Grid Beam Degrees of Freedom

The material properties required for this element are the elasticity of the material, E, the bulk modulus, G, the beam bending moment of inertia, Ixx, and the twisting moment of inertia, Izz, and the per unit length weight, W. Since no cross sectional area is asked for, the per unit length weight must be entered instead of the material density.

Grid Plate

The grid plate is used to model the stiffness of the plates lying on the plane of the grid. They have three degrees of freedom at each corner which are the out of plane deflection and two corresponding rotations matching the grid beams degrees of freedom. They can be used with or without the grid beams to model grid structures. An example of the grid plate's degrees of freedom is shown below.

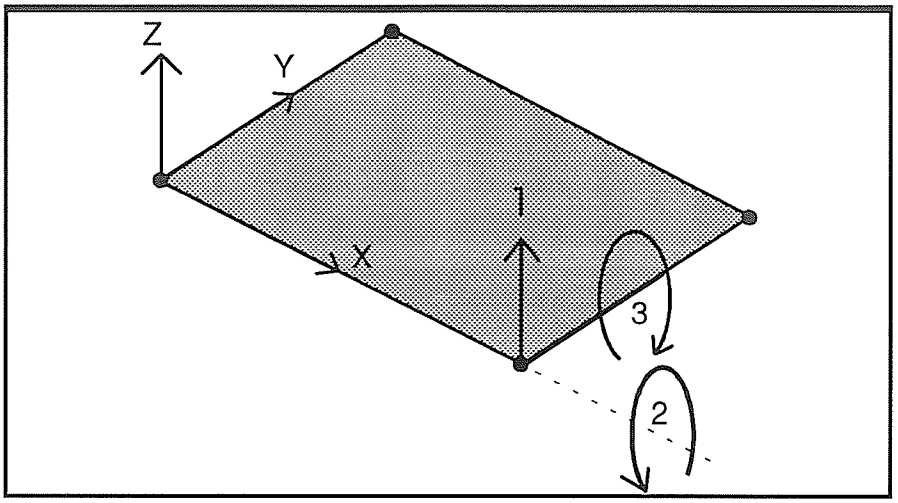


Figure Ten: Grid Plate Degrees of Freedom

The material properties required for a grid beam are the elasticity, E, the bulk modulus, G, the thickness of the plate, t, and the material density, ρ.

Remember, since the deflections and loadings for the grid are not the same as the two dimensional elements, the grid and two dimensional elements must not be mixed in a model. If you wish to model a grid with supports other than just fixing the nodes, you can develop a 3D model using 3D beams, 3D trusses, and 3D plates.

7.1.3D Three Dimensional Problems

Three dimension problems are the most common in real life so the selection of three dimensional elements is the largest in MHYFECS. Three dimensional elements all have at least three linear deflections at each node, and most of the time, three more rotational deflections. The elements available are:

- 3D spring
- 3D truss
- 3D frame
- 3D beam
- 3D plate
- 3D brick.

3D Spring

The first 3D element is the spring. It is exactly what its name implies, a spring element. The element connects two nodes and the stiffness between them is the spring constant. The spring is actually just like a truss but instead of elasticity and area, the only element property required is the spring constant, K. The spring's output is simply the deflection of the nodes connected by the spring.

3D Truss

The 3D truss is just as the 2D truss but is used in problems that can not be modeled in one plane. The ends are ball joint connections instead of pins so there is no rotational stiffness, just axial stiffness. The material properties are the elasticity of the material, E, the cross sectional area, A, and the material density, ρ.

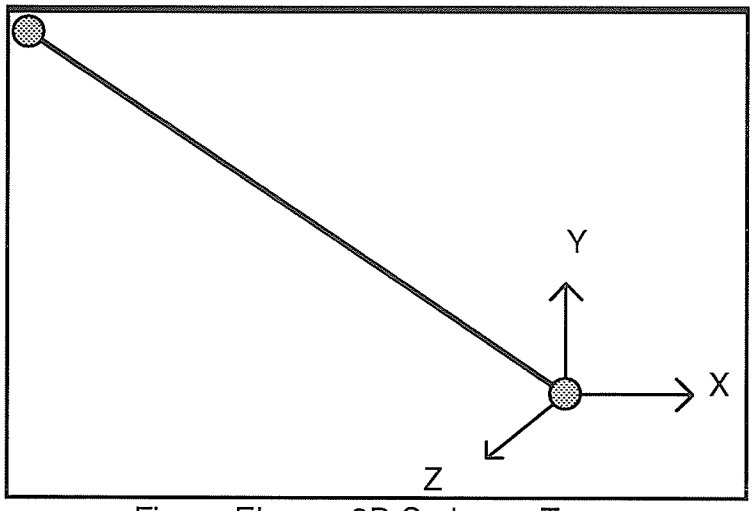


Figure Eleven: 3D Spring or Truss

3D Beam

The 3D beam is used to model beams in 3D structures. One of the major problems in 3D beam modeling is how to align the major and minor bending axis properly. This is accomplished by having the 3D beam element have 6 instead of the 2 nodes the 2d beam has. The element has 3 nodes at each end as shown in figure twelve. Note the alignment of the axis of bending with respect to the element numbering.

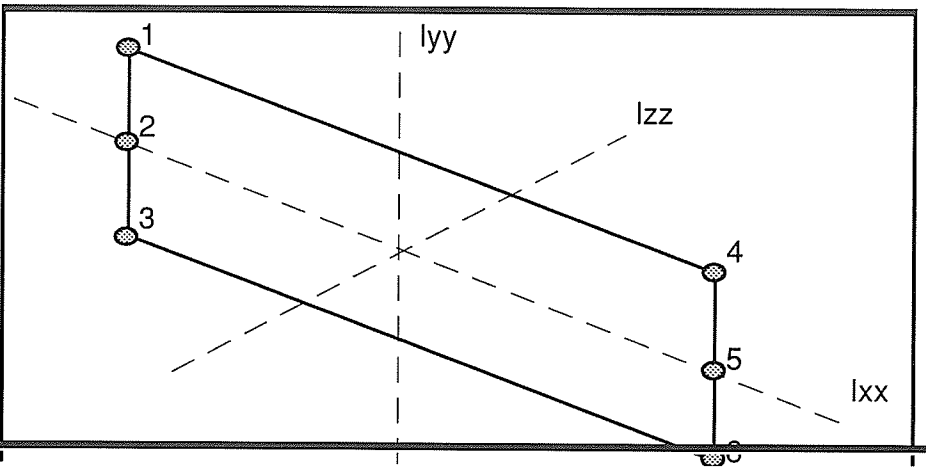


Figure Twelve: 3D Beam Node Numbering and Bending Axis Orientations

The major axis can now easily be fixed

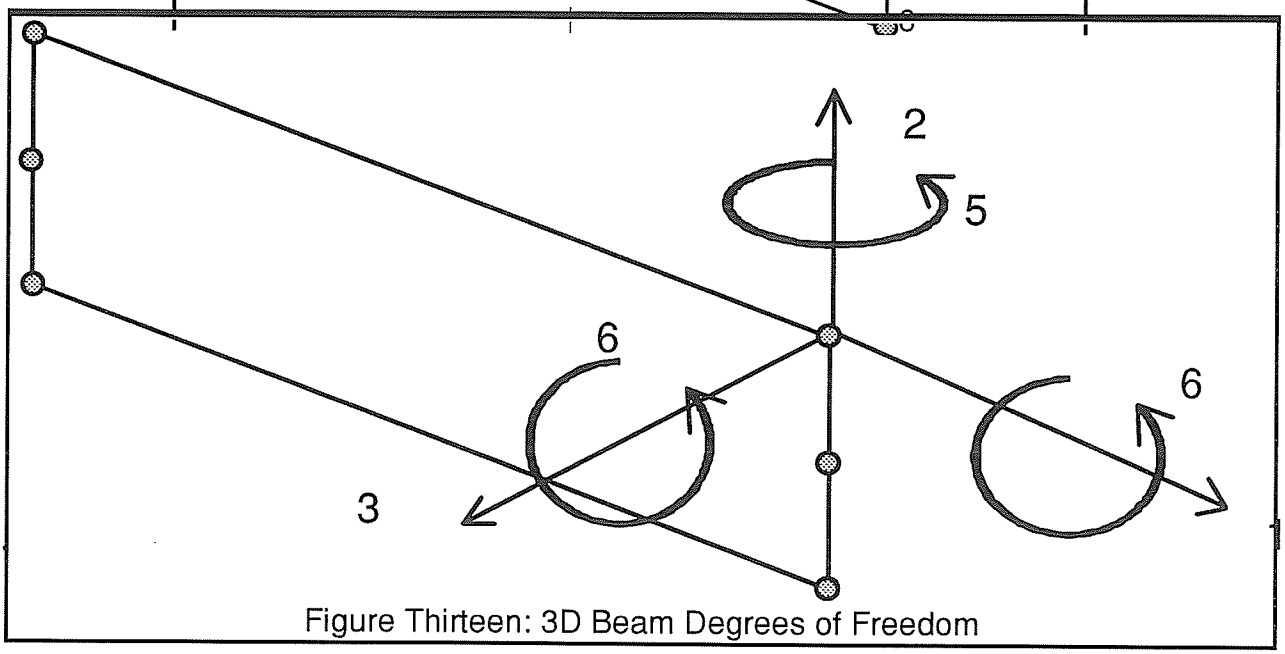


Figure Thirteen: 3D Beam Degrees of Freedom

into the model to insure accurate modeling. The beam has 6 dof at each node from, vertical, lateral, and axial deflections, rotations due to vertical and lateral deflection, and lastly rotation due to axial twist.

There are nine material properties for the 3D beam element. The first two are the elasticity of the material, E, and the bulk modulus of the material, G. The last seven properties describe the geometry of the beam. These are the cross sectional area, A, the maximum C values for the x and y axis, the three moments of inertia, Izz, Iyy, Ixx,

and lastly, the density, ρ. The following figure shows what Cxx and Cyy would be in some typical beam sections.

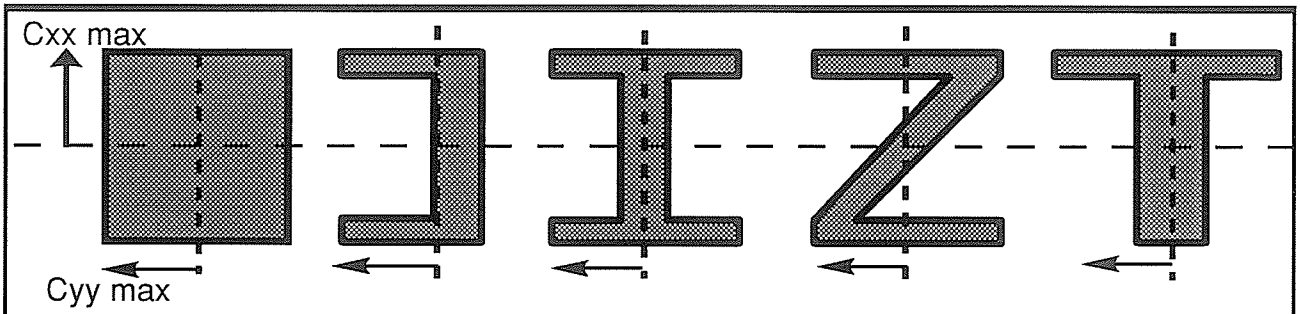


Figure Fourteen: Typical Cross Sections

List of Material Properties:

1) Elasticity, E
2) Bulk Modulus, G
3) Cross Sectional Area, A
4) Cxx max
5) Cyy max
6) Izz (twisting)
7) Iyy (minor bending)
8) Ixx (major bending)
9) Material Density

Restrictions:

Length to height ratio: The ratio of the length to the height should not be less than 5. If the ratio is physically less than this, that part of the structure should be modeled as a plate, or group of plates. Remember, beams are good for modeling lengthwise bending where as plates are better in shorter length shear transfer modeling.

Node positioning: The node positions of the 1,3 and 4,6 nodes must be the same y distance apart in both the beam table or property input as in the physical model of the system. For example, if a standard beam with a height of 24 inches is to be modeled, this beam must only occupy 24 inches in the model an have the 1,3 and 4,6 node distances equal to 24 inches.

3D Frame

The frame element is very similar to the beam element in that it represents a beam in reality. The main difference between the frame and beam is that unlike the beam element, the frame only has two nodes. The two nodes are on the neutral axis at each end of the beam. Due to the lack of the other defining nodes, the Iyy bending axis is always normal towards vertical and the Ixx normal towards horizontal. See figure fifteen. This limits the positioning of the beam, but for most orthogonal systems, the frame provides a fast and simple beam element. The material properties are the same as the beam element and the degrees of freedom are also the same.

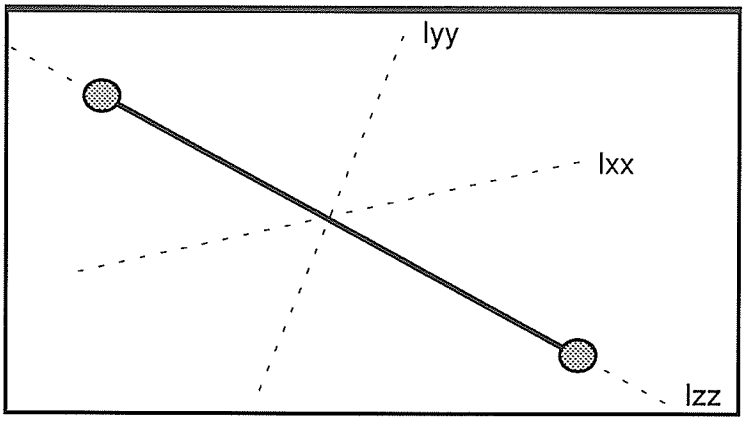


Figure Fifteen: 3D Frame Degrees of Freedom Orientation

3D Plate

The 3D plate element is used to model structures that are made of plates. The plate element included in MHYFECS is a 4 node rectangular plate with 6 degrees of freedom at each node. The dof are three linear displacements, the two rotation displacements in the bending planes, and a pseudo sixth stiffness. These are better shown in figure sixteen.

Restrictions:

Shape: The first, the rectangular shape, must be kept at all times since it is assumed rectangular in the element stiffness calculations. (This is in revision, a non-rectangular plate is under development)

Flatness: The second, the flatness of the plate, must also be obeyed at all times since this plate model can not deal with curvature. A curved plate is considered a shell which may be added to MHYFECS in the future.

Aspect Ratios: The third, the planar aspect ratio or width to length ratio, should always remain between 1/5 and 5. If the structure is out of this range, a 3D beam model is more appropriate. The last restriction on the plate element is the thickness ratio. The

thickness ratio, thickness over length, should never be less than 1*10e-5. When the plate is thinner that this limit, the stiffness calculations done by numerical integration begin to degrade and erroneous results will occur.

The material properties for the 3D plate element are the elasticity, E, the bulk modulus, G, the plate thickness, t, and the material density, ρ.

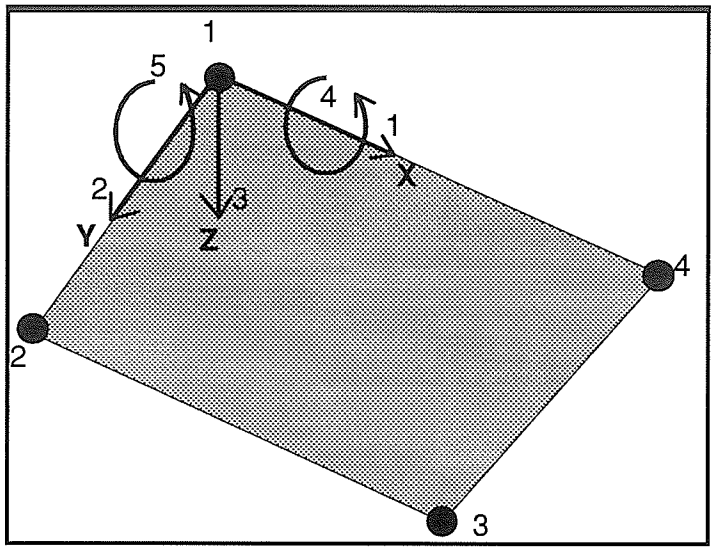


Figure Sixteen: 3D Plate Degrees of Freedom

The sixth degree of freedom, not shown above, is the inplane twisting of the plate which is in comparison to all other degrees of freedom, much stiffer. Due to this, it is modeled as a pseudo stiffness to prevent zeroes in the global stiffness matrix and to ease use of the plate with other three dimensional elements.

3D Solid

At this point in time, the 3D solid has been implemented on a testing level only to investigate it's usefulness and accuracy. The 3d solid is an 8 node brick as shown below. The only material properties to enter are the elasticity, E, the poisson's ratio of the material, ν, and the material density, ρ.

Mixing Elements

All of the 3D element types can be used in the same model since the same 6 dof's are used in each element. Stresses are available for all elements except the 3D spring.

7.2 Element Properties Quick Reference Table

ELEMENT	NUMBER OF NODES	MATERIAL PROPERTIES	DEGREES OF FREEDOM
2D Truss	2	Elasticity Section Area Density	X, Y
2D Beam	2	Elasticity Section Area Bending Moment Density	X, Y, RZ
Grid Beam	2	Elasticity Bulk Modulus Bending Moment Twisting Moment Weight / unit length	Z, RX, RY
Grid Plate	4	Elasticity Bulk Modulus Thickness Density	Z, RX, RY
3D Spring	2	Spring Constant	X, Y, Z
3D Truss	2	Elasticity Section Area Density	X, Y, Z
3D Frame	2	Elasticity Bulk Modulus Section Area Cxx max Cyy max Izz (twisting) Iyy (minor bending) Ixx (major bending) Material Density	X, Y, Z, RX, RY, RZ
3D Beam	6	Elasticity Bulk Modulus Section Area Cxx max Cyy max Izz (twisting) Iyy (minor bending) Ixx (major bending) Material Density	X, Y, Z, RX, RY, RZ
3D Plate	4	Elasticity Bulk Modulus Thickness Material Density	X, Y, Z, RX, RY, RZ
3D Brick	8	Elasticity Bulk Modulus Material Density	X, Y, Z

Appendix B1

GEOMMAKE PROGRAM LISTING

```
                PROGRAM  GEOMGEN_200N_100E
C
C THIS PROGRAM IS USED TO GENERATE 6 NODED
TRIANGULAR
C ELEMENTS FOR A TWO-DIMENSIONAL MESH ONLY!
C
C NE   NUMBER OF ELEMENTS
C NN   NUMBER OF NODES
C PN   NODAL LOCATIONS
C NEC  NODAL ELEMENTAL CONECTIVITY
C INF  NODAL FLAG VECTOR
C IEF  ELEMENTAL FLAG VECTOR
      REAL
PN(200,2),DUM,YMX,X1,Y1,X2,Y2,XMN,YMN,DY,XMX,DX,X(5),Y(5)
      INTEGER NEC(100,6),NE,NN,NUM,INF(200),IEF(100),TEST,FN,FE
C COLORS
      INTEGER WHITE,RED,LTBLUE,GREEN,BLACK,YELLOW
      INTEGER DKGREEN,DKBLUE,DKRED,ORANGE,PURPLE
C SEGMENTS
      integer
FILE,ELEMS,NODES,VIEW,QUIT,HELP,SEG,BACK,FILEIN,FILEOUT
      INTEGER
ZOOMI,ZOOMO,ZOOMA,MOVEU,MOVED,MOVEL,MOVER,MAKE
N,DELN
      INTEGER
MOVEN,MAKEET,MAKEES,DELE,OK,CANCEL,COMP
      CHARACTER*12 NAME,WORD,BUF
      NE=0
      NN=0
      TEST=0
      DO 1 I=1,200
            INF(I)=0
            PN(I,1)=0.0
1     PN(I,2)=0.0
            DO 2 I=1,100
            IEF(I)=0
            DO 3 J=1,6
3     NEC(I,J)=0
2     CONTINUE
C
C *** OPENING GKS ENVIROMENT ***
C
      XMN=0.0
      XMX=10.0
      YMN=0.0
      YMX=10.0
      DX=XMX-XMN
      DY=YMX-YMN
      CALL OPEN(XMN,YMN,XMX,YMX)

C *** COLOR INDEXES ***
C
C           RED
            CALL GSCR(1,1,1.0,0.1,0.1)
C         PURPLE
            CALL GSCR(1,12,0.70,0.0,0.70)
C          GREEN TEXT
            CALL GSCR(1,2,0.1,1.0,0.1)
C           BLACK
            CALL GSCR(1,3,0.0,0.0,0.0)
C         LIGHT BLUE
            CALL GSCR(1,4,0.3,0.3,0.6)
C        DARK BLUE
            CALL GSCR(1,10,0.,0.,0.5)
C         WHITE
            CALL GSCR(1,5,1.0,1.0,1.0)
C         GREY
C     CALL GSCR(1,6,0.5,0.5,0.5)
C         DARK GREEN
            CALL GSCR(1,7,0.,0.5,0.)
C         DARK RED
            CALL GSCR(1,8,0.5,0.,0.)
C         YELLOW
            CALL GSCR(1,9,1.0,1.0,0.)
C         ORANGE
            CALL GSCR(1,11,1.0,0.5,0.)
C
C SET COLORS
C
            RED=1
            GREEN=2
            BLACK=3
            LTBLUE=4
            WHITE=5
            DKGREEN=7
            DKRED=8
            YELLOW=9
            DKBLUE=10
            ORANGE=11
            PURPLE=12
C
C FILL COLOR,STYLE
            CALL GSFACI(LTBLUE)
            CALL GSFAIS(0)
C MARKER TYPE, COLOR
            CALL GSMK(1)
            CALL GSPMCI(ORANGE)
C TEXT REPRESENTATION
C     FONT,HEIGHT,COLOR,ALIGNMENT
            CALL GSTXFP(-101,1)
            CALL GSTXCI(WHITE)
            CALL GSTXAL(1,5)
            CALL GSCHH(2.0)
            CALL GSELNT(2)
C     BACKGROUND BORDER
            CALL BOX(-0.5,-0.5,33.5,100.5,WHITE)
C     MESSAGE BACKGROUND
            CALL BOX(0.,57.,33.,100.,LTBLUE)
C     INPUT BACKGROUND
            CALL BOX(0.,50.,33.,56.,BLACK)
C     ICON BACKGROUND
            CALL BOX(0.,0.,33.,49.,BLACK)

C
C SET SEGMENTS
C
      NODES=1
      ELEMS=2
      FILE=3
      VIEW=4
      HELP=5
      QUIT=6
      ZOOMI=7
      ZOOMO=8
      ZOOMA=9
      MOVEU=10
      MOVED=11
      MOVER=12
      MOVEL=13
      BACK=14
      FILEOUT=15
      FILEIN=16
      MAKEN=17
      DELN=18
      MOVEN=19
      OK=20
      CANCEL=21
      MAKEET=22
      MAKEES=23
      DELE=24
      COMP=25

            CALL GSLWSC(3.0)
C
C MAKE NODE AND ELEMENT MENU SELECTION ICONS
C
      CALL GCRSG(NODES)
      CALL GSVIS(NODES,0)
            CALL GSPLCI(LTBLUE)
            CALL GSFACI(DKBLUE)
            X(1)=4.
            X(2)=14.
            X(3)=14.
            X(4)=4.
            X(5)=X(1)
            Y(1)=35.
            Y(2)=35.
            Y(3)=45.
            Y(4)=45.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(5.0,36.0,'MENU')
            CALL GTX(5.0,40.0,'NODE')
      CALL GCLSG

      CALL GCRSG(ELEMS)
            CALL GSVIS(ELEMS,0)
            X(1)=19.
            X(2)=19.
            X(3)=29.
            X(4)=29.
            X(5)=X(1)
            Y(1)=35.
            Y(2)=45.
            Y(3)=45.
            Y(4)=35.
```

```
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(20.0,40.0,'ELEM')
            CALL GTX(20.0,36.0,'MENU')
      CALL GCLSG
C
C MAKE FILE AND VIEW MENU SELECTION ICONS
C
      CALL GCRSG(FILE)
            CALL GSVIS(FILE,0)
            X(1)=4.
            X(2)=14.
            X(3)=14.
            X(4)=4.
            X(5)=X(1)
            Y(1)=20.
            Y(2)=20.
            Y(3)=30.
            Y(4)=30.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(7.0,23.0,'FILE')
            CALL GCLSG

      CALL GCRSG(VIEW)
            CALL GSVIS(VIEW,0)
            X(1)=19.
            X(2)=29.
            X(3)=29.
            X(4)=19.
            X(5)=X(1)
            Y(1)=20.
            Y(2)=20.
            Y(3)=30.
            Y(4)=30.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(22.,23.0,'VIEW')
            CALL GCLSG
C
C MAKE QUIT/HELP ICONS
C
      CALL GCRSG(QUIT)
            CALL GSVIS(QUIT,0)
            CALL GSPLCI(YELLOW)
            CALL GSFACI(ORANGE)
            X(1)=19.
            X(2)=29.
            X(3)=29.
            X(4)=19.
            X(5)=X(1)
            Y(1)=5.
            Y(2)=5.
            Y(3)=15.
            Y(4)=15.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(22.,8.,'QUIT')
            CALL GCLSG

      CALL GCRSG(HELP)
            CALL GSVIS(HELP,0)
            CALL GSPLCI(LTBLUE)
            CALL GSFACI(DKBLUE)
            X(1)=4.
            X(2)=14.
            X(3)=14.
            X(4)=4.
            X(5)=X(1)
            Y(1)=5.
            Y(2)=5.
            Y(3)=15.
            Y(4)=15.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(7.0,8.0,'HELP')
            CALL GCLSG
C
C MAKE MOVE BUTTONS
C
      CALL GCRSG(MOVEU)
            CALL GSVIS(MOVEU,0)
            X(1)=13.
            X(2)=21.
            X(3)=17.
            X(4)=X(1)
            Y(1)=40.
            Y(2)=40.
            Y(3)=48.
            Y(4)=40.
            CALL GFA(4,X,Y)
            CALL GPL(4,X,Y)
            CALL GCLSG

      CALL GCRSG(MOVER)
            CALL GSVIS(MOVER,0)
            X(1)=21.
            X(2)=21.
            X(3)=29.
            X(4)=21.
            X(5)=X(1)
            Y(1)=40.
            Y(2)=32.
            Y(3)=36.
            Y(4)=Y(1)
            CALL GFA(4,X,Y)
            CALL GPL(4,X,Y)
            CALL GCLSG

      CALL GCRSG(MOVEL)
            CALL GSVIS(MOVEL,0)
            X(1)=13.
            X(2)=13.
            X(3)=5.
            X(4)=13.
            X(5)=X(1)
            Y(1)=40.
            Y(2)=32.
            Y(3)=36.
            Y(4)=Y(1)
            CALL GFA(4,X,Y)
            CALL GPL(4,X,Y)
            CALL GCLSG

      CALL GCRSG(MOVED)
            CALL GSVIS(MOVED,0)
            X(1)=13.
            X(2)=21.
            X(3)=17.
            X(4)=13.
            Y(1)=32.
            Y(2)=32.
            Y(3)=25.
            Y(4)=Y(1)
            CALL GFA(4,X,Y)
            CALL GPL(4,X,Y)
            CALL GTX(14.,33.5,'MOVE')
            CALL GCLSG
C
C MAKE ZOOMS AND RETURN TO MAIN ICONS
C
      CALL GCRSG(BACK)
            CALL GSVIS(BACK,0)
            X(1)=23.
            X(2)=32.
            X(3)=32.
            X(4)=23.
            X(5)=23.
            Y(1)=1.
            Y(2)=1.
            Y(3)=10.
            Y(4)=10.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(25.,3.0,'MENU')
            CALL GTX(25.,6.0,'MAIN')
            CALL GCLSG

      CALL GCRSG(ZOOMI)
            CALL GSVIS(ZOOMI,0)
            X(1)=4.
            X(2)=14.
            X(3)=14.
            X(4)=4.
            X(5)=X(1)
            Y(1)=3.
            Y(2)=3.
            Y(3)=11.
            Y(4)=11.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(4.1,5.0,'ZOOM IN')
            CALL GCLSG
```

```
      CALL GCRSG(ZOOMA)
            CALL GSVIS(ZOOMA,0)
            X(1)=19.
            X(2)=29.
            X(3)=29.
            X(4)=19.
            X(5)=19.
            Y(1)=15.
            Y(2)=15.
            Y(3)=23.
            Y(4)=23.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(22.,16.,' ALL')
            CALL GTX(22.,19.,'VIEW')
            CALL GCLSG

      CALL GCRSG(ZOOMO)
            CALL GSVIS(ZOOMO,0)
            X(1)=4.
            X(2)=14.
            X(3)=14.
            X(4)=4.
            X(5)=X(1)
            Y(1)=15.
            Y(2)=15.
            Y(3)=23.
            Y(4)=23.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(5.5,19.0,'ZOOM')
            CALL GTX(5.,16.0,' OUT')
            CALL GCLSG
C
C MAKE FILE MENU ICONS
C
      CALL GCRSG(FILEIN)
            CALL GSVIS(FILEIN,0)
            X(1)=10.
            X(2)=10.
            X(3)=23.
            X(4)=23.
            X(5)=X(1)
            Y(1)=37.
            Y(2)=45.
            Y(3)=45.
            Y(4)=37.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(10.5,38.0,'DATA FILE')
            CALL GTX(10.5,41.5,' READ IN')
      CALL GCLSG

      CALL GCRSG(FILEOUT)
            CALL GSVIS(FILEOUT,0)
            X(1)=10.
            X(2)=23.
            X(3)=23.
            X(4)=10.
            X(5)=X(1)
            Y(1)=25.
            Y(2)=25.
            Y(3)=34.
            Y(4)=34.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(10.5,26.0,'DATA FILE')
            CALL GTX(10.5,29.5,'WRITE OUT')
            CALL GCLSG

      CALL GCRSG(COMP)
            CALL GSVIS(COMP,0)
            X(1)=10.
            X(2)=23.
            X(3)=23.
            X(4)=10.
            X(5)=10.
            Y(1)=13.
            Y(2)=13.
            Y(3)=22.
            Y(4)=22.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(10.5,14.,'  DATA ')
            CALL GTX(10.5,17.,' CHECK ')
            CALL GCLSG
C
C MAKE NODE MENU ICONS
C
      CALL GCRSG(MAKEN)
            CALL GSVIS(MAKEN,0)
            CALL GSPLCI(GREEN)
            CALL GSFACI(DKGREEN)
            X(1)=12
            X(2)=21.
            X(3)=21.
            X(4)=12
            X(5)=12.
            Y(1)=13.
            Y(2)=13.
            Y(3)=22.
            Y(4)=22.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(13.,14.0,'NODE')
            CALL GTX(13.,17.0,'MAKE')
            CALL GCLSG

      CALL GCRSG(DELN)
            CALL GSVIS(DELN,0)
            CALL GSPLCI(RED)
            CALL GSFACI(DKRED)
            X(1)=12
            X(2)=21.
            X(3)=21.
            X(4)=12
            X(5)=12.
            Y(1)=37.
            Y(2)=37.
            Y(3)=45.
            Y(4)=45.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(13.,38.0,' NODE')
            CALL GTX(13.,41.0,'DELETE')
            CALL GCLSG

      CALL GCRSG(MOVEN)
            CALL GSVIS(MOVEN,0)
            CALL GSPLCI(YELLOW)
            CALL GSFACI(ORANGE)
            X(1)=12
            X(2)=21.
            X(3)=21.
            X(4)=12
            X(5)=12.
            Y(1)=25.
            Y(2)=25.
            Y(3)=34.
            Y(4)=34.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(13.,26.0,'NODE')
            CALL GTX(13.,29.0,'MOVE')
            CALL GCLSG
C
C MAKE OK AND CANCLE ICONS
C
      CALL GCRSG(OK)
            CALL GSVIS(OK,0)
            CALL GSPLCI(ORANGE)
            CALL GSFACI(YELLOW)
            X(1)=1.
            X(2)=10.
            X(3)=10.
            X(4)=1.
            X(5)=1.
            Y(1)=1.
            Y(2)=1.
            Y(3)=10.
            Y(4)=10.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GSTXCI(BLACK)
            CALL GTX(3.,4.,' OK?')
            CALL GSTXCI(WHITE)
            CALL GCLSG

      CALL GCRSG(CANCEL)
```

```
            CALL GSVIS(CANCEL,0)
            CALL GSPLCI(RED)
            CALL GSFACI(DKRED)
            X(1)=12.
            X(2)=21.
            X(3)=21.
            X(4)=12.
            X(5)=12.
            Y(1)=1.
            Y(2)=1.
            Y(3)=10.
            Y(4)=10.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(12.5,4.,'CANCEL')
            CALL GCLSG
C
C MAKE ELEMENT MENU ICONS
C
    CALL GCRSG(MAKEET)
            CALL GSVIS(MAKEET,0)
            CALL GSPLCI(GREEN)
            CALL GSFACI(DKGREEN)
            X(1)=11.
            X(2)=22.
            X(3)=22.
            X(4)=11
            X(5)=11.
            Y(1)=13.
            Y(2)=13.
            Y(3)=22.
            Y(4)=22.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(11.5,14.0,'3 NODES')
            CALL GTX(11.5,17.0,'MAKE BY')
            CALL GCLSG

    CALL GCRSG(MAKEES)
            CALL GSVIS(MAKEES,0)
            X(1)=11.
            X(2)=22.
            X(3)=22.
            X(4)=11.
            X(5)=11.
            Y(1)=37.
            Y(2)=37.
            Y(3)=45.
            Y(4)=45.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(11.5,38.0,'? NODES')
            CALL GTX(11.5,41.0,'MAKE BY')
            CALL GCLSG

    CALL GCRSG(DELE)
            CALL GSVIS(DELE,0)
            CALL GSPLCI(RED)
            CALL GSFACI(DKRED)
            X(1)=11.
            X(2)=22.
            X(3)=22.
            X(4)=11.
            X(5)=11.
            Y(1)=25.
            Y(2)=25.
            Y(3)=34.
            Y(4)=34.
            Y(5)=Y(1)
            CALL GFA(5,X,Y)
            CALL GPL(5,X,Y)
            CALL GTX(11.5,26.0,'ELEMENT')
            CALL GTX(11.5,29.0,'DELETE')
            CALL GCLSG


    CALL GASGWK(1,NODES)
    CALL GASGWK(1,ELEMS)
    CALL GASGWK(1,FILE)
    CALL GASGWK(1,QUIT)
    CALL GASGWK(1,HELP)
    CALL GASGWK(1,VIEW)
    CALL GASGWK(1,BACK)
    CALL GASGWK(1,ZOOMI)
    CALL GASGWK(1,ZOOMO)
    CALL GASGWK(1,ZOOMA)
    CALL GASGWK(1,MOVEU)
    CALL GASGWK(1,MOVED)
    CALL GASGWK(1,MOVER)
    CALL GASGWK(1,MOVEL)
    CALL GASGWK(1,FILEIN)
    CALL GASGWK(1,FILEOUT)
    CALL GASGWK(1,MOVEN)
    CALL GASGWK(1,MAKEN)
    CALL GASGWK(1,DELN)
    CALL GASGWK(1,OK)
    CALL GASGWK(1,CANCEL)
    CALL GASGWK(1,DELE)
    CALL GASGWK(1,MAKEET)
    CALL GASGWK(1,MAKEES)
    CALL GASGWK(1,COMP)

    CALL GSDTEC(NODES,1)
    CALL GSDTEC(ELEMS,1)
    CALL GSDTEC(FILE,1)
    CALL GSDTEC(VIEW,1)
    CALL GSDTEC(HELP,1)
    CALL GSDTEC(QUIT,1)
    CALL GSDTEC(ZOOMI,1)
    CALL GSDTEC(ZOOMO,1)
    CALL GSDTEC(ZOOMA,1)
    CALL GSDTEC(MOVEU,1)
    CALL GSDTEC(MOVED,1)
    CALL GSDTEC(MOVER,1)
    CALL GSDTEC(MOVEL,1)
    CALL GSDTEC(BACK,1)
    CALL GSDTEC(FILEIN,1)
    CALL GSDTEC(FILEOUT,1)
    CALL GSDTEC(MAKEN,1)
    CALL GSDTEC(DELN,1)
    CALL GSDTEC(MOVEN,1)
    CALL GSDTEC(OK,1)
    CALL GSDTEC(CANCEL,1)
    CALL GSDTEC(MAKEET,1)
    CALL GSDTEC(MAKEES,1)
    CALL GSDTEC(DELE,1)
    CALL GSDTEC(COMP,1)
    GOTO 99
C
C *** MOVE NODE ROUTINE ***
C
10  FN=1
            FE=1
            CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
            DY=YMX-YMN
            DX=XMX-XMN
11  CALL GSVIS(OK,0)
            CALL BOX(0.0,59.0,33.0,100.0,LTBLUE)
            CALL GTX(0.0,90.,'PICK NODE')
            CALL GTX(0.0,80.,'MENU AREA=QUIT')
            CALL GRQLC(1,2,II,JJ,X1,Y1)
            IF(X1.GT.0.74) THEN
             CALL GSVIS(OK,0)
             CALL GSVIS(CANCEL,0)
             GOTO 300
             END IF
            X1=X1/0.7314678
            Y1=Y1/0.7487394
            X1=X1*DX + XMN
            Y1=Y1*DY + YMN
            CALL FIND(X1,Y1,PN,NUM,NN)
            CALL BOX(0.0,50.0,33.0,56.0,BLACK)
            WRITE(BUF,*) NUM
            CALL GTX(0.0,50.0,BUF)
            CALL GSVIS(OK,1)
            CALL GSVIS(CANCEL,1)
12  CALL GRQPK(1,2,II,SEG,JJ)
            IF(SEG.EQ.CANCEL) GOTO 11
            IF(SEG.EQ.OK) GOTO 13
            GOTO 12
13  CALL GSVIS(OK,0)
            CALL BOX(0.0,59.0,33.0,100.0,LTBLUE)
            CALL GTX(0.0,90.,'PICK POSITION')
            CALL GRQLC(1,2,II,JJ,X1,Y1)
            X1=X1/0.7314678
            Y1=Y1/0.7487394
            X1=X1*DX + XMN
            Y1=Y1*DY + YMN
            PN(NUM,1)=X1
            PN(NUM,2)=Y1
            GOTO 10

C
C *** NODAL INPUT ROUTINE ***
C
20  FN=1
```

```
                FE=0
                CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
25    CALL BOX(0.0,59.0,33.,100.,LTBLUE)
                DY=YMX-YMN
                CALL GTX(0.0,90.,'NODAL CREATION')
                CALL GTX(0.0,84.,'INPUT NODAL #')
                CALL GTX(0.0,78.,' ENTER=QUIT ')
21    FORMAT(I6)
22    FORMAT(F6.2)
                CALL GRQST(1,1,II,10,NAME)
                READ(NAME,21,ERR=25) NUM
                IF(NUM.GT.0) THEN
                 INF(NUM)=1
                 IF(NUM.GT.NN) NN=NUM
                 IF(NUM.LT.NN) THEN
                  CALL GSTXCI(RED)
                  CALL GTX(0.0,70.,'WARNING! REPEATED NODE')
                  CALL GSTXCI(WHITE)
                 END IF
23     CALL GTX(0.0,60.,'INPUT X COORD')
                CALL GRQST(1,1,II,10,NAME)
                READ(NAME,22,ERR=23) PN(NUM,1)
                CALL BOX(0.0,59.0,33.0,69.0,LTBLUE)
24     CALL GTX(0.0,60.,'INPUT Y COORD')
                CALL GRQST(1,1,II,10,NAME)
                READ(NAME,22,ERR=24) PN(NUM,2)
                CALL BOX(0.0,59.0,33.0,89.0,LTBLUE)
                CALL SEARCH(NN,PN,XMN,XMX,YMN,YMX)
                GOTO 20
                ELSE
                 GOTO 300
                END IF
C
C *** ELEMENT INPUT ***
C
29    TEST=1
30    FN=1
                FE=1
                CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
31    CALL BOX(0.0,59.0,33.,100.,LTBLUE)
                CALL GTX(0.0,90.,'ELEMENT CREATION')
                CALL GTX(0.0,84.,'INPUT ELEMENT #')
                CALL GTX(0.0,78.,' ENTER=QUIT ')
                CALL GRQST(1,1,II,10,NAME)
                READ(NAME,21,ERR=31) NUM
                IF(NUM.GT.0) THEN
                  IF(NUM.GT.NE) THEN
                          NE=NUM
                    END IF
                    IF(NUM.LT.NE) THEN
                          CALL GSTXCI(RED)
                          CALL GTX(0.0,70.,'WARNING!
REPEATED ELEMENT')
                          CALL GSTXCI(WHITE)
                    END IF
                  DX=XMX-XMN
                  DY=YMX-YMN

                  CALL GTX(0.0,60.,'PICK FIRST NODE')
                   CALL GRQLC(1,2,II,JJ,X1,Y1)
                   IF(X1.GT.0.74) GOTO 31
                   X1=X1/0.7314678
                   Y1=Y1/0.7487394
                   X1=X1*DX + XMN
                   Y1=Y1*DY + YMN
                   CALL FIND(X1,Y1,PN,SEG,NN)
                   CALL BOX(0.0,50.0,33.0,56.0,BLACK)
                   WRITE(BUF,*) SEG
                   CALL GTX(0.0,50.0,BUF)
                   NEC(NUM,1)=SEG
                   CALL BOX(0.0,59.0,33.0,69.0,LTBLUE)

                  CALL GTX(0.0,60.,'PICK SECOND NODE')
                   CALL GRQLC(1,2,II,JJ,X1,Y1)
                   IF(X1.GT.0.74) GOTO 31
                   X1=X1/0.7314678
                   Y1=Y1/0.7487394
                   X1=X1*DX + XMN
                   Y1=Y1*DY + YMN
                   CALL FIND(X1,Y1,PN,SEG,NN)
                   CALL BOX(0.0,50.0,33.0,56.0,BLACK)
                   WRITE(BUF,*) SEG
                   CALL GTX(0.0,50.0,BUF)
                   NEC(NUM,2)=SEG
                   CALL BOX(0.0,59.0,33.0,69.0,LTBLUE)

                  CALL GTX(0.0,60.,'PICK THIRD NODE')
                   CALL GRQLC(1,2,II,JJ,X1,Y1)
                   IF(X1.GT.0.74) GOTO 31
                   X1=X1/0.7314678
                   Y1=Y1/0.7487394
                   X1=X1*DX + XMN
                   Y1=Y1*DY + YMN
                   CALL FIND(X1,Y1,PN,SEG,NN)
                   CALL BOX(0.0,50.0,33.0,56.0,BLACK)
                   WRITE(BUF,*) SEG
                   CALL GTX(0.0,50.0,BUF)
                   NEC(NUM,3)=SEG
                   CALL BOX(0.0,59.0,33.0,69.0,LTBLUE)

                IF(TEST.EQ.1) THEN
C AUTO NODES
                  NEC(NUM,4)=NN+1
                  NEC(NUM,5)=NN+2
                  NEC(NUM,6)=NN+3
                  INF(NN+1)=1
                  INF(NN+2)=1
                  INF(NN+3)=1
                  NN=NN+3
                  GOTO 33
                 END IF
                CALL BOX(0.0,59.0,33.,83.,LTBLUE)
                CALL GTX(0.0,75.,'PICK NODE IF EXISTS')
                CALL GTX(0.0,70.,' OR MENU AREA FOR ')
                CALL GTX(0.0,66.,'AUTO NODE GENERATION')

                 CALL GTX(0.0,60.,'PICK FORTH NODE')
                  CALL GRQLC(1,2,II,JJ,X1,Y1)
                  IF(X1.GT.0.74) THEN
                   NEC(NUM,4)=NN+1
                   INF(NN+1)=1
                   NN=NN+1
                   GOTO 32
                   END IF
                  X1=X1/0.7314678
                  Y1=Y1/0.7487394
                  X1=X1*DX + XMN
                  Y1=Y1*DY + YMN
                  CALL FIND(X1,Y1,PN,SEG,NN)
                  CALL BOX(0.0,50.0,33.0,56.0,BLACK)
                  WRITE(BUF,*) SEG
                  CALL GTX(0.0,50.0,BUF)
                  NEC(NUM,4)=SEG
32      CALL BOX(0.0,59.0,33.0,66.0,LTBLUE)

                 CALL GTX(0.0,60.,'PICK FIFTH NODE')
                  CALL GRQLC(1,2,II,JJ,X1,Y1)
                  IF(X1.GT.0.74) THEN
                   NEC(NUM,5)=NN+1
                   INF(NN+1)=1
                   NN=NN+1
                   GOTO 34
                   END IF
                  X1=X1/0.7314678
                  Y1=Y1/0.7487394
                  X1=X1*DX + XMN
                  Y1=Y1*DY + YMN
                  CALL FIND(X1,Y1,PN,SEG,NN)
                  CALL BOX(0.0,50.0,33.0,56.0,BLACK)
                  WRITE(BUF,*) SEG
                  CALL GTX(0.0,50.0,BUF)
                  NEC(NUM,5)=SEG
34      CALL BOX(0.0,59.0,33.0,66.0,LTBLUE)

                 CALL GTX(0.0,60.,'PICK SIXTH NODE')
                  CALL GRQLC(1,2,II,JJ,X1,Y1)
                  IF(X1.GT.0.74) THEN
                   NEC(NUM,6)=NN+1
                   INF(NN+1)=1
                   NN=NN+1
                   GOTO 33
                   END IF
                  X1=X1/0.7314678
                  Y1=Y1/0.7487394
                  X1=X1*DX + XMN
                  Y1=Y1*DY + YMN
                  CALL FIND(X1,Y1,PN,SEG,NN)
                  CALL BOX(0.0,50.0,33.0,56.0,BLACK)
                  WRITE(BUF,*) SEG
                  CALL GTX(0.0,50.0,BUF)
                  NEC(NUM,6)=SEG

33      IEF(NUM)=1

C
C *** GENERATE MIDSIDE NODES ***
C
                I=NUM
```

```
                PN(NEC(I,4),1)=( PN(NEC(I,1),1) + PN(NEC(I,2),1)
)/2.0D+00
                PN(NEC(I,4),2)=( PN(NEC(I,1),2) + PN(NEC(I,2),2)
)/2.0D+00
                PN(NEC(I,5),1)=( PN(NEC(I,3),1) + PN(NEC(I,2),1)
)/2.0D+00
                PN(NEC(I,5),2)=( PN(NEC(I,3),2) + PN(NEC(I,2),2)
)/2.0D+00
                PN(NEC(I,6),1)=( PN(NEC(I,3),1) + PN(NEC(I,1),1)
)/2.0D+00
                PN(NEC(I,6),2)=( PN(NEC(I,3),2) + PN(NEC(I,1),2)
)/2.0D+00
C
                GOTO 30
                ELSE
                TEST=0
                GOTO 350
                END IF
C
C *** NODAL DELETION ***
C
40    FN=1
              FE=0
              CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
              DX=XMX-XMN
              DY=YMX-YMN
              CALL BOX(0.0,59.0,33.,100.,LTBLUE)
              CALL GTX(0.0,90.,'NODAL DELETION ')
              CALL GTX(0.0,84.,' PICK NODE ')
              CALL GTX(0.0,78.,'MENU AREA=QUIT')
               CALL GRQLC(1,2,II,JJ,X1,Y1)
               IF(X1.GT.0.74) GOTO 300
               X1=X1/0.7314678
               Y1=Y1/0.7487394
               X1=X1*DX + XMN
               Y1=Y1*DY + YMN
               CALL FIND(X1,Y1,PN,NUM,NN)
               CALL BOX(0.0,50.0,33.0,56.0,BLACK)
               WRITE(BUF,*) NUM
               CALL GTX(0.0,50.0,BUF)
               DUM=0
               DO 42 I=1,NE
               DO 42 J=1,6
                IF(NUM.EQ.NEC(I,J).AND.IEF(I).EQ.1) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(0.0,74.,'NODE BELONGS TO ELEMENT')
                 CALL GTX(0.0,66.,' DELETE ELEMENT FIRST! ')
                 CALL GSTXCI(WHITE)
                 DUM=1
                 CALL GSVIS(OK,1)
41      CALL GRQPK(1,2,II,SEG,JJ)
                 IF(SEG.EQ.OK) GOTO 45
                 GOTO 41
                 END IF
42    CONTINUE
                 IF(NUM.GT.NN) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(0.0,70.,'NODE DOES NOT EXIST')
                 CALL GSTXCI(WHITE)
                 DUM=1
                 CALL GSVIS(OK,1)
44      CALL GRQPK(1,2,II,SEG,JJ)
                 IF(SEG.EQ.OK) GOTO 45
                 GOTO 44
                 END IF
45      CALL GSVIS(OK,0)
                 IF(DUM.EQ.0) THEN
                          CALL GSVIS(OK,1)
                          CALL GSVIS(CANCEL,1)
47          CALL GRQPK(1,2,II,SEG,JJ)
                          IF(SEG.EQ.CANCEL) THEN
                                   CALL GSVIS(OK,0)
                                   CALL GSVIS(CANCEL,0)
                          ELSE IF (SEG.EQ.OK) THEN
                           INF(NUM)=0
                           CALL GTX(0.0,70.,'NODE DELETED!')
                           CALL GSVIS(OK,0)
                           CALL GSVIS(CANCEL,0)
                          ELSE
                           GOTO 47
                          END IF
                 END IF
                GOTO 40
C
C *** ELEMENTAL DELETION ***
C
50    FN=0
              FE=1
              CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
              CALL BOX(0.0,59.0,33.,100.,LTBLUE)
              CALL GTX(0.0,90.,'ELEMENTAL DELETION ')
              CALL GTX(0.0,84.,' INPUT ELEMENT #')
              CALL GTX(0.0,78.,'  ENTER=QUIT  )
              CALL BOX(0.0,59.0,33.0,78.0,LTBLUE)
              CALL GRQST(1,1,II,10,NAME)
              READ(NAME,21,ERR=299) NUM
              IF(NUM.LT.1) GOTO 350
              IF(NUM.GT.NE) THEN
               CALL GSTXCI(RED)
               CALL GTX(0.0,70.,'ELEMENT DOES NOT EXIST')
               CALL GSTXCI(WHITE)
               CALL GSVIS(OK,1)
53      CALL GRQPK(1,2,II,SEG,JJ)
               IF (SEG.EQ.OK) GOTO 54
               GOTO 53
              END IF
              CALL GTX(0.0,70.,'ELEMENT DELETED! ')
              IEF(NUM)=0
54    CALL GSVIS(OK,0)
              GOTO 50
C
C *** IMPORT DATA FILE ***
C
60    CALL BOX(0.0,59.0,33.0,100.0,LTBLUE)
61    FORMAT(A10)
              FE=1
              FN=1
              CALL GTX(0.0,90.,'  FILE INPUT ')
              CALL GTX(0.0,80.,'ENTER FILENAME')
              CALL GRQST(1,1,II,10,WORD)
              READ(WORD,61,ERR=299,END=250) NAME
              OPEN(UNIT=2,FILE=NAME)
              READ(2,*,ERR=299,END=250) NE,NN,NBC
              NBC=NBC*1
              DO 64 I=1,NE

READ(2,*,ERR=299,END=250)J,NEC(J,1),NEC(J,4),NEC(J,2),NEC(J,5),
  &NEC(J,3),NEC(J,6)
64    IEF(J)=1
              DO 66 I=1,NN
               READ(2,*,ERR=299,END=250)J,PN(J,1),PN(J,2)
66    INF(J)=1
              DO 68 I=1,NE
               PN(NEC(I,4),1)=( PN(NEC(I,1),1) + PN(NEC(I,2),1)
)/2.0D+00
               PN(NEC(I,4),2)=( PN(NEC(I,1),2) + PN(NEC(I,2),2)
)/2.0D+00
               PN(NEC(I,5),1)=( PN(NEC(I,3),1) + PN(NEC(I,2),1)
)/2.0D+00
               PN(NEC(I,5),2)=( PN(NEC(I,3),2) + PN(NEC(I,2),2)
)/2.0D+00
               PN(NEC(I,6),1)=( PN(NEC(I,3),1) + PN(NEC(I,1),1)
)/2.0D+00
               PN(NEC(I,6),2)=( PN(NEC(I,3),2) + PN(NEC(I,1),2)
)/2.0D+00
               INF(NEC(I,4))=1
               INF(NEC(I,5))=1
68    INF(NEC(I,6))=1
              CALL SEARCH(NN,PN,XMN,XMX,YMN,YMX)
              CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
              GOTO 250
C
C *** ZOOM ROUTINE ***
C
70    DY=YMX-YMN
              DX=XMX-XMN
              CALL BOX(0.0,59.0,33.0,100.0,LTBLUE)
              CALL GTX(0.0,90.,'PICK CENTER')
              CALL GRQLC(1,2,II,JJ,X1,Y1)
              X1=X1/0.7314678
              Y1=Y1/0.7487394
              X1=X1*DX + XMN
              Y1=Y1*DY + YMN
              CALL GTX(0.0,80.,'PICK RADIUS')
              CALL GRQLC(1,2,II,JJ,X2,Y2)
              X2=X2/0.7314678
              Y2=Y2/0.7487394
              X2=X2*DX + XMN
              Y2=Y2*DY + YMN
              DUM=( (Y1-Y2)**2.0 + (X1-X2)**2.0 )**0.50
              YMN=Y1-DUM
              YMX=Y1+DUM
              XMN=X1-DUM
              XMX=X1+DUM
              DY=YMX-YMN
              DX=XMX-XMN
```

```
                CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
                GOTO 200

C
C *** END MENU ***
C
99    CALL BOX(0.0,59.0,33.0,100.0,LTBLUE)
                CALL GSVIS(NODES,1)
                CALL GSVIS(ELEMS,1)
                CALL GSVIS(FILE,1)
                CALL GSVIS(VIEW,1)
                CALL GSVIS(QUIT,1)
                CALL GSVIS(HELP,1)
C       CALL GRSGWK(1)
                CALL GSLWSC(3.0)
                CALL GSPLCI(PURPLE)
                CALL GTX(0.0,90.,' MAIN MENU ')
                CALL GRQPK(1,2,II,SEG,JJ)
                IF(SEG.EQ.NODES) GOTO 300
                IF(SEG.EQ.ELEMS) GOTO 350
                IF(SEG.EQ.FILE) GOTO 250
                IF(SEG.EQ.VIEW) GOTO 200

                IF(SEG.EQ.QUIT) GOTO 999
                IF(SEG.EQ.HELP) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(0.0,50.0,'READ MANUAL!')
                 CALL GSTXCI(WHITE)
                 END IF
                GOTO 99
C
C *** WRITE DATA FILE ***
C
134  CALL BOX(0.0,59.0,33.,100.,LTBLUE)
     CALL GTX(0.0,90.,' FILE OUTPUT ')
     CALL GTX(0.0,80.,'ENTER FILENAME')

     CALL GRQST(1,1,II,10,WORD)
     READ(WORD,61,ERR=134,END=134) NAME
     OPEN(UNIT=4,FILE=NAME)
     WRITE(4,*) NE,NN,NN
     DO 135 J=1,NE
135  WRITE(4,145)J,NEC(J,1),NEC(J,4),NEC(J,2),
    &           NEC(J,5),NEC(J,3),NEC(J,6)
                DO 140 I=1,NN
140     WRITE(4,150) I,PN(I,1),PN(I,2)
145     FORMAT(7I7)
150     FORMAT(I7,F10.3,F10.3)
                CALL GTX(0.0,70.0,'GEOMETRY FILE WRITTEN')
                GOTO 250

299     CALL GTX(0.0,70.0,' ERROR HAS OCCURED ')
                CALL GTX(0.0,65.0,'  IN DATA READING  ')

                PAUSE
                GOTO 250
C
C VIEW MENU
C
200     CALL BOX(0.0,59.0,33.,100.,LTBLUE)
                CALL GSVIS(NODES,0)
                CALL GSVIS(ELEMS,0)
                CALL GSVIS(FILE,0)
                CALL GSVIS(VIEW,0)
                CALL GSVIS(QUIT,0)
                CALL GSVIS(HELP,0)
                CALL GSVIS(ZOOMI,1)
                CALL GSVIS(ZOOMO,1)
                CALL GSVIS(ZOOMA,1)
                CALL GSVIS(MOVEU,1)
                CALL GSVIS(MOVED,1)
                CALL GSVIS(MOVER,1)
                CALL GSVIS(MOVEL,1)
                CALL GSVIS(BACK,1)
205     CALL GTX(0.0,90.,' VIEW MENU ')
                CALL GRQPK(1,2,II,SEG,JJ)
                IF(SEG.EQ.ZOOMI) GOTO 70
                IF(SEG.EQ.ZOOMA) THEN
                                CALL
SEARCH(NN,PN,XMN,XMX,YMN,YMX)
                                FN=1
                                FE=1
                                CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
                                GOTO 205
                END IF
                IF(SEG.EQ.BACK) THEN
                 CALL GSVIS(ZOOMI,0)
                 CALL GSVIS(ZOOMO,0)
                 CALL GSVIS(ZOOMA,0)
                 CALL GSVIS(MOVEU,0)
                 CALL GSVIS(MOVED,0)
                 CALL GSVIS(MOVER,0)
                 CALL GSVIS(MOVEL,0)
                 CALL GSVIS(BACK,0)
                 GOTO 99
                END IF
                DY=YMX-YMN
                IF(SEG.EQ.ZOOMO) THEN
                  YMX=YMX+0.5*DY
                  YMN=YMN-0.5*DY
                  XMX=XMX+0.5*DY
                  XMN=XMN-0.5*DY
                  CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
                  GOTO 205
                END IF
                IF(SEG.EQ.MOVEU) THEN
                  YMX=YMX-0.2*DY
                  YMN=YMN-0.2*DY
                  CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
                  GOTO 205
                END IF
                IF(SEG.EQ.MOVED) THEN
                  YMX=YMX+0.2*DY
                  YMN=YMN+0.2*DY
                  CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
                  GOTO 205
                END IF
                IF(SEG.EQ.MOVER) THEN
                  XMX=XMX-0.2*DY
                  XMN=XMN-0.2*DY
                  CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
                  GOTO 205
                END IF
                IF(SEG.EQ.MOVEL) THEN
                  XMX=XMX+0.2*DY
                  XMN=XMN+0.2*DY
                  CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
                  GOTO 205
                END IF

                GOTO 205
C
C FILE MENU
C
250     CALL BOX(0.0,59.0,33.,100.,LTBLUE)
                CALL GSVIS(NODES,0)
                CALL GSVIS(ELEMS,0)
                CALL GSVIS(FILE,0)
                CALL GSVIS(VIEW,0)
                CALL GSVIS(QUIT,0)
                CALL GSVIS(HELP,0)
                CALL GSVIS(FILEIN,1)
                CALL GSVIS(FILEOUT,1)
                CALL GSVIS(COMP,1)
                CALL GSVIS(BACK,1)
255     CALL GTX(0.0,90.,' FILE MENU ')
                CALL GRQPK(1,2,II,SEG,JJ)
                IF(SEG.EQ.FILEIN) GOTO 60
                IF(SEG.EQ.FILEOUT) GOTO 134
                IF(SEG.EQ.COMP) THEN
                                CALL CHECK(NN,PN,NE,NEC,INF,IEF)
                                FN=1
                                FE=1
                                CALL
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,FN,FE)
                                END IF
                IF(SEG.EQ.BACK) THEN
                 CALL GSVIS(FILEIN,0)
                 CALL GSVIS(FILEOUT,0)
                 CALL GSVIS(BACK,0)
                 CALL GSVIS(COMP,0)
                 GOTO 99
                END IF
                GOTO 255
C
C NODAL MENU
C
300     CALL BOX(0.0,59.0,33.,100.,LTBLUE)
                CALL GSVIS(NODES,0)
                CALL GSVIS(ELEMS,0)
                CALL GSVIS(FILE,0)
                CALL GSVIS(VIEW,0)
                CALL GSVIS(QUIT,0)
```

```
              CALL GSVIS(HELP,0)
              CALL GSVIS(MAKEN,1)
              CALL GSVIS(MOVEN,1)
              CALL GSVIS(DELN,1)
              CALL GSVIS(BACK,1)
305   CALL GTX(0.0,90.,' NODE MENU ')
              CALL GRQPK(1,2,II,SEG,JJ)
              IF(SEG.EQ.MOVEN) GOTO 10
              IF(SEG.EQ.MAKEN) GOTO 20
              IF(SEG.EQ.DELN) GOTO 40
              IF(SEG.EQ.BACK) THEN
               CALL GSVIS(MAKEN,0)
               CALL GSVIS(MOVEN,0)
               CALL GSVIS(DELN,0)
               CALL GSVIS(BACK,0)
               GOTO 99
              END IF
              GOTO 305
C
C ELEMENTAL MENU
C
350   CALL BOX(0.0,59.0,33.,100.,LTBLUE)
              TEST=0
              CALL GSVIS(NODES,0)
              CALL GSVIS(ELEMS,0)
              CALL GSVIS(FILE,0)
              CALL GSVIS(VIEW,0)
              CALL GSVIS(QUIT,0)
              CALL GSVIS(HELP,0)
              CALL GSVIS(MAKEET,1)
              CALL GSVIS(MAKEES,1)
              CALL GSVIS(DELE,1)
              CALL GSVIS(BACK,1)
355   CALL GTX(0.0,90.,'ELEMENT MENU')
              CALL GRQPK(1,2,II,SEG,JJ)
              IF(SEG.EQ.MAKEET) GOTO 29
              IF(SEG.EQ.MAKEES) GOTO 30
              IF(SEG.EQ.DELE) GOTO 50
              IF(SEG.EQ.BACK) THEN
               CALL GSVIS(MAKEET,0)
               CALL GSVIS(MAKEES,0)
               CALL GSVIS(DELE,0)
               CALL GSVIS(BACK,0)
               GOTO 99
              END IF
              GOTO 355

C999   CALL CLOSE
999   END
C
C ******************************************
C
              SUBROUTINE
DRAW(PN,NEC,NN,NE,XMN,XMX,YMN,YMX,INF,IEF,JN,JE)
              REAL PN(200,2),XMX,XMN,X(4),Y(4),DY,YMX,YMN
              INTEGER NEC(100,6),NN,NE,INF(200),IEF(100),JN,JE
              CHARACTER*12 BUF
C LOOPS TO DRAW GRAPHIC CHECKS OF GEOMETRY
C
C *** DRAWING ELEMENTS  ***
C
              CALL GSELNT(1)
              CALL GSLWSC(1.0)
              CALL GSWN(1,XMN,XMX,YMN,YMX)
              CALL BOX(XMN,YMN,XMX,YMX,3)
              CALL GSFAIS(1)
              DY=YMX-YMN
              CALL GSPLCI(5)
              CALL GSTXCI(5)
              CALL GSFACI(10)
              CALL GSCHH(DY/50.0)
              IF(JE.EQ.1) THEN
              DO 4 I=1,NE
              IF(IEF(I).EQ.0) GOTO 4
              X(1)=PN(NEC(I,1),1)
              X(2)=PN(NEC(I,2),1)
              X(3)=PN(NEC(I,3),1)
              Y(1)=PN(NEC(I,1),2)
              Y(2)=PN(NEC(I,2),2)
              Y(3)=PN(NEC(I,3),2)
              X(4)=X(1)
              Y(4)=Y(1)
              CALL GFA(4,X,Y)
              CALL GPL(4,X,Y)
              X(4)=(X(1)+X(2)+X(3))/3.0
              Y(4)=(Y(1)+Y(2)+Y(3))/3.0
              WRITE(BUF,*) I
              CALL GTX(X(4)-DY/7.0,Y(4),BUF)
4     CONTINUE
              END IF
c
C DRAWING NODES
C
              IF(JN.EQ.1) THEN
              CALL GSTXCI(11)
              DO 2 I=1,NN
               IF(INF(I).EQ.0) GOTO 2
               X(1)=PN(I,1)
               Y(1)=PN(I,2)
               WRITE(BUF,*)I
               CALL GTX(X(1)-DY/7.0,Y(1),BUF)
               CALL GPM(1,X,Y)
2     CONTINUE
              END IF
C
              CALL BOX(XMN,YMN,XMX,YMX,12)
              CALL GSCHH(2.0)
              CALL GSELNT(2)
              CALL GSTXCI(5)
              CALL GSLWSC(3.0)
              CALL GSPLCI(12)

              RETURN
              END

C
C************************************************
C
              SUBROUTINE
SEARCH(NN,PN,XMN,XMX,YMN,YMX)
              REAL PN(200,2),XMN,XMX,YMN,YMX,DY,DX
              INTEGER NN
              YMN=100.0
              XMN=100.0
              XMX=0.0
              YMX=0.0
              DO 50 I=1,NN
              IF(PN(I,1).GT.XMX) THEN
               XMX=PN(I,1)
              END IF
              IF(PN(I,2).GT.YMX) THEN
               YMX=PN(I,2)
              END IF
              IF(PN(I,1).LT.XMN) THEN
               XMN=PN(I,1)
              END IF
              IF(PN(I,2).LT.YMN) THEN
               YMN=PN(I,2)
50    END IF
              DY=YMX-YMN
              DX=XMX-XMN
              IF (DY.GT.DX) THEN
               XMX=XMN+DY
              ELSE
               YMX=YMN+DX
              END IF
              DY=YMX-YMN
              DX=XMX-XMN

              YMN=YMN-0.05*DY
              YMX=YMX+0.05*DY
              XMN=XMN-0.05*DX
              XMX=XMX+0.05*DX

              DY=YMX-YMN
              DX=XMX-XMN

              RETURN
              END
C************************************************************
*******

    SUBROUTINE CLOSE

*    CLOSE GKS
    CALL GDAWK (0)
    CALL GDAWK (1)
    CALL GCLWK (0)
    CALL GCLWK (1)
    CALL GCLKS ()
    CLOSE (14)
    RETURN
    END


C************************************************************
*******
    SUBROUTINE OPEN(XMN,YMN,XMX,YMX)
C    OPEN GKS AND SET VIEWPORT AND WINDOW
    INTEGER*2 ERR,DCUNIT,XRAS,YRAS
```

```
      REAL
XDCMX,YDCMX,SCALE,XNDC,YNDC,XMN,XMX,YMN,YMX
      CHARACTER*12 PROMPT
      prompt='
      OPEN (14,FILE='ERRORS')
      CALL GOPKS (14,1024)
      CALL GOPWK (0,0,0)
      CALL GOPWK (1,0,1)
      CALL GACWK (0)
      CALL GACWK (1)
      CALL GQDSP (1,ERR,DCUNIT,XDCMX,YDCMX,XRAS,YRAS)
      IF (XDCMX.GT.YDCMX) THEN
                  SCALE=XDCMX
      ELSE
                  SCALE=YDCMX
      END IF
      XNDC=XDCMX/SCALE
      YNDC=YDCMX/SCALE
C
C TRANSFORMATION #1
C
      CALL GSWN (1,XMN,XMX,YMN,YMX)
      CALL GSVP (1,0.0,0.73*XNDC,0.0,YNDC)
C
C TRANSFORMATION #2
C
      CALL GSWN (2,-1.0,34.0,-1.0,101.0)
      CALL GSVP (2,0.74*XNDC,XNDC,0.0,YNDC)

      CALL GSWKWN (1,0.0,XNDC,0.0,YNDC)
      CALL GSWKVP (1,0.0,XDCMX,0.0,YDCMX)
      CALL GSELNT (1)
      CALL GSTXFP (-101,2)
      CALL GSLCM(1,2,0,1)
      CALL
GINST(1,1,12,PROMPT,1,0.75*XDCMX,XDCMX,0.5*YDCMX,0.6*YD
CMX,12
     &,1,10,PROMPT)
      CALL
GINPK(1,2,1,1,1,1,0.75*XDCMX,XDCMX,0.0,YDCMX,10,PROMPT)
      RETURN
      END
C
C*************************************************************
*********
C
                  subroutine box (xmin,YMIN,xmax,ymax,IC)
                  real xmin, ymin, xmax, ymax,x(5), y(5)
                  INTEGER IC
c
                  x(1) = xmin
                  y(1) = ymin
                  x(2) = xmax
                  y(2) = ymin
                  x(3) = xmax
                  y(3) = ymax
                  x(4) = xmin
                  y(4) = ymax
                  x(5) = xmin
                  y(5) = ymin
                  IF (IC.EQ.12) THEN
                   CALL GSFAIS(0)
                  ELSE
                   CALL GSFAIS(1)
                  END IF
                  CALL GSFACI(IC)
                  call gFA (5, x, y)
                  return
                  end
C
C*************************************************************
***
C
                  SUBROUTINE FIND(X,Y,PN,NUM,NN)
                  REAL X,Y,PN(200,2),DIST,TEST
                  INTEGER NUM,NN
C
C THIS SUBROUTINE FINDS THE CLOSEST NODE TO THE PICK
POINT
C
                  NUM=1
                  TEST=((X-PN(1,1))**2.0+(Y-PN(1,2))**2.0)**0.50
                  DO 10 I=2,NN
                  DIST=((X-PN(I,1))**2.0+(Y-PN(I,2))**2.0)**0.50
                  IF (DIST.LT.TEST) THEN
                   TEST=DIST
                   NUM=I
                  END IF
10    CONTINUE
                  RETURN
                  END
C
C**************************************
C
                  SUBROUTINE CHECK(NN,PN,NE,NEC,INF,IEF)
C
C THIS SUBROUTINE CHECKS THE DATA FOR DOUBLE NODES
AND
C COMPRESSES THE DATA TO ITS LOWEST NUMERICAL ORDER.
C
C *** AS OF OCTOBER 24 1991, THERE'S A BUG IN THIS ROUTINE
C *** THE BUG DELETES SOME MIDSIDE NODES FOR UNKNOWN
REASONS!!
C
                  REAL PN(200,2),TEST,DUM
                  INTEGER
NN,NE,RNE,RNN,NEC(100,6),INF(200),IEF(100)
                  TEST=0.01
C
C HERE, THE NODES ARE CHECKED FOR REDUNDANCY AND IF
REDUNDANT,
C THEY ARE ELIMINATED AND THE NEC VECTOR RENUMBERED
ACCORDINGLY.
C
                  DO 10 I=1,NN-1
                   IF(INF(I).EQ.0) GOTO 10
                  DO 15 J=I+1,NN
                   IF(INF(J).EQ.0) GOTO 15
                  DUM=( (PN(I,1)-PN(J,1))**2.0 + (PN(I,2)-PN(J,2))**2.0
)**0.50
                  IF (DUM.LT.TEST) THEN
                    INF(J)=0
                    DO 25 II=1,NE
                    IF(IEF(II).EQ.0) GOTO 25
                    DO 20 JJ=1,6
20      IF( NEC(II,JJ).EQ.J) NEC(II,JJ)=I
25    CONTINUE
                  END IF
15    CONTINUE
10    CONTINUE
C
C NEXT, THE NODES ARE COMPRESSED TO THE LOWEST ORDER
POSSIBLE
C
C FINDING TOTAL NUMBER USED NODES
                  DO 27 I=1,NN
                  INF(I)=0
                  DO 27 J=1,NE
                  DO 27 JJ=1,6
                  IF(NEC(J,JJ).EQ.I.AND.IEF(J).EQ.1) INF(I)=1
27    CONTINUE
                  RNN=0
                  DO 30 I=1,NN
30    IF(INF(I).EQ.1) RNN=RNN+1
                  NN=RNN
C
C SHIFTING DOWN NODES
                  K=0
                  DO 35 I=1,RNN
31    K=K+1
                  IF(INF(K).EQ.0) GOTO 31
                  PN(I,1)=PN(K,1)
                  PN(I,2)=PN(K,2)
C
C RENUMBERING ELEMENT CONNECTIVITY MATRIX
C
                  DO 33 J=1,NE
                  DO 33 JJ=1,6
33    IF(NEC(J,JJ).EQ.K) NEC(J,JJ)=I
                  INF(I)=1
35    CONTINUE
                  DO 40 I=NN+1,200
40    INF(I)=0
C
C ELEMENTS ARE NOW COMPRESSED TO LOWEST NUMERICAL
ORDER
C
                  RNE=0
                  DO 50 I=1,NE
50    IF(IEF(I).EQ.1) RNE=RNE+1
                  NE=RNE
C
C SHIFTING DOWN
                  K=0
                  DO 55 I=1,RNE
51    K=K+1
                  IF(IEF(K).EQ.0) GOTO 51
                  NEC(I,1)=NEC(K,1)
                  NEC(I,2)=NEC(K,2)
                  NEC(I,3)=NEC(K,3)
```

```
                NEC(I,4)=NEC(K,4)
                NEC(I,5)=NEC(K,5)
                NEC(I,6)=NEC(K,6)
55      IEF(I)=1
                DO 60 I=RNE+1,100
60      IEF(I)=0
C
C RESETTING MIDSIDE NODES
C
                DO 70 I=1,NE
                 PN(NEC(I,4),1)=( PN(NEC(I,1),1) + PN(NEC(I,2),1)
)/2.0D+00
                 PN(NEC(I,4),2)=( PN(NEC(I,1),2) + PN(NEC(I,2),2)
)/2.0D+00
                 PN(NEC(I,5),1)=( PN(NEC(I,3),1) + PN(NEC(I,2),1)
)/2.0D+00
                 PN(NEC(I,5),2)=( PN(NEC(I,3),2) + PN(NEC(I,2),2)
)/2.0D+00
                 PN(NEC(I,6),1)=( PN(NEC(I,3),1) + PN(NEC(I,1),1)
)/2.0D+00
70      PN(NEC(I,6),2)=( PN(NEC(I,3),2) + PN(NEC(I,1),2) )/2.0D+00

                RETURN
                END
```

Appendix B2

POTENTIAL FLOW SOLVER PROGRAM

```
                PROGRAM UNDER_GATE_FLOW
C
C FINITE ELEMENT PROGRAM TO SOLVE POTENTIAL
C       FLOW THEORY 2-D PROBLEMS
C
C NOMENCLATURE
C
C NE    NUMBER OF ELEMENTS
C NN    NUMBER OF NODES
C NDN   NUMBER OF DEFINED NODES
C PN    NODAL LOCATIONS
C KL    LOCAL STIFFNESS MATRIX
C QL    LOCAL FORCE VECTOR
C NOC   NUMBER OF BC TYPE 0 = ENTRANCE EDGE
C                   1 = EXIT EDGE
C NOE   NUMBER OF ELEMENT
C NOS   NUMBER OF SIDE ON ELEMENT
C KE    GLOBAL STIFFNESS MATRIX
C QE    GLOBAL FORCE VECTOR AND RETURNED POTENTIAL
C NEC   NODAL ELEMENTAL CONECTIVITY

C U     HORIZONTAL VELOCITY AT NODE
C V     VERTICAL VELOCITY AT NODE
C P     PRESSURE AT NODE

C A     AREA OF ELEMENT
C C,B DISTANCES USED IN B MATRIX

C SL    SIDE LENGTH OF AN ELEMENT
C NHIT  TOP NODE OF HEAD IN
C NHIB  BOTTOM NODE OF HEAD IN
C NHOT  TOP NODE OF HEAD OUT
C NHOB  BOTTOM NODE OF HEAD OUT
C QI    FLOW IN
C QO    FLOW OUT
C ZI    FREE SURFACE HEIGHT OF FLOW IN
C ZO    FREE SURFACE HEIGHT OF FLOW OUT
C HI    HEAD IN
C HO    HEAD OUT

C ZLR  LAGRAINGIAN MATRIX
C ZDL  SHAPE FUNCTION DERIVATIVE MATRIX
C ZDXY REAL WORLD DERIVATIVES
C
C GKS VARIABLES
C X Y POSITION ARRAYS
C XMX YMX MAXIMUMS
C
                REAL*8
KE(190,190),QE(190),PN(190,3),KL(6,6),QL(6),SUM
                REAL*8 C(3),B(3),A,U(190),V(190),P(190),VMAX,SF(15)
                REAL*8
DUM,GRAV,HI,HO,ZI,ZO,QO,QI,DENS,Q(190),TEST,MAX
                REAL*8
ZLR(2,3),ZDL(3,6),ZDXY(2,6),TEMP,CONCRI,DIFF
                REAL*8 SH,SL,PA,PB,ST,PMIN,PMAX,PSCA
                REAL X(7),Y(7),XMX,YMX,S,T,RED,GRN,BLU
                INTEGER
NEC(100,6),NOC(100),NOS(100),CCN,NIS(10),NVA(6)
                INTEGER
NOE(100),NFN(7,4),ITLIM,LENGTH,DEPTH,WIDTH
                CHARACTER*32 BUF
                ITER=0
                PMAX=0.0
                PMIN=100.0
C
C *** READ DATA ***
C
    OPEN(UNIT=2,FILE='GEOM.DAT')
    READ(2,*) NE,NN,NBC
    DO 10 I=1,NE
10   READ(2,*)
J,NEC(J,1),NEC(J,4),NEC(J,2),NEC(J,5),NEC(J,3),NEC(J,6)
    DO 20 I=1,NN
20   READ(2,*) J,PN(J,1),PN(J,2)
    DO 30 I=1,NBC
30   READ(2,*) J,NOC(J),NOE(J),NOS(J)
                READ(2,*)NHIT,NHIB
                READ(2,*)NHOT,NHOB
                READ(2,*)WIDTH
                READ(2,*) (NIS(I),I=1,WIDTH)
                READ(2,*)LENGTH,DEPTH
                DO 33 I=1,LENGTH
33    READ(2,*) (NFN(I,J),J=1,DEPTH)
                READ(2,*)CONCRI,ITLIM
                READ(2,*)GRAV,DENS
                READ(2,*) (NVA(I),I=1,LENGTH-1)
                CLOSE(UNIT=2)
C
C *** OPEN OUTPUT FILES ***
C
                open(unit=4,file='VALUES.OUT')
                OPEN(UNIT=5,FILE='NEWGEOM.DAT')
                OPEN(UNIT=6,FILE='ANIMATE.DAT')

C
C *** OPENING GKS ENVIROMENT ***
C
C *** SEARCH FOR LARGEST DIMENSIONS ***
                XMX=0.0D+00
                YMX=0.0D+00
                DO 50 I=1,NN
                IF(PN(I,1).GT.XMX) THEN
                 XMX=PN(I,1)
                END IF
                IF(PN(I,2).GT.YMX) THEN
                 YMX=PN(I,2)
50    END IF
                DUM=1.36111*YMX
                IF(XMX.LT.DUM) THEN
                 XMX=DUM
                END IF
                DUM=0.734694*XMX
                IF(YMX.LT.DUM) THEN
                 YMX=DUM
                END IF
                CALL OPEN(XMX,YMX)
C
C COLOUR INDEXES
C             GREEN TEXT
                CALL GSCR(1,13,0.2,0.9,0.2)
C       LIGHT BLUE ELEMENTS AND NODES
                CALL GSCR(1,11,0.9,0.0,0.9)
C           RED VELOCITY VECTORS
                CALL GSCR(1,14,1.0,0.6,0.6)
C              BLACK
                CALL GSCR(1,15,0.0,0.0,0.0)
C
C PRESSURE GRADIENT COLOURS
C
                OPEN (UNIT=9,FILE='COLOUR.MAP')
                read(9,*) CCN
                IF(CCN.GT.12) CCN=12
                DO 51 I=1,CCN
                READ(9,*) RED,GRN,BLU
51    CALL GSCR(1,I,RED,GRN,BLU)
                CLOSE(UNIT=9)
C
C FILL COLOUR,STYLE
                CALL GSFACI(12)
                CALL GSFAIS(0)
C LINE COLOUR,STYLE
                CALL GSLN(1)
                CALL GSPLCI(11)
C MARKER TYPE, COLOUR
                CALL GSPMCI(11)
                CALL GSMK(1)
C TEXT REPRESENTATION
C     HEIGHT,COLOUR,ALIGNMENT
                CALL GSTXFP(-101,0)
                CALL GSCHH(YMX/32.50)
                CALL GSTXCI(13)
                CALL GSTXAL(1,5)
C
C ****************************************
C LOOPS TO DRAW GRAPHIC CHECKS OF GEOMETRY
C
C DRAWING NODES
C
                CALL GCLRWK(1,1)
                CALL GTX(-0.1*XMX,-0.1*YMX,' NODAL PLOT ')
                CALL GSCHH(YMX/48.75)
                DO 2 I=1,NN
                 PN(I,3)=PN(I,2)
                 WRITE(BUF,*) I
                 X(1)=PN(I,1)
                 Y(1)=PN(I,2)
                 CALL GTX(X(1)-XMX/10.0,Y(1),BUF)
2     CALL GPM(1,X,Y)
                PAUSE
                CALL GCLRWK(1,1)
C
```

```
C *** DRAWING ELEMENTS  ***
C
                CALL GSCHH(YMX/32.50)
                CALL GTX(-0.1*XMX,-0.1*YMX,' ELEMENT PLOT ')
                CALL GSCHH(YMX/48.75)
                DO 4 I=1,NE
                X(1)=PN(NEC(I,1),1)
                X(2)=PN(NEC(I,2),1)
                X(3)=PN(NEC(I,3),1)
                Y(1)=PN(NEC(I,1),2)
                Y(2)=PN(NEC(I,2),2)
                Y(3)=PN(NEC(I,3),2)
                X(4)=X(1)
                Y(4)=Y(1)
                CALL GFA(4,X,Y)
                X(4)=(X(1)+X(2)+X(3))/3.0
                Y(4)=(Y(1)+Y(2)+Y(3))/3.0
                WRITE(BUF,*) I
                CALL GTX(X(4)-XMX/10.0,Y(4)-YMX/50.0,BUF)
4     CONTINUE

C
C ***********************************************
C *** MAIN LOOP TO CALCULATE STIFFNESS MATRIX ***
C ***********************************************

C
C *** CALCULATING FLOW VARIABLES ***
C
15    ZI=PN(NHIT,2)
                ZO=PN(NHOT,2)
                HI=PN(NHIT,2)-PN(NHIB,2)
                HO=PN(NHOT,2)-PN(NHOB,2)
C
C *** CALCULATING FLOW IN AND OUT ***
C

                QI=2.0D+00*GRAV*(ZI-ZO)
                QI=QI/((HI/HO)**2 - 1.0D+00)
                QI=(QI)**0.5D+00
                QO=QI*HI/HO

C
C *** REGENERATE MIDSIDE NODES ***
C
                DO 35 I=1,NE
                PN(NEC(I,4),1)=( PN(NEC(I,1),1) + PN(NEC(I,2),1)
)/2.0D+00
                PN(NEC(I,4),2)=( PN(NEC(I,1),2) + PN(NEC(I,2),2)
)/2.0D+00
                PN(NEC(I,5),1)=( PN(NEC(I,3),1) + PN(NEC(I,2),1)
)/2.0D+00
                PN(NEC(I,5),2)=( PN(NEC(I,3),2) + PN(NEC(I,2),2)
)/2.0D+00
                PN(NEC(I,6),1)=( PN(NEC(I,3),1) + PN(NEC(I,1),1)
)/2.0D+00
35    PN(NEC(I,6),2)=( PN(NEC(I,3),2) + PN(NEC(I,1),2) )/2.0D+00
C
C *** RESET GLOBAL MATRIX ***
C
                DO 39 I=1,NN
                PN(I,3)=PN(I,2)
                QE(I)=0.0D+00
                DO 39 J=1,NN
39    KE(I,J)=0.0D+00

C
C *** LOOP THROUGH ALL OF ELEMENTS ***
C
                DO 55 I=1,NE
C
C *** RESETTING LOCAL VALUES ***
C
                DO 60 K=1,6
                QL(K)=0.0D+00
                DO 60 II=1,6
60    KL(K,II)=0.0D+00
C
C *** CALCULATING C's B's AND AREA ***
C
                CALL ABC(I,PN,NEC,A,B,C)
C
C *** CALCULATING KL MATRIX ***
C
                CALL LOCAL(KL,A,B,C)
C
C *** CALCULATING LOCAL LOAD VECTOR ***
C
                DO 65 J=1,NBC
                IF(NOE(J).EQ.I) THEN
                  IF(NOC(J).EQ.0) THEN
                   CALL SIDE(NOS(J),C,B,SL)
                   CALL FORCE(QL,NOS(J),-1.0D+00*QI,SL)
                  ELSE
                   CALL SIDE(NOS(J),C,B,SL)
                   CALL FORCE(QL,NOS(J),QO,SL)
                  END IF
                 END IF
65    CONTINUE
C
C *** ADDING LOCAL MATRIX INTO GLOBAL MATRIX ***
C
                DO 70 J=1,6
                 QE(NEC(I,J))=QE(NEC(I,J))+QL(J)
                DO 70 K=1,6

KE(NEC(I,J),NEC(I,K))=KE(NEC(I,J),NEC(I,K))+KL(J,K)
70    CONTINUE
C
C *** END OF MATRIX GENERATION ***
C
55    CONTINUE
C
C *** CONSTANT PRESSURE ON FREE SURFACE ***
C
                KE(NFN(LENGTH,1),NFN(LENGTH,1))=1.0D+30
                QE(NFN(LENGTH,1)) = 1.0D+32
C

                Q(NFN(LENGTH,1))=QO
                DUM= 1.0D+02
C
                DO 110 J=LENGTH,2,-1
                I=NFN(J-1,1)
                K=NFN(J,1)
C
                Q(I)=( 2.0*GRAV*(PN(K,2)-PN(I,2)) +Q(K)**2 )**0.50
                SL=( (PN(K,1)-PN(I,1))**2 + (PN(K,2)-PN(I,2))**2
)**0.50
                DUM = DUM - (Q(K)+Q(I))*SL/2.0
                KE(I,J) = 1.0D+30
110   QE(I) = DUM*1.0D+30

C
C
C *** THE SOLVER WE USE IS A GAUSS-JORDAN SOLVER ***
C
74     CALL GSCHH(YMX/32.50)
                CALL CLEARWORDS(XMX,YMX)
                CALL JORDAN(KE,QE,NN,XMX,YMX)
C
C LOOP TO CALCULATE VELOCITIES AND PRESSURES
C
                VMAX=0.0D+00
                DO 80 I=1,NE
C
C *** CALCULATING C's B's AND AREA ***
C
                CALL ABC(I,PN,NEC,A,B,C)
C
C *** MAKE LAGRANGIAN MATRIX ***
C
                CALL MAKEZLR(A,B,C,ZLR)
C
C *** LOOP AROUND EACH NODE ***
C
                DO 90 J=1,6
C
C *** MAKE SHAPE FUNCTION DERIVATIVES ***
C
                CALL MAKEZDL(J,ZDL)
C
C *** MAKE REAL WORLD DERIVATIVES ***
C
                CALL MAKEZDXY(ZLR,ZDL,ZDXY)
C
C *** CALCULATE U AND V VELOCITIES ***
                U(NEC(I,J))=0.0D+00
                V(NEC(I,J))=0.0D+00
C *** LOOP THROUGH NODES ***
                DO 100 K=1,6
                U(NEC(I,J))=U(NEC(I,J))+ZDXY(1,K)*QE(NEC(I,K))
                V(NEC(I,J))=V(NEC(I,J))+ZDXY(2,K)*QE(NEC(I,K))
100   CONTINUE
C
C *** CALCULATE MAX VELOCITY ***
                DUM=( U(NEC(I,J))**2 + V(NEC(I,J))**2 ) **0.5D+00
                IF (DUM.GT.VMAX) THEN
                                VMAX=DUM
                                JJ=(NEC(I,J))
```

```
                    END IF
90  CONTINUE
80  CONTINUE

C
C *** PLOTTING VELOCITIES ***
C
            CALL GCLRWK(1,1)
C
C DRAWING ELEMENTS
C
            CALL GSFACI(12)
            DO 94 I=1,NE
             X(1)=PN(NEC(I,1),1)
             X(2)=PN(NEC(I,2),1)
             X(3)=PN(NEC(I,3),1)
             Y(1)=PN(NEC(I,1),2)
             Y(2)=PN(NEC(I,2),2)
             Y(3)=PN(NEC(I,3),2)
             X(4)=X(1)
             Y(4)=Y(1)
94    CALL GFA(4,X,Y)
C
C DRAWING VELOCITY VECTORS
C
            CALL CLEARWORDS(XMX,YMX)
            CALL GTX(-0.1*XMX,-0.1*YMX,' VELOCITY VECTOR
PLOT')
            CALL GSPLCI(14)
            CALL GSTXCI(13)
            DO 111 I=1,NN
            X(1)=PN(I,1)
            Y(1)=PN(I,2)
            X(2)=X(1)+(U(I)/10.)*xmx/vmax
            Y(2)=Y(1)+(V(I)/10.)*xmx/vmax
            CALL GPL(2,X,Y)
            IF(I.EQ.JJ) THEN
              S=PN(I,1)
              T=PN(I,2)
              END IF
111   CONTINUE
            CALL GTX(S,T,'MAX')
C
C *** NORMAL VELOCITY ON FREE SURFACE ***
C
             CALL GSPMCI(3)
             SUM=0.0
C LOOP THRU SURFACE
             DO 125 I=2,LENGTH
C DEPTH OF NETWORK
             DUM=PN(NFN(I,1),2)-PN(NFN(I,DEPTH),2)
C SET SCALE FACTORS
             DO 127 J=1,DEPTH
             TEMP=PN(NFN(I,1),2)-PN(NFN(I,J),2)
             TEMP=TEMP/DUM
127    SF(J)=1.0-TEMP
C FIND FLOW AND SURFACE SLOPES
C
C THIS VERSION USES AN MIDSIDED NODE VELOCITY SLOPE
C
             DUM=V(NVA(I-1))/U(NVA(I-1))
             TEMP=(PN(NFN(I,1),2)-PN(NFN(I-
1,1),2))/(PN(NFN(I,1),1)-PN(NFN
  &(I-1,1),1))
             DIFF=(DUM-TEMP)
             SUM=SUM+0.2*DIFF*(PN(NFN(I,1),1)-PN(NFN(I-
1,1),1))
C MOVE NETWORK LINE
             DO 128 J=1,DEPTH
128    PN(NFN(I,J),2)=PN(NFN(I,J),2)+SUM*SF(J)

C
C PLOT NEW NODE POSITIONS
C
             DO 130 J=1,DEPTH
             X(1)=PN(NFN(I,J),1)
             Y(1)=PN(NFN(I,J),2)
130    CALL GPM(1,X,Y)

125    CONTINUE
C
C IN SURFACE BY PRESSURE HEIGHT VARYING
C
C FIND PRESSURES
            DUM=QI*QI*DENS/2.0 + PN(1,2)*DENS*GRAV
             DO 120 I=1,NN
             Q(I)=( U(I)**2.0 + V(I)**2.0 )**0.50
120   P(I)=DUM - Q(I)*Q(I)*DENS/2.0 - PN(I,2)*DENS*GRAV
C
C MOVE SURFACE
            DO 121 I=1,WIDTH
121   PN(NIS(I),2)=PN(NIS(I),2)+P(NIS(I))/GRAV/DENS
C
C *** CHECK FOR CONVERGENCE ***
C
             MAX=0.0D+00
             DO 115 I=2,LENGTH
             TEST=DABS(PN(NFN(I,1),2)-PN(NFN(I,1),3))
             IF(TEST.GT.MAX) MAX=TEST
115    CONTINUE
             DO 116 I=1,WIDTH
             TEST=DABS(PN(NIS(I),2)-PN(NIS(I),3))
             IF(TEST.GT.MAX) MAX=TEST
116    CONTINUE
             WRITE(BUF,*) MAX
             CALL GTX(0.0,1.01*YMX,BUF)
             IF(MAX.LT.CONCRI) GOTO 98
             IF(ITER.GT.ITLIM) GOTO 97
            ITER=ITER+1
            GOTO 15
C
C *** CONVERGENCE HAS HAPPENED ***
C
98    CALL CLEARWORDS(XMX,YMX)
            CALL GTX(-0.1*XMX,-0.1*YMX,' CONVERGENCE
HAS HAPPENED')
            GOTO 135
C
C *** END OF ITERATIONS ***
C
97    CALL CLEARWORDS(XMX,YMX)
            CALL GTX(-0.1*XMX,-0.1*YMX,' END OF
ITERATIONS')
C
C *** CALCULATE PRESSURE LOOP ***
C
135   PAUSE
             WRITE(4,*)' ITERATION ',ITER
             WRITE(4,*)' Vin  =',QI
             WRITE(4,*)' Vout =',QO
             WRITE(4,*)
             WRITE(4,*)' NODAL VALUES '
             WRITE(4,*)
             DO 137 I=1,NN
             IF( P(I).LT.PMIN ) PMIN=P(I)
             IF( P(I).GT.PMAX ) PMAX=P(I)
             WRITE(4,*)'  FOR NODE #',I
             WRITE(4,*)'  X VELOCITY IS (m/s)',U(I)
             WRITE(4,*)'  Y VELOCITY IS (m/s)',V(I)
             WRITE(4,*)'  PRESSURE IS (kPa)',P(I)/1000.0
137   WRITE(4,*)
C
C *** WRITE NEW GEOMETRY FILE ***
C
    WRITE(5,*) NE,NN,NBC
    WRITE(6,*) NE,NN
    DO 140 J=1,NE
140   WRITE(5,141)
J,NEC(J,1),NEC(J,4),NEC(J,2),NEC(J,5),NEC(J,3),NEC(J
  &,6)
141   FORMAT(7I8)
    DO 142 J=1,NN
    P(J)=P(J)/1000.0D+00
    WRITE(6,*) J,PN(J,1),PN(J,2)
142   WRITE(5,*) J,PN(J,1),PN(J,2)
    PMIN=PMIN/1000.0
    PMAX=PMAX/1000.0
    DO 144 J=1,NN
144   WRITE(6,*) U(J),V(J)
    CLOSE(UNIT=5)
    CLOSE(UNIT=6)
143   FORMAT(F7.2)
C
C *** PRESSURE PLOTTING ROUTINE ***
C
             CALL GCLRWK(1,1)
             CALL GSFAIS(3)
             X(1)=20.0
             X(2)=X(1)
             X(3)=21.3
             X(4)=X(3)
             X(5)=X(1)
             Y(1)=16.0
             Y(2)=5.0
             Y(3)=5.0
             Y(4)=16.0
             Y(5)=Y(1)
             CALL GSFACI(12)
             CALL GFA(5,X,Y)
             CALL GSFAIS(1)
```

```
C
C  SET SCALE
C
                PSCA=(PMAX-PMIN)/FLOAT(CCN)
                WRITE(4,*)' PMAX=',PMAX
                WRITE(4,*)' PMIN=',PMIN
                WRITE(4,*)' PSCA=',PSCA
C
C LOOP THRU ELEMENTS
C
                CALL GSTXFP(-101,0)
                DO 155 I=1,NE
C       write(4,*)' element #=',i
C
C LOOP THRU CONTOURS
C
                DO 160 J=1,CCN
                CALL GSFACI(J)
                K=0
                SH=PMIN + PSCA*FLOAT(J)
                SL=SH-PSCA
C       write(4,*)' search values=',sl,sh
C
C LOOP THRU SIDES OF ELEMENT
C
                DO 170 L=1,3
C SET NODE NUMBERS
                  NA=NEC(I,L)
                  NB=NEC(I,L+1)
                  IF(L.EQ.3) NB=NEC(I,1)
                  PA=P(NA)
                  PB=P(NB)
                  IF(P(NA).EQ.PMAX) THEN
                                S=PN(NA,1)
                                T=PN(NA,2)
                                CALL GTX(S,T,'MAX')
                                END IF
                  IF(P(NA).EQ.PMIN) THEN
                                S=PN(NA,1)
                                T=PN(NA,2)
                                CALL GTX(S,T,'MIN')
                                END IF
C         write(4,*)' pressures are=',pa,pb
C
C IF STRUCTURE FOR FINDING VALUES
C
C IF BOTH OUT OF RANGE
                IF(PA.LT.SL.AND.PB.LT.SL) THEN
C       WRITE(4,*)' BOTH OUT '
                ELSE IF(PB.GT.SH.AND.PA.GT.SH) THEN
C       WRITE(4,*)' BOTH OUT '
C IF BOTH IN RANGE
                ELSE
IF(PA.LE.SH.AND.PA.GE.SL.AND.PB.LE.SH.AND.PB.GE.SL) THEN
C       WRITE(4,*)' BOTH IN '
                  K=K+1
                  X(K)=PN(NA,1)
                  Y(K)=PN(NA,2)
C IF THRU RANGE 1
                ELSE IF(PA.LT.SL.AND.PB.GT.SH) THEN
C       WRITE(4,*)' THRU 1'
                  K=K+1
                  CALL INPL(PA,SL,PB,X,Y,K,PN(NA,1),PN(NA,2),
     &                        PN(NB,1),PN(NB,2))
                  K=K+1
                  CALL INPL(PA,SH,PB,X,Y,K,PN(NA,1),PN(NA,2),
     &                        PN(NB,1),PN(NB,2))
C IF THRU RANGE 2
                ELSE IF(PB.LT.SL.AND.PA.GT.SH) THEN
C       WRITE(4,*)' THRU 2 '
                  K=K+1
                  CALL INPL(PA,SH,PB,X,Y,K,PN(NA,1),PN(NA,2),
     &                        PN(NB,1),PN(NB,2))
                  K=K+1
                  CALL INPL(PA,SL,PB,X,Y,K,PN(NA,1),PN(NA,2),
     &                        PN(NB,1),PN(NB,2))
C IF PA IN BUT PB OUT
                ELSE IF(PA.GE.SL.AND.PA.LE.SH) THEN
C       WRITE(4,*)' PA IN  PB OUT '
                  K=K+1
                  X(K)=PN(NA,1)
                  Y(K)=PN(NA,2)
                   ST=SL
                   IF(PB.GT.SH) ST=SH
                   K=K+1
                  CALL INPL(PA,ST,PB,X,Y,K,PN(NA,1),PN(NA,2),
     &                        PN(NB,1),PN(NB,2))
C           WRITE(4,*)' IN MAIN ',X(K),Y(K)
C IF PB IN BUT PA OUT
                ELSE IF(PB.GE.SL.AND.PB.LE.SH) THEN
C       WRITE(4,*)' PB IN  PA OUT '
                    K=K+1
                     ST=SL
                     IF(PA.GT.SH) ST=SH
                    CALL INPL(PA,ST,PB,X,Y,K,PN(NA,1),PN(NA,2),
     &                        PN(NB,1),PN(NB,2))
                END IF
C     NEXT EDGE OF ELEMENT
170   CONTINUE
C     INSURE CLOSED POLYGON??
                  IF(K.GT.0) THEN
                  K=K+1
                  X(K)=X(1)
                  Y(K)=Y(1)
                  END IF
C
C PLOTTING A CONTOUR
C
                IF(K.GT.0) CALL GFA(K,X,Y)
                K=0
C     NEXT CONTOUR
160   CONTINUE
C     NEXT ELEMENT
155   CONTINUE
C
C DRAWING ELEMENTS
C
                CALL GSPLCI(12)
                CALL GSCHH(YMX/37.5)
                CALL GTX(-0.1*XMX,-0.1*YMX,' PRESSURE
CONTOUR PLOT ')
                DO 174 I=1,NE
                X(1)=PN(NEC(I,1),1)
                X(2)=PN(NEC(I,2),1)
                X(3)=PN(NEC(I,3),1)
                Y(1)=PN(NEC(I,1),2)
                Y(2)=PN(NEC(I,2),2)
                Y(3)=PN(NEC(I,3),2)
                X(4)=X(1)
                Y(4)=Y(1)
C       CALL GPL(4,X,Y)
174   CONTINUE
C
C DRAW LEGEND
C
C SET X COORDS
                XW=1.01*XMX
                X(1)=1.05*XMX
                X(2)=1.10*XMX
                X(3)=X(2)
                X(4)=X(1)
                X(5)=X(1)
                CALL GSCHH(YMX/37.5)
C
C LOOP THRU CONTOURS
C
                CALL GSCHH(YMX/4.0/FLOAT(CCN))
                DO 180 I=1,CCN
                Y(3)=FLOAT(I)*YMX/FLOAT(CCN)
                Y(1)=Y(3)-YMX/2.0/FLOAT(CCN)
                Y(2)=Y(1)
                Y(4)=Y(3)
                Y(5)=Y(1)
                WRITE(BUF,143) (PMIN+I*PSCA)
                CALL GTX( XW, Y(3), BUF)
                CALL GSFACI(I)
180   CALL GFA(5,X,Y)
                WRITE(BUF,143) PMIN
                CALL GTX( XW , 0.0 ,BUF)
                CALL GTX(XW*0.95,-0.05*YMX,'units = kPa')
                 PAUSE

                CLOSE(UNIT=4)
                CALL CLOSE
C
                STOP
                END
C
C*************************************************************
                SUBROUTINE CLEARWORDS(XMX,YMX)
                REAL XMX,X(4),Y(4),YMX
                X(1)=-0.1*XMX
                X(2)=X(1)
                X(3)=1.1*XMX
                X(4)=X(3)
                Y(1)=-0.1
                Y(2)=-0.1*YMX
                Y(3)=Y(2)
                Y(4)=Y(1)
                CALL GSFACI(15)
```

```
            CALL GSFAIS(1)
            CALL GFA(4,X,Y)
            CALL GSFAIS(0)
            RETURN
            END
C*************************************************************
************
            SUBROUTINE MAKEZLR(A,B,C,ZLR)
            REAL*8 A,B(3),C(3),ZLR(2,3)
            A=A*2.0D+00
            ZLR(1,1)=B(1)/A
            ZLR(1,2)=B(2)/A
            ZLR(1,3)=B(3)/A
            ZLR(2,1)=C(1)/A
            ZLR(2,2)=C(2)/A
            ZLR(2,3)=C(3)/A
            RETURN
            END
C
C*************************************************************
*************
            SUBROUTINE MAKEZDL(J,ZDL)
            INTEGER J
            REAL*8 ZDL(3,6),L1,L2,L3
            L1=0.0D+00
            L2=0.0D+00
            L3=0.0D+00
            IF(J.EQ.1) THEN
            L1=1.0D+00
            ELSE IF(J.EQ.2) THEN
            L2=1.0D+00
            ELSE IF(J.EQ.3) THEN
            L3=1.0D+00
            ELSE IF(J.EQ.4) THEN
            L1=0.5D+00
            L2=0.5D+00
            ELSE IF(J.EQ.5) THEN
            L2=0.5D+00
            L3=0.5D+00
            ELSE
            L1=0.5D+00
            L3=0.5D+00
            END IF
            ZDL(1,1)=4.0D+00*L1-1.0D+00
            ZDL(2,2)=4.0D+00*L2-1.0D+00
            ZDL(3,3)=4.0D+00*L3-1.0D+00
            ZDL(1,4)=4.0D+00*L2
            ZDL(1,6)=4.0D+00*L3
            ZDL(2,4)=4.0D+00*L1
            ZDL(2,5)=4.0D+00*L3
            ZDL(3,5)=4.0D+00*L2
            ZDL(3,6)=4.0D+00*L1
            RETURN
            END
C
C*************************************************************
************
            SUBROUTINE MAKEZDXY(ZLR,ZDL,ZDXY)
            REAL*8 ZLR(2,3),ZDL(3,6),ZDXY(2,6)
            INTEGER I,J
            DO 5 I=1,6
            ZDXY(1,I)=0.0D+00
5     ZDXY(2,I)=0.0D+00
            DO 10 I=1,6
            DO 10 J=1,3
            ZDXY(1,I)=ZDXY(1,I)+ZLR(1,J)*ZDL(J,I)
            ZDXY(2,I)=ZDXY(2,I)+ZLR(2,J)*ZDL(J,I)
10    CONTINUE
            RETURN
            END

C
C*************************************************************
************
            SUBROUTINE ABC(I,PN,NEC,A,B,C)
C THIS SUBROUTINE CALCULATES THE DISTANCES
C USED IN CALCULATING K MATRIX AND Q VECTOR
            REAL*8 PN(190,2),C(3),B(3),A
            INTEGER I,NEC(100,6)
            C(3)= PN(NEC(I,2),1) - PN(NEC(I,1),1)
            C(1)= PN(NEC(I,3),1) - PN(NEC(I,2),1)
            C(2)= PN(NEC(I,1),1) - PN(NEC(I,3),1)
            B(1)= PN(NEC(I,2),2) - PN(NEC(I,3),2)
            B(2)= PN(NEC(I,3),2) - PN(NEC(I,1),2)
            B(3)= PN(NEC(I,1),2) - PN(NEC(I,2),2)
            A=DABS(C(1)*B(3)-C(3)*B(1))/2.0D+00
            RETURN
            END
C
C*************************************************************
************
            SUBROUTINE LOCAL(KL,A,B,C)
C THIS SUBROUTINE CALCULATES THE
C LOCAL K MATRIX
            REAL*8 KL(6,6),A,C(3),B(3),S(3,3)
            INTEGER I,J
            DO 15 I=1,3
            DO 15 J=1,3
15    S(I,J)=(C(I)*C(J)+B(I)*B(J))/A/12.0D+00
C
            KL(1,1)=3.0D+00*S(1,1)
            KL(1,2)=-1.0D+00*S(1,2)
            KL(1,3)=-1.0D+00*S(1,3)
            KL(1,4)=4.0D+00*S(1,2)
            KL(1,5)=0.0D+00
            KL(1,6)=4.0D+00*S(1,3)
C
            KL(2,2)=3.0D+00*S(2,2)
            KL(2,3)=-1.0D+00*S(2,3)
            KL(2,4)=4.0D+00*S(1,2)
            KL(2,5)=4.0D+00*S(2,3)
            KL(2,6)=0.0D+00
C
            KL(3,3)=3.0D+00*S(3,3)
            KL(3,4)=0.0D+00
            KL(3,5)=4.0D+00*S(2,3)
            KL(3,6)=4.0D+00*S(1,3)
C
            KL(4,4)=8.0D+00*(S(3,3)-S(1,2))
            KL(4,5)=8.0D+00*S(1,3)
            KL(4,6)=8.0D+00*S(2,3)
C
            KL(5,5)=8.0D+00*(S(1,1)-S(2,3))
            KL(5,6)=8.0D+00*S(1,2)
C
            KL(6,6)=8.0D+00*(S(2,2)-S(3,1))
C
C FILLING IN LOWER HALF OF MATRIX
C
            DO 20 I=2,6
            DO 20 J=1,I-1
20    KL(I,J)=KL(J,I)
            RETURN
            END
C
C*************************************************************
            SUBROUTINE SIDE(NOS,C,B,SI)
C THIS SUBROUTINE CALCULATES THE
C LENGTH OF THE SIDE NEED FOR THE
C BOUNDARY CONDITION CALCULATION
C
            REAL*8 C(3),B(3),SI
            INTEGER NOS
            IF(NOS.EQ.1) THEN
             SI=( C(3)**2 + B(3)**2 ) **0.5D+00
            ELSE IF(NOS.EQ.2) THEN
             SI=( C(1)**2 + B(1)**2 ) **0.5D+00
            ELSE
             SI=( C(2)**2 + B(2)**2 ) **0.5D+00
            END IF
            RETURN
            END

C*************************************************************
*******
            SUBROUTINE JORDAN(A,Y,N,xmx,ymx)
            real*8 A(190,190),T,PIVOT,Y(190),B(190)
            REAL xmx,ymx
            INTEGER I,N,NCOL
            CHARACTER*12 BUF
C-------------------------------------------------------
C  This subroutine inverts an n x n matrix on top
C  of itself using Gauss-Jordan elimination technique
C-------------------------------------------------------
C     -.2   .1  .3  .5 .7
            CALL GSTXFP(1,0)
            CALL GTX(-0.10*XMX,-0.1*YMX,'SOLVER
WORKING')
            CALL GTX(0.15*XMX,-0.1*YMX,'finish')
            write(buf,*) n
            CALL GTX(0.25*XMX,-0.1*YMX,buf)
            CALL GTX(0.55*XMX,-0.1*YMX,'present')
   do 50 i=1,n
   WRITE(BUF,*) I
   CALL GTX(0.65*XMX,-0.1*YMX,buf)
   pivot=A(i,i)
   T=1.0d0/pivot
   do 10 ncol=1,n
10   A(i,ncol)=A(i,ncol)/pivot
```

```
      do 40 k=1,n
      if(k-i) 20,40,20
 20     pivot=A(k,i)
                do 30 ncol=1,n
 30     A(k,ncol)=A(k,ncol)-A(i,ncol)*pivot
                A(k,i)=-pivot*T
 40   continue
 50   A(i,i)=T
      CALL GSTXFP(-101,0)
C
C MULTILPY TO GET ANSWER
C
      DO 11 I=1,N
      B(I)=0.0D+00
      DO 31 J=1,N
 31   B(I)=B(I)+A(I,J)*Y(J)
 11   CONTINUE
C
      DO 41 I=1,N
 41   Y(I)=B(I)
C
      RETURN
      END
C*************************************************************
*******
      SUBROUTINE OPEN(XMX,YMX)
C     OPEN GKS AND SET VIEWPORT AND WINDOW
      INTEGER*2 ERR,DCUNIT,XRAS,YRAS
      REAL XDCMAX,YDCMAX,SCALE,XNDC,YNDC,XMX,YMX
      OPEN (14,FILE='ERRORS')
      CALL GOPKS (14,1024)
      CALL GOPWK (0,0,0)
      CALL GOPWK (1,0,1)
      CALL GACWK (0)
      CALL GACWK (1)
      CALL GQDSP
(1,ERR,DCUNIT,XDCMAX,YDCMAX,XRAS,YRAS)
      IF (XDCMAX.GT.YDCMAX) THEN
                SCALE=XDCMAX
      ELSE
                SCALE=YDCMAX
      END IF
      XNDC=XDCMAX/SCALE
      YNDC=YDCMAX/SCALE
      CALL GSWN (1,-0.1*XMX,1.1*XMX,-0.1*YMX,1.1*YMX)
      CALL GSVP (1,0.0,XNDC,0.0,YNDC)
      CALL GSWKWN (1,0.0,XNDC,0.0,YNDC)
      CALL GSWKVP (1,0.0,XDCMAX,0.0,YDCMAX)
      CALL GSELNT (1)
      CALL GSTXFP (-101,2)
      RETURN
      END

      SUBROUTINE CLOSE

*     CLOSE GKS

      CALL GDAWK (0)
      CALL GDAWK (1)
      CALL GCLWK (0)
      CALL GCLWK (1)
      CALL GCLKS ()
      CLOSE (14)
      RETURN
      END


C*************************************************************
                SUBROUTINE FORCE(QL,SN,VAL,SI)
                REAL*8 VAL,QL(6),SI
                INTEGER SN
                  IF(SN.EQ.1) THEN
                   QL(1)=SI*VAL/6.0D+00
                   QL(2)=QL(1)
                   QL(4)=4.0D+00*QL(1)
                  ELSE IF (SN.EQ.2) THEN
                   QL(2)=SI*VAL/6.0D+00
                   QL(3)=QL(2)
                   QL(5)=4.0D+00*QL(2)
                  ELSE
                   QL(3)=SI*VAL/2.0D+00
                   QL(1)=QL(3)
                   QL(6)=4.0D+00*QL(3)
                  END IF
                  RETURN
                  END
C
C*************************************************************
*****
C
      SUBROUTINE INPL(PA,SV,PB,X,Y,K,XA,YA,XB,YB)
C
C THIS SUBROUTINE IS A LINEAR INTERPOLATER
C TO FIND THE CONTOUR CORNER LOCATIONS
C
                REAL*8 PA,SV,PB,XA,XB,YA,YB
                REAL X(7),Y(7)
                X(K)=(XA+(XB-XA)*(SV-PA)/(PB-PA))
                Y(K)=(YA+(YB-YA)*(SV-PA)/(PB-PA))
                RETURN
                END
```

Appendix B3

INPUT PROGRAM

```
                PROGRAM MHYFEC_INPUT
C
C THIS IS THE PREPROCESSOR FOR Manitoba HYdro Finite Element
Code
C
C     WRITTEN BY:
C          DOUGLAS G. SCOTT
C          MECH. ENG. GRADUATE STUDENT
C          NOVEMBER/DECEMBER/JANUARY 1991/1992
C
C     IT IS DESIGNED TO DEVELOPE FINITE ELEMENT MODELS
OF
C     2 AND 3 DIMENSIONAL STRUCTURES AS WELL AS AXIS-
SYMMETRIC
C     OBJECTS. AS OF JANUARY 1992, THE 2D AND 3D HAVE
BEEN
C     IMPLEMENTED. AXISYM WILL DEPEND ON FURTHER
DEVELOPEMENT
C     AS IT NOT REQUIRED BY HYDRO AND NO TIME FOR
WRITING EXISTS.
C
C
C     VARIABLE LIST
C
C     COUNTERS
C
C     NOG    NUMBER OF GROUPS
C     NOE    NUMBER OF ELEMENTS
C     NON    NUMBER OF NODES
C     NBC    NUMBER OF BOUNDARY CONDITIONS
C
C     POINTERS
C
C     NPRESG    GROUP#
C     NPRESE    ELEMENT#
C     NPRESN    NODE#
C     NPDOF     DEGREE OF FREEDOM#
C     NODIM     NUMBER OF DIMENSIONS
C
C     ELEMENT PROPERTIES
C
C     NUMMAT  NUMBER OF MATERIAL CONSTANTS(TYPE#)
C     NUMNOD  NUMBER OF NODES IN ELEMENT  (TYPE#)
C     NUMCOL  NUMBER OF COLOUR OF ELEMENT (TYPE#)
C     NEC     ELEMENT CONECTIVITY,G# (9)
C
C     GROUP
C
C     NOEIG   NUMBER OF ELEMENTS IN THIS GROUP
C     NOETY   NUMBER OF ELEMENT TYPE IN THIS GROUP
C     NOGAC   ACTIVITY STATUS OF GROUP (SHOW=1/NO=-1)
C
C     GRAPHIC FLAGS
C
C     KNN     FLAG FOR NODE NUMBER 1=ON -1=OFF
C     KEN     FLAG FOR ELEM NUMBER 1=ON -1=OFF
C     KNS     FLAG FOR NODE GRAPH  1=ON -1=OFF
C     KES     FLAG FOR ELEM GRAPH  1=ON -1=OFF
C     MOKB    FLAG FOR MOUSE OR KEYBOARD INPUT
C                  1=MOUSE 0=KEYBOARD
c     KGRAV   1 FOR GRAVITY 0 FOR NOT
C     NOTE: K*S MUST=1 FOR NUMBERING TO FUNCTION
C     KSAV    FLAG FOR SAVED 1=SAVED 0=NOT
C     KSOW    FLAG SOLID OR WIREFRAME 0=WIRE,1=SOLID
C     KBC     BOUNDARY CONDITIONS SHOW=1/NO=-1
C     KFOR    FORCES SHOW=1/NO=-1
C
C     BOUNDARY CONDITIONS
C
C     ID     BOUNDARY CONDITION ARRAY  6dof
C     IDL    B.C. LOCATIONS
C     SD     SPECIFIED DISPLACEMENT ARRAY ( NOT USED! )
C
C     OTHERS
C     PN     NODAL POSITIONS(5) (3D + 2 FOR GRAPHIC)
C     PMAT   MATERIALS PROPERTIES
C     F      NODAL FORCE ARRAY
C     INF    NODAL FLAG IF EXISTS 0=NO
C     IEF    ELEMT FLAG IF EXISTS 1=YES
C
C 700 NODES  PN(700,5) (X,Y,Z,XX,YY)
C               X Y Z ARE REAL COORDINATES
C               XX YY ARE MAPPED ONTO SCREEN COORDS.
C 700 ELEMENTS   NEC(700,9) (8 NODES, GROUP#)
C 25 GROUPS AND MATERIALS #ELM IN GROUP, ELEMTYPE,
MAT#, ACTIVITY
C 700 FORCES  F(NODE# , 6 DOF FORCES)
C 200 BOUNDARY CONDITIONS  ID(6 DOF FIX), IDL(NODE WHICH
FIX)

C
C     DECLARATIONS
C
            CHARACTER*20 TITLE,BUF
            INCLUDE 'NCOMMON.F'
C
C     OPEN GKS ENVIROMENT
C
            XMN=0.0
            YMN=0.0
            XMX=30.0
            YMX=22.0
            XDIS=XMX
            YDIS=YMX
            CALL OPEN(XMN,YMN,XMN,YMX)
            CALL GSELNT (2)
C
C     INCLUDE COLOR,BUTTON,INFO
            INCLUDE 'COLOUR.F'
C
C     DRAW BACKGROUND
C
            CALL BOX(-1.0,31.0,-21.0,100.0,GREY,1)

            INCLUDE 'BUTTONS.F'
            INCLUDE 'NEWINFO.F'
            CALL GSTXCI(14)
            CALL GSCHH(3.0)
            CALL GTX(1.0,-20.0,'MHYFECS INPUT')
C
C     SET DEFAULTS
C
            TITLE='UNTITLED'
            KNN=-1
            KEN=-1
            KNS=1
            KES=1
            MOKB=0
            AS=0.0
            AP=1.57079
            KSAV=1
            KSOW=1
            KBC=-1
            KFOR=-1
            NODIM=1
            KGRAV=0
            DO 3 I=1,700
3     NOOSE(I)=I
C
C *** TYPE OF PROBLEM SETUP ROUTINE ***
C
            CALL GSWN (1,XMN,XMX,YMN,YMX)
            CALL GSELNT (1)
            CALL GSCHH ( YDIS/35.0 )
c     CALL BOX(xmn,xmx,ymn,ymx,black,1)
            CALL GTX(1.0,19.0,' MHYFECS STARTING VARIABLE
SETUP MENU ')
C PROBLEM TYPE
17    CALL GSTXCI(WHITE)
            CALL BOX(xmn,xmx,ymn,18.9,black,1)
            CALL GTX(2.0,17.0,' ENTER PROBLEM TYPE')
            CALL GTX(2.0,16.0,'   1=SAVED FILE  ')
            CALL GTX(2.0,15.0,'   2=2D 3=3D   ')
            CALL GRQST(1,1,K,J,BUF)
            READ(BUF,*,ERR=99) I
            CALL BOX(xmn,xmx,ymn,18.9,black,1)
           IF(I.GT.3.OR.I.LT.1) THEN
            CALL GSTXCI(RED)
            CALL GTX(2.0,17.0,'  YOU MUST PICK 1-3')
            CALL GTX(2.0,16.0,'  ENTER TO CONTINUE')
            CALL GRQST(1,1,K,J,BUF)
            CALL BOX(xmn,xmx,ymn,18.9,black,1)
            GOTO 17
            END IF
           NODIM=I
            IF(NODIM.EQ.1) THEN
             CALL GTX(2.0,17.0,' SAVED FILE  ')
            ELSEIF(NODIM.EQ.2) THEN
             CALL GTX(2.0,17.0,' 2 DIMENSIONAL')
```

```
                ELSEIF(NODIM.EQ.3) THEN
                 CALL GTX(2.0,17.0,' 3 DIMENSIONAL')
                END IF

                CALL GTX(2.0,15.0,' DO YOU WANT GRAVITY
FORCES? ')
                CALL GTX(2.0,14.0,' NOTE: ALWAYS IN -Y
DIRECTION ')
                CALL GTX(2.0,13.0,'    Y for yes  N for no    ')
                CALL GRQST(1,1,K,J,BUF)
                BUF=BUF(1:1)
                CALL BOX(xmn,xmx,ymn,16.9,black,1)
                IF(BUF.EQ.'Y'.OR.BUF.EQ.'y') THEN
                 KGRAV=1
                 CALL GTX(2.0,15.0,' GRAVITY ON ')
                ELSE
                 KGRAV=0
                 CALL GTX(2.0,15.0,' GRAVITY OFF)
                END IF
11     CALL GSTXCI(WHITE)
                CALL BOX(xmn,xmx,ymn,14.9,black,1)
                CALL GTX(2.0,13.0,' WHAT TYPE OF NODE MARKER
')
                CALL GTX(2.0,12.0,' PICK  1   2   3   4 ')
                CALL GSMK(1)
                CALL GPM(1,4.0,11.0)
                CALL GSMK(2)
                CALL GPM(1,6.0,11.0)
                CALL GSMK(3)
                CALL GPM(1,8.0,11.0)
                CALL GSMK(-1)
                CALL GPM(1,10.0,11.0)
                CALL GRQST(1,1,K,J,BUF)
                READ(BUF,*,ERR=99) KIJ
                CALL BOX(xmn,xmx,ymn,14.9,black,1)
               IF(KIJ.GT.4.OR.KIJ.LT.1) THEN
                CALL GSTXCI(RED)
                CALL GTX(2.0,13.0,'   YOU MUST PICK 1-4')
                CALL GTX(2.0,12.0,'   ENTER TO CONTINUE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(xmn,xmx,ymn,14.9,black,1)
                GOTO 11
                END IF
                IF(KIJ.EQ.4) KIJ=-1
                CALL GSMK(KIJ)
                CALL GPM(1,4.0,14.0)
               GOTO 44

C ERROR CONTROL
99     CALL GSTXCI(RED)
                CALL BOX(xmn,xmx,ymn,18.9,black,1)
                CALL GTX(2.0,15.,'  ERROR IN READING DATA! ')
                CALL GTX(2.0,13.,'    ENTER TO CONTINUE    ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(xmn,xmx,ymn,18.9,black,1)
                GOTO 17
C
44     CALL GTX(2.0,13.0,' ARE THESE CHOICES OK? ')
                CALL GTX(2.0,12.0,'  Y for yes  N for no  ')
                CALL GRQST(1,1,K,J,BUF)
                BUF=BUF(1:1)
                CALL BOX(xmn,xmx,ymn,16.9,black,1)
                IF(BUF.NE.'Y'.AND.BUF.NE.'y') GOTO 17
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
C
C TITLE AND ICON HIGHLIGHT
C
45     CALL GSELNT (2)
                CALL GSLWSC(3.0)
                CALL GSPLCI(PURPLE)
                CALL GSCHH(3.0)
                CALL GSTXCI(ORANGE)
                CALL GTX(1.0,95.0,TITLE)
                KOMT=KIJ
                CALL GSMK(KIJ)
                CALL GSCHH(1.5)

C
C *** START OF MAIN ROUTINE ***
C
50     CALL GRQPK(1,2,II,SEG,JJ)
C
C     MAIN PICK ROUTINE
C
C
                IF(SEG.EQ.BVAL) THEN
                        CALL SEARCH()
                        CALL DRAW(1,'  ',0)
                ELSEIF(SEG.EQ.BU) THEN
                        CALL ROTATE(0.0,-1.0)
                ELSEIF(SEG.EQ.BD) THEN
                        CALL ROTATE(0.0,1.0)
                ELSEIF(SEG.EQ.BR) THEN
                        CALL ROTATE(1.0,0.0)
                ELSEIF(SEG.EQ.BL) THEN
                        CALL ROTATE(-1.0,0.0)
                ELSEIF(SEG.EQ.REDRAW) THEN
                        CALL DRAW(1,'  ',1)
                ELSEIF(SEG.EQ.BPC) THEN
                        CALL CENTER()
                ELSEIF(SEG.EQ.BZI) THEN
                        CALL ZOOMIN()
                ELSEIF(SEG.EQ.BZO) THEN
                        CALL ZOOMOUT()
                ELSEIF(SEG.EQ.BRES) THEN
                        CALL RESET()
                ELSEIF(SEG.EQ.BEON) THEN
                        KES=-KES
                ELSEIF(SEG.EQ.BENON) THEN
                        KEN=-KEN
                ELSEIF(SEG.EQ.BNON) THEN
                        KNS=-KNS
                ELSEIF(SEG.EQ.BSOW) THEN
                    IF(KSOW.EQ.0) THEN
                        KSOW=1
                        ELSE
                        KSOW=0
                        END IF
                ELSEIF(SEG.EQ.BNNON) THEN
                        KNN=-KNN
                ELSEIF(SEG.EQ.BBC) THEN
                        KBC=-KBC
                ELSEIF(SEG.EQ.BFOR) THEN
                        KFOR=-KFOR
                ELSEIF(SEG.EQ.BLIST) THEN
                        CALL LIST()
                        CALL DRAW(1,'  ',0)
                ELSEIF(SEG.EQ.NODE) THEN
                        CALL NODESUB()
                ELSEIF(SEG.EQ.ELEM) THEN
                        CALL ELEMSUB()
                ELSEIF(SEG.EQ.FILE) THEN
                        CALL FILESUB(TITLE)
                ELSEIF(SEG.EQ.BMETA) THEN
                        CALL META()
                ELSEIF(SEG.EQ.BMOUSE) THEN
                        MOKB=1
                ELSEIF(SEG.EQ.BKEYB) THEN
                        MOKB=0
                END IF
                GOTO 50
C
C *** END OF MAIN ROUTINE ***
C
                END
C*************************************************************
*******

    SUBROUTINE CLOSE

*   CLOSE GKS
    CALL GCLRWK(1,1)
    CALL GDAWK (0)
    CALL GDAWK (1)
    CALL GCLWK (2)
    CALL GCLWK (0)
    CALL GCLWK (1)
    CALL GCLKS ()
    CLOSE (14)
    RETURN
    END


C*************************************************************
*******
    SUBROUTINE OPEN(XMN,YMN,XMX,YMX)
C   OPEN GKS AND SET VIEWPORT AND WINDOW
    INTEGER*2 ERR,DCUNIT,XRAS,YRAS
    REAL
XDCMX,YDCMX,SCALE,XNDC,YNDC,XMN,XMX,YMN,YMX
    CHARACTER*15 PROMPT
    prompt='>              '
    OPEN (14,FILE='ERRORS')
    CALL GOPKS (14,1024)
    CALL GOPWK (0,0,0)
    CALL GOPWK (1,0,1)
    CALL GOPWK (2,0,2)
    CALL GACWK (0)
    CALL GACWK (1)
    CALL GACWK (2)
```

```
      CALL GQDSP (1,ERR,DCUNIT,XDCMX,YDCMX,XRAS,YRAS)
      IF (XDCMX.GT.YDCMX) THEN
                  SCALE=XDCMX
      ELSE
                  SCALE=YDCMX
      END IF
      XNDC=XDCMX/SCALE
      YNDC=YDCMX/SCALE
      CALL GSWKWN (1,0.0,XNDC,0.0,YNDC)
      CALL GSWKVP (1,0.0,XDCMX,0.0,YDCMX)

C
C TRANSFORMATION #1
C
      CALL GSWN (1,XMN,XMX,YMN,YMX)
      CALL GSVP (1,0.0,0.73*XNDC,0.0,YNDC)
C
C TRANSFORMATION #2
C
      CALL GSWN (2,-1.0,31.0,-21.0,100.0)
      CALL GSVP (2,0.74*XNDC,XNDC,0.0,YNDC)
C TEST FONT
      CALL GSTXFP (-101,2)
C LOCATOR MODE
      CALL GSLCM(1,2,0,1)
C INITIALIZE STRING
      CALL
GINST(1,1,15,PROMPT,1,0.75*XDCMX,XDCMX,0.8347*YDCMX,0.9*
YDCMX
     & ,20,1,10,PROMPT)
C INITIALIZE PICK
      CALL GINPK(1,2,1,1,1,1,0.0,XDCMX,0.0,YDCMX,10,PROMPT)
      CALL GDAWK (2)
      RETURN
      END
C
C*************************************************************
*********
C
                  subroutine box (xmin,xmax,ymin,ymax,IC,IS)
                  real xmin, ymin, xmax, ymax,x(5), y(5)
                  INTEGER IC,IS
c
                  x(1) = xmin
                  y(1) = ymin
                  x(2) = xmax
                  y(2) = ymin
                  x(3) = xmax
                  y(3) = ymax
                  x(4) = xmin
                  y(4) = ymax
                  x(5) = xmin
                  y(5) = ymin
                  CALL GSFAIS(IS)
                  CALL GSFACI(IC)
                  CALL GFA (5, x, y)
                  return
                  end
C
C*************************************************************
***
C
                  SUBROUTINE FIND(XX,YY,NUM)
                  REAL XX,YY,DIST,TEST
                  INCLUDE 'NCOMMON.F'
C
C THIS SUBROUTINE FINDS THE CLOSEST NODE TO THE PICK
POINT
C
C SEND BACK NODE 0 IF OUT OF WINDOW
                  IF(XX.GT.XMX) THEN
                   NUM=0
                   RETURN
                  END IF

                  NUM=1
                  TEST=((XX-PN(1,4))**2.0+(YY-PN(1,5))**2.0)**0.50
                  DO 10 I=2,NON
                  DIST=((XX-PN(I,4))**2.0+(YY-PN(I,5))**2.0)**0.50
                  IF (DIST.LT.TEST.AND.INF(I).EQ.1) THEN
                   TEST=DIST
                   NUM=I
                  END IF
10    CONTINUE
                  RETURN
                  END
C
C*************************************************************
**************
C
                  SUBROUTINE FILESUB(TITLE)
                  CHARACTER*20 TITLE,BUF,DUM,PATH,STAT
                  CHARACTER*40 TOTAL
                  INCLUDE 'NCOMMON.F'
C
C THIS SUBROUTINE  i) OPENS & READS A DATA FILE
C                 ii) SAVES THE FILE (SAME OF NEW NAME)
C                iii) CLEARS THE DATA FOR A NEW FILE
C                 iv) CHECKS THE DATA FOR COMPLETENESS
C                  v) SETS THE PATH FOR FILE USAGE
C                 vi) QUITS THE MAIN PROGRAM
C
C     LOCAL VARIABLES
C
C     IPL...LENGTH OF PATH NAME
C     USED TO JOIN TOGETHER TO MAKE TOTAL
C     WHICH IS USED TO READ/WRITE FILES
C
C     TITLE...NAME OF MODEL/FILE
C     PATH....PATH TO DIRECTORY USED FOR
READING/WRITING
C     TOTAL....PATH & TITLE JOINED TOGETHER FOR OPEN A
FILE
C
C     BUF.....CHARACTER BUFFER THAT ALL INPUT IS READ
INTO
C     DUM.....DUMMY CHARACTER FOR COMPARING INPUT
Y/N?
C
C     DEFAULT PATH
                  IF(NOE.EQ.0) PATH='.   '
                  IPL=1
C     WRITING BUTTON NAMES
C
                  CALL GSTXCI(ORANGE)
                  CALL GTX( 2.0, 34.0, 'OPEN')
                  CALL GTX( 2.5, 24.0, 'NEW')
                  CALL GTX(12.0, 34.0, 'SAVE')
                  CALL GTX(11.5, 14.0, 'CHECK')
                  CALL GTX(10.0, 22.0, 'DIMENSIONS')
                  CALL GTX(10.0, 25.0, ' SELECT ')
                  CALL GTX(21.0, 34.0, 'SAVE AS')
                  CALL GTX(22.0,24.0, 'PATH')
                  CALL GSTXCI(14)
                  CALL GTX( 2.0, 14.0, 'DONE')
                  CALL GSTXCI(DKRED)
                  CALL GTX(22.0, 14.0, 'QUIT')
                  CALL GSTXCI(ORANGE)
C
C  MAIN PICK
C
5     CALL GRQPK(1,2,II,SEG,JJ)
                  CALL GSCHH(1.5)
                  CALL GSTXCI(ORANGE)
C QUIT
                  IF(SEG.EQ.NINE) THEN
                   IF(KSAV.EQ.1) THEN
                     CALL CLOSE
                     STOP
                   ELSE
                     CALL GSTXCI(RED)
                     CALL GTX(2.0,85.0,'QUIT WITHOUT SAVING?
Y/N')
                     CALL GRQST(1,1,I,J,DUM)
                     DUM=DUM(1:1)
                     BUF='Y'
                     IF(BUF.EQ.DUM) THEN
                              CALL CLOSE
                              STOP
                     END IF
                     BUF='y'
                     IF(BUF.EQ.DUM) THEN
                              CALL CLOSE
                              STOP
                     END IF
                              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                              CALL GSTXCI(ORANGE)
                   END IF

C NEW
                  ELSEIF(SEG.EQ.FOUR) THEN
                   NOG=0
                   NOE=0
                   NON=0
                   NAN=0
                   NAE=0
                   NBC=0
                   KSAV=1
                   NODIM=1
                   DO 81 I=1,700
```

```
                IEF(I)=0
                INF(I)=0
                INF(I)=0
                DO 81 J=1,6
                F(I,J)=0.0
81      ID(I,J)=0
                CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
                CALL GSCHH(3.0)
                CALL GSTXCI(ORANGE)
                TITLE='UNTITLED'
                CALL GTX(1.0,95.0,TITLE)
                CALL GSCHH(1.5)
                CALL DRAW(1,'  ',0)
C SAVE
              ELSEIF(SEG.EQ.TWO) THEN
                        STAT='OLD'
                        GOTO 250
C SAVE AS
              ELSEIF(SEG.EQ.THREE) THEN
                CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
                CALL GTX(2.0,85.0,'ENTER FILE NAME')
                CALL GRQST(1,1,I,J,TITLE)
                CALL GSCHH(3.0)
                CALL GTX(1.0,95.0,TITLE)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                STAT='NEW'
                CALL GSCHH(1.5)
                GOTO 250
C PATH
              ELSEIF(SEG.EQ.SIX) THEN
                CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
                CALL GTX(2.0,85.0,'ENTER FILE PATH')
                CALL GRQST(1,1,I,J,PATH)
                DO 700 I=1,15
                DUM=PATH(1:(17-I))
                BUF=PATH(1:(16-I))
700     IF (BUF.EQ.DUM) IPL=16-I
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
C CHECK
              ELSEIF(SEG.EQ.EIGHT) THEN
                CALL BOX(22.0,31.0,89.0,96.0,GREY,1)
                CALL CHECK()
C SELECT PROBLEM TYPE
              ELSEIF(SEG.EQ.FIVE) THEN
              IF(NOE.GT.0.OR.NON.GT.0) THEN
               CALL GSTXCI(RED)
               CALL GTX(2.,88., 'ELEMENTS/NODES EXIST!')
               CALL GTX(2.5,85.,'  NEW FILE FIRST    ')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               GOTO 5
               END IF
              CALL GSTXCI(ORANGE)
              CALL GTX(2.0,88.0,'1=SAVED FILE  2= 2D ')
              CALL GTX(2.0,85.0,'3= 3D  ')
              CALL GRQST(1,1,I,J,BUF)
              CALL BOX(-1.0,31.0,80.4,91.0,GREY,1)
              READ(BUF,*,ERR=999) I
              IF(I.GT.3.OR.I.LT.1) THEN
               CALL GSTXCI(RED)
               CALL GTX(2.,88.,' YOU MUST PICK 1-3')
               CALL GTX(2.5,85.,' ENTER TO CONTINUE')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,91.0,GREY,1)
               GOTO 5
               END IF
               NODIM=I
C DONE
              ELSEIF(SEG.EQ.SEVEN) THEN
              CALL GSTXCI(DKBLUE)
              CALL GTX( 2.0, 34.0, 'OPEN')
              CALL GTX( 2.5, 24.0, 'NEW')
              CALL GTX( 2.0, 14.0, 'DONE')
              CALL GTX(12.0, 34.0, 'SAVE')
              CALL GTX(11.5, 14.0, 'CHECK')
              CALL GTX(10.0, 22.0, 'DIMENSIONS')
              CALL GTX(10.0, 25.0, '  SELECT  ')
              CALL GTX(21.0, 34.0, 'SAVE AS')
              CALL GTX(22.0, 14.0, 'QUIT')
              CALL GTX(22.0,24.0, 'PATH')
1       RETURN
C
C OPEN
              ELSEIF(SEG.EQ.ONE) THEN
              CALL GSTXCI(WHITE)
              XMN=0.0
              YMN=0.0
              XMX=30.0
              YMX=22.0
              XDIS=XMX
              YDIS=YMX
              CALL GSWN (1,XMN,XMX,YMN,YMX)
              CALL GSELNT (1)
              CALL GSCHH ( YDIS/27.0 )
              OPEN(UNIT=8,FILE='TEMP.LST')
              X(1)=XMN+0.05*XDIS
78      CALL BOX(xmn,xmx,ymn,ymx,black,1)

              DO 77 I=1,15
              READ(8,*,ERR=999,END=79) BUF
              Y(1)=YMN+0.97*YDIS-I*0.055*YDIS
77      CALL GTX(X(1),Y(1),BUF)

              Y(1)=YMN+0.97*YDIS-I*0.055*YDIS
              CALL GTX(X(1),Y(1),'ENTER FOR MORE')
              CALL GRQST(1,1,I,J,DUM)
              GOTO 78

79      Y(1)=YMN+0.97*YDIS-(I+1)*0.055*YDIS
              CALL GTX(X(1),Y(1),'END OF DATA FILES ')
              CALL GSELNT (2)
               CALL GSTXCI(ORANGE)
               CALL GSCHH(1.5)
               CLOSE(UNIT=8)
               IJK=0
               CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
               CALL GTX(2.0,85.0,'ENTER FILE NAME')
               CALL GRQST(1,1,I,J,TITLE)
                CALL GSELNT (1)
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                CALL GSELNT (2)
               CALL BOX(-1.0,31.0,80.4,90.7,GREY,1)
               TOTAL=PATH(1:IPL)//\'//TITLE
               OPEN(UNIT=4,FILE=TOTAL,STATUS='OLD',ERR=799)

READ(4,*,ERR=990)NON,NAN,NOE,NAE,NODIM,NOG,NBC,I,J
C
C CONVERT 4D INTO AXIS-SYMMETRIC
C       IF(NODIM.EQ.4) THEN
C        NODIM=2
C        IJK=1
C       END IF
              DO 60 I=1,NAN
               READ(4,*,ERR=991) K,(PN(K,J),J=1,NODIM)
               INF(K)=1
               PN(K,4)=PN(K,1)
60      PN(K,5)=PN(K,2)

              DO 65 I=1,NOG
               NOGAC(I)=1
               READ(4,*,ERR=992) NOETY(I),NOEIG(I),J
              DO 65 J=1,NOEIG(I)
               READ(4,*,ERR=993) K,(
NEC(K,IJ),IJ=1,NUMNOD(NOETY(I) ) )
               IEF(K)=1
               NEC(K,9)=I
65      CONTINUE

              DO 70 I=1,NBC
               READ(4,*,ERR=994) IDL(I),( ID(I,J),J=1,6)
70      CONTINUE

              DO 75 I=1,NOG
               READ(4,*,ERR=995) NPRESG,J
75      READ(4,*,ERR=996) (PMAT(NPRESG,k),k=1,j)

C
C ELASTIC FOUNDATION #
C
              READ(4,*,ERR=997) ELFKE
              READ(4,*,ERR=998) I,J
              DO 80 I=1,NON
80      READ(4,*,ERR=999) J,F(I,1),F(I,2),F(I,3)
   @                ,F(I,4),F(I,5),F(I,6)
              CLOSE(UNIT=4)
C       IF (IJK.EQ.1) NODIM=4
              CALL GSTXCI(ORANGE)
              CALL GSCHH(3.0)
              CALL GTX(1.0,95.0,TITLE)
              CALL GSCHH(1.5)
              CALL GTX(22.0,91.0,'saved')
              CALL SEARCH()
              CALL DRAW(1,'  ',0)
              goto 1000

990     CALL GSELNT (2)
              CALL GSTXCI(RED)
              CALL GTX(2.,88., '* ERROR IN CONTROL DATA! *')
              CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
              CALL GRQST(1,1,K,J,BUF)
```

```
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

991   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN NODE POSN! *')
      CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

992   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN ELEMENTS HEADER!
*')
      CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

993   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN ELEMENT CONCT! *')
      CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

994   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN BC DATA! *')
      CALL GTX(2.5,85.,' ENTER TO CONTINUE ')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

995   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN MATERIAL HEADER!
*')
      CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

996   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN MATERIAL PROPS! *')
      CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

997   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN ELASTIC FND! *')
      CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

998   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN FORCE HEADER! *')
      CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)
      GOTO 1000

999   CALL GSELNT (2)
      CALL GSTXCI(RED)
      CALL GTX(2.,88., '* ERROR IN FORCE DATA! *')
      CALL GTX(2.5,85.,'  ENTER TO CONTINUE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
      TITLE='UNTITLED'
      CALL GSTXCI(ORANGE)
      CALL GSCHH(3.0)
      CALL GTX(1.0,95.0,TITLE)
      CALL GSCHH(1.5)


1000  CONTINUE
      END IF
      GOTO 5


C ERROR IN FILE/PATH
799   CALL GSTXCI(RED)
      CALL GTX(2.0,88.0,'ERROR IN PATH/FILE')
      CALL GTX(2.0,85.0,' RENTER PATH/FILE')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
      GOTO 5

C ERROR IN FILE/PATH WRITING
798   CALL GSTXCI(RED)
      CALL GTX(2.0,88.0,'  ERROR IN PATH OR')
      CALL GTX(2.0,85.0,'FILE ALREADY EXISTS!')
      CALL GRQST(1,1,K,J,BUF)
      CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
      GOTO 5


C
C SAVING
C
250   IF(NODIM.GT.1) THEN
      TOTAL=PATH(1:IPL)//\//TITLE
      OPEN(UNIT=2,FILE=TOTAL,STATUS=STAT,ERR=798)
      WRITE(2,91)
NON,NAN,NOE,NAE,NODIM,NOG,NBC,NOG,1
      ELSE
      GOTO 125
      END IF
      DO 10 I=1,NON
      IF(INF(I).EQ.1) WRITE(2,95) I,(PN(I,J),J=1,NODIM)
10    CONTINUE
      ijk=1
      DO 20 I=1,NOG
      WRITE(2,93) NOETY(I),NOEIG(I),I
      DO 20 J=1,NOE
      IF(NEC(J,9).EQ.I.AND.IEF(I).EQ.1) then

WRITE(2,92)ijk,(NEC(J,K),K=1,NUMNOD(NOETY(I)))
        ijk=ijk+1
        end if
20    CONTINUE

      DO 30 I=1,NBC
30    WRITE(2,96) IDL(I),(ID(I,J),J=1,6)

      DO 40 I=1,NOG
      WRITE(2,*)I,NUMMAT(NOETY(I)),'    GROUP# #OF
MAT PROPS'
```

```
40      WRITE(2,97)(PMAT(I,J),J=1,NUMMAT(NOETY(I)))
C
C WRITE ELASTIC FOUNDATION #
C
                WRITE(2,*) 0.0
                WRITE(2,94)NON,0
                DO 50 I=1,NON
50      WRITE(2,99)I,F(I,1),F(I,2),F(I,3),F(I,4),F(I,5),F(I,6)
                CLOSE(UNIT=2)
                CALL GSTXCI(ORANGE)
                CALL GSCHH(3.0)
                CALL GTX(1.0,95.0,TITLE)
                CALL GSCHH(1.5)
                CALL GTX(22.0,91.0,'saved')
                KSAV=1
                GOTO 5
C
C ERROR MESSAGE FOR NO TYPE SELECTED
C
125     CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'PROBLEM TYPE NOT DEFINED')
                CALL GTX(1.0,85.0,' SELECT PROBLEM TYPE  ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
    GOTO 5

C FORMATS
91      FORMAT(9I6,' #NODES<U&A>#ELEM #DIM #GROUP #BC
#MAT UNUSED#')
92      FORMAT(9I6,'   ELEM#   CONNECTIVITY')
93      FORMAT(3I7,'   ELEM TYPE#   # IN GROUP    GROUP#')
94      FORMAT(2I7,'    # OF FORCES   UNUSED #')
95      FORMAT(I7,3F9.3,'    NODE#  POSITIONS X Y Z')
96      FORMAT(7I7,'    NODE# DOFS 1=FIXED 0=FREE')
97      FORMAT(9F13.2,'      MATERIAL PROPERTIES')
99      FORMAT(I6,8F9.3,'  NODE# WITH FORCE    FORCES IN DOF
ORDER')
                END
C
C*************************************************************
*************
C
                SUBROUTINE DRAW(IKJ,BUF,ISORT)
C
C     THIS ROUTINE DRAWS THE GRAPHICS
C
C     IKJ.... = 0 FOR META DRAW, 1 FOR SCREEN
C     BUF.... IS TITLE FOR META DRAW
C     ISORT.. 1=SORT, 0=NO SORT
C     CF..... CORRECTION FACTOR FOR # PLACEMENT
(META/SCREEN)
C     FMAX....MAXIMUM FORCE
C     MMAX....MAXIMUM MOMENT
C     KSOW....FLAG SOLID OR WIREFRAME 0=WIRE,1=SOLID
C
                CHARACTER*20 BUF
                INTEGER IKJ,ISORT
                REAL CP,CS,SS,SP,FMAX,MMAX,CF
                INCLUDE NCOMMON.F
C
C GOING TO TRANSFORMATION #1
C
                CALL GSELNT (1)
                CP=COS(AP)
                SP=SIN(AP)
                CS=COS(AS)
                SS=SIN(AS)
                CALL GSLWSC(1.0)
C
C CODE FOR META DUMP OR NOT
C
               IF(IKJ.EQ.0) THEN
                 CF=0.0
                 XMN=XMN-1.5*XDIS
                 XMX=XMX+1.5*XDIS
                 XDIS=XMX-XMN
                 YMN=YMN-1.5*YDIS
                 YMX=YMX+1.5*YDIS
                 YDIS=YMX-YMN
                 CALL GSWN (1,XMN,XMX,YMN,YMX)
                 CALL GSTXCI(BLACK)
                 CALL GTX(XMN+0.37*XDIS,YMN+0.37*YDIS,BUF)
                 CALL GSPMCI(BLACK)
               ELSE
                 CF=0.12
                 CALL BOX(xmn,xmx,ymn,ymx,black,1)
                 CALL GSTXCI(WHITE)
                 CALL GSPMCI(YELLOW)
                 CALL GSMK(KOMT)
                 END IF
                 CALL GSCHH(YDIS/55.0)
C
C CALLING SORT
C
                IF(ISORT.EQ.1) CALL SORT()

C
C SET SOLID OR WIREFRAME
                CALL GSFAIS(KSOW)
C IF ELEMENTS ACTIVE...
   IF(KES.EQ.1) THEN
C LOOP THRU ELEMENTS
                DO 5 JJ=1,NOE
                J=NOOSE(JJ)
C CHECK IF ELEMENT IS ACTIVE
                IF(IEF(J).NE.0) THEN
C CHECK IF ELEMENTS GROUP IS ACTIVE
                IF(NOGAC(NEC(J,9)).EQ.1) THEN
C GOTO APPROPRIATE PLOTTING CODE
                 GOTO
(10,10,10,10,10,20,30,40,10,30,30,30),NOETY(NEC(J,9))
C           1 2 3 4 5 6 7 8 9 10 11 12
C 3D SPRING, 2D TRUSS.BEAM.GRID, 3D TRUSS.FRAME
10      X(1)=PN(NEC(J,1),4)
                X(2)=PN(NEC(J,2),4)
                Y(1)=PN(NEC(J,1),5)
                Y(2)=PN(NEC(J,2),5)
                CALL GSPLCI( NEC(J,9) )
                CALL GPL(2,X,Y)
                IF(KEN.EQ.1) THEN
                  X(1)=(X(1)+X(2))/2.0
                  Y(1)=(Y(1)+Y(2))/2.0
                  X(1)=X(1)-CF*XDIS
                  WRITE(BUF,*) J
                  CALL GTX(X(1),Y(1),BUF)
                  END IF
                 GOTO 5
C
C #2 3D BEAM
20      X(1)=PN(NEC(J,1),4)
                X(2)=PN(NEC(J,3),4)
                X(3)=PN(NEC(J,4),4)
                X(4)=PN(NEC(J,6),4)
                Y(1)=PN(NEC(J,1),5)
                Y(2)=PN(NEC(J,3),5)
                Y(3)=PN(NEC(J,4),5)
                Y(4)=PN(NEC(J,6),5)
                CALL GSFACI( NEC(J,9) )
                CALL GFA(4,X,Y)
                IF(KSOW.EQ.0) GOTO 132
                CALL GSFACI(DKRED)
                CALL GSFAIS(0)
                CALL GFA(4,X,Y)
132     CALL GSFAIS(KSOW)
                IF(KEN.EQ.1) THEN
                  X(1)=(X(1)+X(2)+X(3)+X(4))/4.0
                  Y(1)=(Y(1)+Y(2)+Y(3)+Y(4))/4.0
                  X(1)=X(1)-CF*XDIS
                  WRITE(BUF,*) J
                  CALL GTX(X(1),Y(1),BUF)
                  END IF
                 GOTO 5
C
C #3 3D PLATE.GRID PLATE.2D STRESS_STRAIN
30      X(1)=PN(NEC(J,1),4)
                X(2)=PN(NEC(J,2),4)
                X(3)=PN(NEC(J,3),4)
                X(4)=PN(NEC(J,4),4)
                Y(1)=PN(NEC(J,1),5)
                Y(2)=PN(NEC(J,2),5)
                Y(3)=PN(NEC(J,3),5)
                Y(4)=PN(NEC(J,4),5)
                X(5)=X(1)
                Y(5)=Y(1)
                CALL GSFACI( NEC(J,9) )
                CALL GFA(5,X,Y)
                IF(KSOW.EQ.0) GOTO 133
                CALL GSFACI(DKRED)
                CALL GSFAIS(0)
                CALL GFA(5,X,Y)
133     CALL GSFAIS(KSOW)
                IF(KEN.EQ.1) THEN
                  X(1)=(X(1)+X(2)+X(3)+X(4))/4.0
                  X(1)=X(1)-CF*XDIS
                  Y(1)=(Y(1)+Y(2)+Y(3)+Y(4))/4.0
                  WRITE(BUF,*) J
                  CALL GTX(X(1),Y(1),BUF)
                  END IF
                 GOTO 5
C
```

```
C 3D BRICK
C
C VIEW POINT
40    XV=100000.0*COS(AS)*COS(AP)
              YV=100000.0*SIN(AS)
              ZV=100000.0*COS(AS)*SIN(AP)
C TOP BOTTOM PICK

SX=PN(NEC(J,1),1)+PN(NEC(J,2),1)+PN(NEC(J,3),1)+PN(NEC(J,4),1)/4
.

SY=PN(NEC(J,1),2)+PN(NEC(J,2),2)+PN(NEC(J,3),2)+PN(NEC(J,4),2)/4
.

SZ=PN(NEC(J,1),3)+PN(NEC(J,2),3)+PN(NEC(J,3),3)+PN(NEC(J,4),3)/4.
              EDTVPA= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(J,5),1)+PN(NEC(J,6),1)+PN(NEC(J,7),1)+PN(NEC(J,8),1)/4
.

SY=PN(NEC(J,5),2)+PN(NEC(J,6),2)+PN(NEC(J,7),2)+PN(NEC(J,8),2)/4
.

SZ=PN(NEC(J,5),3)+PN(NEC(J,6),3)+PN(NEC(J,7),3)+PN(NEC(J,8),3)/4.
              EDTVPB= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
    IF(EDTVPA.LT.EDTVPB) THEN
              X(1)=PN(NEC(J,1),4)
              Y(1)=PN(NEC(J,1),5)
              X(2)=PN(NEC(J,2),4)
              Y(2)=PN(NEC(J,2),5)
              X(3)=PN(NEC(J,3),4)
              Y(3)=PN(NEC(J,3),5)
              X(4)=PN(NEC(J,4),4)
              Y(4)=PN(NEC(J,4),5)
             ELSE
              X(1)=PN(NEC(J,5),4)
              Y(1)=PN(NEC(J,5),5)
              X(2)=PN(NEC(J,6),4)
              Y(2)=PN(NEC(J,6),5)
              X(3)=PN(NEC(J,7),4)
              Y(3)=PN(NEC(J,7),5)
              X(4)=PN(NEC(J,8),4)
              Y(4)=PN(NEC(J,8),5)
              END IF
             X(5)=X(1)
             Y(5)=Y(1)
             CALL GSFACI( NEC(J,9) )
             CALL GFA(5,X,Y)
             IF(KSOW.EQ.0) GOTO 143
             CALL GSFACI(DKRED)
             CALL GSFAIS(0)
             CALL GFA(5,X,Y)
143   CALL GSFAIS(KSOW)
C LEFT RIGHT PICK

SX=PN(NEC(J,1),1)+PN(NEC(J,2),1)+PN(NEC(J,6),1)+PN(NEC(J,5),1)/4
.

SY=PN(NEC(J,1),2)+PN(NEC(J,2),2)+PN(NEC(J,6),2)+PN(NEC(J,5),2)/4
.

SZ=PN(NEC(J,1),3)+PN(NEC(J,2),3)+PN(NEC(J,6),3)+PN(NEC(J,5),3)/4.
              EDTVPA= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(J,4),1)+PN(NEC(J,3),1)+PN(NEC(J,8),1)+PN(NEC(J,7),1)/4
.

SY=PN(NEC(J,4),2)+PN(NEC(J,3),2)+PN(NEC(J,8),2)+PN(NEC(J,7),2)/4
.

SZ=PN(NEC(J,4),3)+PN(NEC(J,3),3)+PN(NEC(J,8),3)+PN(NEC(J,7),3)/4.
              EDTVPB= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
    IF(EDTVPA.LT.EDTVPB) THEN
              X(1)=PN(NEC(J,1),4)
              Y(1)=PN(NEC(J,1),5)
              X(2)=PN(NEC(J,2),4)
              Y(2)=PN(NEC(J,2),5)
              X(3)=PN(NEC(J,6),4)
              Y(3)=PN(NEC(J,6),5)
              X(4)=PN(NEC(J,5),4)
              Y(4)=PN(NEC(J,5),5)
             ELSE
              X(1)=PN(NEC(J,3),4)
              Y(1)=PN(NEC(J,3),5)
              X(2)=PN(NEC(J,4),4)
              Y(2)=PN(NEC(J,4),5)
              X(3)=PN(NEC(J,8),4)
              Y(3)=PN(NEC(J,8),5)
              X(4)=PN(NEC(J,7),4)
              Y(4)=PN(NEC(J,7),5)
              END IF
             X(5)=X(1)
             Y(5)=Y(1)
             CALL GSFACI( NEC(J,9) )
             CALL GFA(5,X,Y)
             IF(KSOW.EQ.0) GOTO 144
             CALL GSFACI(DKRED)
             CALL GSFAIS(0)
             CALL GFA(5,X,Y)
144   CALL GSFAIS(KSOW)
C FRONT BACK PICK

SX=PN(NEC(J,1),1)+PN(NEC(J,4),1)+PN(NEC(J,8),1)+PN(NEC(J,5),1)/4
.

SY=PN(NEC(J,1),2)+PN(NEC(J,4),2)+PN(NEC(J,8),2)+PN(NEC(J,5),2)/4
.

SZ=PN(NEC(J,1),3)+PN(NEC(J,4),3)+PN(NEC(J,8),3)+PN(NEC(J,5),3)/4.
              EDTVPA= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(J,2),1)+PN(NEC(J,3),1)+PN(NEC(J,7),1)+PN(NEC(J,6),1)/4
.

SY=PN(NEC(J,2),2)+PN(NEC(J,3),2)+PN(NEC(J,7),2)+PN(NEC(J,6),2)/4
.

SZ=PN(NEC(J,2),3)+PN(NEC(J,3),3)+PN(NEC(J,7),3)+PN(NEC(J,6),3)/4.
              EDTVPB= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
    IF(EDTVPA.LT.EDTVPB) THEN
              X(1)=PN(NEC(J,1),4)
              Y(1)=PN(NEC(J,1),5)
              X(2)=PN(NEC(J,4),4)
              Y(2)=PN(NEC(J,4),5)
              X(3)=PN(NEC(J,8),4)
              Y(3)=PN(NEC(J,8),5)
              X(4)=PN(NEC(J,5),4)
              Y(4)=PN(NEC(J,5),5)
             ELSE
              X(1)=PN(NEC(J,2),4)
              Y(1)=PN(NEC(J,2),5)
              X(2)=PN(NEC(J,3),4)
              Y(2)=PN(NEC(J,3),5)
              X(3)=PN(NEC(J,7),4)
              Y(3)=PN(NEC(J,7),5)
              X(4)=PN(NEC(J,6),4)
              Y(4)=PN(NEC(J,6),5)
              END IF
             X(5)=X(1)
             Y(5)=Y(1)
             CALL GSFACI( NEC(J,9) )
             CALL GFA(5,X,Y)
             IF(KSOW.EQ.0) GOTO 145
             CALL GSFACI(DKRED)
             CALL GSFAIS(0)
             CALL GFA(5,X,Y)
145   CALL GSFAIS(KSOW)
C BRICK NUMBER
              IF(KEN.EQ.1) THEN
               X(1)=PN(NEC(J,1),4)+PN(NEC(J,2),4)/4.0
               Y(1)=PN(NEC(J,1),5)+PN(NEC(J,2),5)/4.0
               X(1)=X(1)-CF*XDIS
               WRITE(BUF,*) J
               CALL GTX(X(1),Y(1),BUF)
               END IF
C END OF BRICK
              GOTO 5
              END IF
             END IF
5     CONTINUE
   END IF
C
C AXIS X,R=DKRED,Y,Z=DKBLUE,Z=DKGREEN
              CALL GSLWSC(0.5)
              X(1)=-AX(1)-AY(1)-AZ(1)
              Y(1)=-AX(2)-AY(2)-AZ(2)
              X(2)=X(1)+AX(1)
              Y(2)=Y(1)+AX(2)
              CALL GSTXCI(DKRED)
              CALL GSPLCI(DKRED)
              CALL GPL(2,X,Y)
              IF(NODIM.LT.4) THEN
               CALL GTX(X(2),Y(2),'X')
               ELSE
```

```
                CALL GTX(X(2),Y(2),'R')
                END IF
                X(2)=X(1)+AY(1)
                Y(2)=Y(1)+AY(2)
                CALL GSTXCI(DKBLUE)
                CALL GSPLCI(DKBLUE)
                CALL GPL(2,X,Y)
                IF(NODIM.LT.4) THEN
                 CALL GTX(X(2),Y(2),'Y')
                 ELSE
                 CALL GTX(X(2),Y(2),'Z')
                END IF
                IF(NODIM.EQ.3) THEN
                X(2)=X(1)+AZ(1)
                Y(2)=Y(1)+AZ(2)
                CALL GSTXCI(DKGREEN)
                CALL GSPLCI(DKGREEN)
                CALL GPL(2,X,Y)
                CALL GTX(X(2),Y(2),'Z')
                CALL GSTXCI(WHITE)
                END IF
C
C NODAL LOOP
C
                IF(KNS.EQ.1.AND.IKJ.EQ.1) THEN
                  CALL GSTXCI(YELLOW)
                  CF=CF*1.1
                  DO 70 I=1,NON
                  IF(INF(I).EQ.0) GOTO 70
                  X(1)=PN(I,4)
                  Y(1)=PN(I,5)
                  CALL GPM(1,X,Y)
                  X(1)=X(1)-CF*XDIS
                  Y(1)=Y(1)+0.01*YDIS
                  IF(KNN.EQ.1) THEN
                   WRITE(BUF,*) I
                   CALL GTX(X(1),Y(1),BUF)
                   END IF
70     CONTINUE
                END IF
C
C BOUNDARY CONDITION LOOP
C
                IF(KBC.EQ.1) THEN
                CALL GSTXCI(PURPLE)
                DO 80 I=1,NBC
                 X(1)=PN(IDL(I),4)
                 Y(1)=PN(IDL(I),5)
                 IF(ID(I,1).EQ.1) CALL GTX(X(1)+0.0*XDIS ,Y(1),'DX')
                 IF(ID(I,2).EQ.1) CALL GTX(X(1)+0.03*XDIS,Y(1),'DY')
                 IF(ID(I,3).EQ.1) CALL GTX(X(1)+0.06*XDIS ,Y(1),'DZ')
                 IF(ID(I,4).EQ.1) CALL GTX(X(1)+0.0*XDIS ,Y(1)-
0.03*YDIS,'RX')
                 IF(ID(I,5).EQ.1) CALL GTX(X(1)+0.03*XDIS,Y(1)-
0.03*YDIS,'RY')
80      IF(ID(I,6).EQ.1) CALL GTX(X(1)+0.06*XDIS ,Y(1)-
0.03*YDIS,'RZ')
                END IF
C
C FORCES LOOP
C FORCES IN YELLOW , MOMENTS IN PURPLE
     IF(KFOR.EQ.1) THEN
                FMAX=0.0
                MMAX=0.0
                DO 89 I=1,NON
                DO 89 J=1,3
                IF(F(I,J).GT.FMAX) FMAX=F(I,J)
89      IF(F(I,J+3).GT.MMAX) MMAX=F(I,J+3)
                FMAX=FMAX*4.0/YDIS
                MMAX=MMAX*4.0/YDIS
                DO 90 I=1,NON
                IF(MMAX.EQ.0.0) GOTO 87
C MOMENTS
                CALL GSLWSC(2.0)
                CALL GSLN(3)
                CALL GSPLCI(PURPLE)
                X(1)=PN(I,4)
                Y(1)=PN(I,5)
                X(2)=X(1)+F(I,4)*SP/MMAX
                Y(2)=Y(1)-F(I,4)*SS*CP/MMAX
                CALL GPL(2,X,Y)
                X(2)=X(1)
                Y(2)=Y(1)+F(I,5)*CS/MMAX
                CALL GPL(2,X,Y)
                X(2)=X(1)-F(I,6)*CP/MMAX
                Y(2)=Y(1)-F(I,6)*SS*SP/MMAX
                CALL GPL(2,X,Y)
                CALL GSLN(1)
                CALL GSLWSC(1.0)

C FORCES
87     IF(FMAX.EQ.0.0) GOTO 90
                CALL GSPLCI(YELLOW)
                CALL GSLWSC(2.0)
                X(1)=PN(I,4)
                Y(1)=PN(I,5)
                X(2)=X(1)+F(I,1)*SP/FMAX
                Y(2)=Y(1)-F(I,1)*SS*CP/FMAX
                CALL GPL(2,X,Y)
                X(2)=X(1)
                Y(2)=Y(1)+F(I,2)*CS/FMAX
                CALL GPL(2,X,Y)
                X(2)=X(1)-F(I,3)*CP/FMAX
                Y(2)=Y(1)-F(I,3)*SS*SP/FMAX
                CALL GPL(2,X,Y)
90     CONTINUE
                END IF

C
C
C BOUNDARY
                IF(IKJ.EQ.1) CALL
BOX(xmn,xmx,ymn,ymx,DKBLUE,0)
                CALL GSLWSC(3.0)
                CALL GSPLCI(PURPLE)
C
C END OF ROUTINE
                CALL GSELNT (2)
                CALL GSCHH(1.50)
                RETURN
                END
C
C**********************************************************
**************
C
                SUBROUTINE SEARCH()
                INCLUDE 'NCOMMON.F'
C
C THIS SUBROUTINE SEARCHES FOR MAX DIMENSIONS AND
SETS SCREEN ACCORD.
C
                XMN=PN(1,4)
                XMX=PN(1,4)
                YMN=PN(1,5)
                YMX=PN(I,5)
C
C FIND MAX
C
                DO 10 I=2,NON
                 IF(PN(I,4).GT.XMX) XMX=PN(I,4)
                 IF(PN(I,4).LT.XMN) XMN=PN(I,4)
                 IF(PN(I,5).GT.YMX) YMX=PN(I,5)
                 IF(PN(I,5).LT.YMN) YMN=PN(I,5)
10     CONTINUE
C
C RATIO CHECK
C
                XDIS=XMX-XMN
                YDIS=YMX-YMN
                IF(YDIS.GT.XDIS) THEN
                 XDIS=YDIS
                 XMX=XMN+XDIS
                ELSE
                 YDIS=XDIS
                 YMX=YMN+YDIS
                END IF
                YMN=YMN-0.05*YDIS
                YMX=YMX+0.05*YDIS
                XMN=XMN-0.05*XDIS
                XMX=XMX+0.05*XDIS
                XDIS=XMX-XMN
                YDIS=YMX-YMN

C
C SCALE WINDOW
C
                CALL GSELNT (1)
                CALL GSWN (1,XMN,XMX,YMN,YMX)
                CALL GSELNT (2)
C
                RETURN
                END
C
C**********************************************************
**************
C
                SUBROUTINE RESET()
                INCLUDE 'NCOMMON.F
C
C THIS SUBROUTINE RESETS THE ANGLE AND 2D DRAWING
```

```
C
            DO 10 I=1,NON
             PN(I,4)=PN(I,1)
10      PN(I,5)=PN(I,2)
            AX(1)=2.0
            AX(2)=0.0
            AY(1)=0.0
            AY(2)=2.0
            AZ(1)=0.0
            AZ(2)=0.0
            AS=0.0
            AP=1.57079
            CALL DRAW(1,'  ',1)
            RETURN
            END
C
C*************************************************************
**************
C
            SUBROUTINE ROTATE(FP,FS)
            REAL FP,FS,CP,SP,CS,SS
            INCLUDE 'NCOMMON.F'
C
C THIS SUBROUTINE ROTATES THE 3D COORDS AND MAPS
C THEM ONTO THE 2D VIEWING SURFACE
C
            AP=AP+FP*0.523599
            AS=AS+FS*0.523599
            IF(NODIM.EQ.2.OR.NODIM.EQ.4) THEN
              AS=0.0
              AP=1.57079
              END IF
            CP=COS(AP)
            SP=SIN(AP)
            CS=COS(AS)
            SS=SIN(AS)
            DO 10 I=1,NON
             PN(I,4)=PN(I,1)*SP-PN(I,3)*CP
10      PN(I,5)=PN(I,2)*CS-PN(I,1)*SS*CP-PN(I,3)*SS*SP
            AX(1)=2.0*SP
            AX(2)=-2.0*SS*CP
            AY(1)=0.0
            AY(2)=2.0*CS
            AZ(1)=-2.0*CP
            AZ(2)=-2.0*SS*SP
            CALL DRAW(1,'  ',1)
            RETURN
            END
C
C
*************************************************************
*************
C
            SUBROUTINE ZOOMIN()
            INCLUDE 'NCOMMON.F'
            CHARACTER*15 BUF
C
C    THIS SUBROUTINE ZOOMS IN ON THE DRAWING
C
            CALL GSTXCI(WHITE)
            BUF='          '
            CALL GTX(2.0,88.0,BUF)
            CALL GTX(2.0,85.0,BUF)
            BUF=' PICK CORNERS '
            CALL GTX(7.0,88.0,BUF)
            BUF=' OF ZOOM BOX '
            CALL GTX(7.0,85.0,BUF)
            CALL GRQLC(1,2,I,NT,X(1),Y(1))
            X(1)=XMN+X(1)*XDIS/0.73
            Y(1)=YMN+Y(1)*YDIS/0.74874
            CALL GRQLC(1,2,I,NT,X(2),Y(2))
            X(2)=XMN+X(2)*XDIS/0.73
            Y(2)=YMN+Y(2)*YDIS/0.74874
            YMN=MIN0(Y(1),Y(2))
            XMN=MIN0(X(1),X(2))
            YMX=MAX0(Y(1),Y(2))
            XMX=MAX0(X(1),X(2))
            XDIS=XMX-XMN
            YDIS=YMX-YMN
            IF(YDIS.GT.XDIS) THEN
             XDIS=YDIS
             XMX=XMN+XDIS
            ELSE
             YDIS=XDIS
             YMX=YMN+YDIS
            END IF
           CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
           CALL GSWN (1,XMN,XMX,YMN,YMX)
           CALL DRAW (1,'  ',0)
            RETURN
            END
C
C
*************************************************************
*************
C
            SUBROUTINE ZOOMOUT()
            INCLUDE 'NCOMMON.F'
C
C    THIS SUBROUTINE ZOOMS OUT TO A FACTOR OF 2
C
            XMN=XMN-0.5*XDIS
            XMX=XMX+0.5*XDIS
            YMN=YMN-0.5*YDIS
            YMX=YMX+0.5*YDIS
            YDIS=YMX-YMN
            XDIS=XMX-XMN
            CALL GSWN (1,XMN,XMX,YMN,YMX)
            CALL DRAW (1,'  ',0)
            RETURN
            END
C
C*************************************************************
**************
C
            SUBROUTINE CENTER()
            INCLUDE 'NCOMMON.F'
            CHARACTER*15 BUF
C
            CALL GSTXCI(WHITE)
            BUF='123456789012345'
            BUF='          '
            BUF='PICK NEW CENTER'
            CALL GTX(7.0,88.0,BUF)
            BUF='OF VIEWING AREA'
            CALL GTX(7.0,85.0,BUF)
            CALL GRQLC(1,2,I,NT,X(1),Y(1))
            X(1)=XMN+X(1)*XDIS/0.73
            Y(1)=YMN+Y(1)*YDIS/0.74874
            YMN=Y(1)-(YDIS/2)
            YMX=Y(1)+(YDIS/2)
            XMN=X(1)-(XDIS/2)
            XMX=X(1)+(XDIS/2)
            CALL GSWN (1,XMN,XMX,YMN,YMX)
            CALL DRAW(1,'  ',0)
            CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
            RETURN
            END
C
C*************************************************************
**************
C
            SUBROUTINE NODESUB()
            CHARACTER*20 BUF,DUM
            REAL HOLD
            INCLUDE 'NCOMMON.F'
            KSAV=0
            CALL BOX(22.0,31.0,89.0,96.0,GREY,1)

C
C THIS SUBROUTINE  i) APPLIES/DELETES FORCES
C          ii) APPLIES/DELETES BOUNDARY CONDITIONS
C         iii) CREATES NODES
C         iv) DELETES NODES
C          v) MODIFIES NODES
C         vi) DUPLICATES NODES
C    WRITING BUTTON NAMES
C
            CALL GSTXCI(YELLOW)
            CALL GSCHH(1.5)
C 1
            CALL GTX( 2.0,36.0,'  ')
            CALL GTX( 2.0, 36.0, 'APPLY')
            CALL GTX( 2.0, 33.0, 'FORCE')
C 2
            CALL GTX(12.0, 36.0, 'APPLY')
            CALL GTX(12.0, 33.0, ' B.C.')
C 3
            CALL GTX(21.0, 36.0, 'CREATE')
            CALL GTX(21.0, 33.0, ' NODE ')
C 6
            CALL GTX(21.0, 26.0, 'DELETE')
            CALL GTX(21.0, 23.0, ' NODE ')
C 9
            CALL GTX(21.0, 16.0, ' MODIFY')
            CALL GTX(21.0, 13.0, 'POSITION')
C 7
            CALL GSTXCI(14)
```

```
                CALL GTX( 2.0, 14.0, 'DONE')
                CALL GSTXCI(YELLOW)
C 4
                CALL GTX( 2.0, 26.0, 'DELETE')
                CALL GTX( 2.0, 23.0, 'FORCE')
C 5
                CALL GTX(12.0, 26.0, 'DELETE')
                CALL GTX(12.0, 23.0, ' B.C.')
C 8
                CALL GTX(11.0, 16.0, 'DUPLICATE')
                CALL GTX(11.0, 13.0, ' NODES ')


C
C  MAIN PICK
C
5     CALL GRQPK(1,2,II,SEG,JJ)
                CALL GSTXCI(YELLOW)
                CALL GSCHH(1.5)

C
C DONE
                IF(SEG.EQ.SEVEN) THEN
                CALL GSTXCI(DKBLUE)
                CALL GSCHH(1.5)
                CALL GTX( 2.0,36.0,'   ')
                CALL GTX( 2.0, 36.0, 'APPLY')
                CALL GTX( 2.0, 33.0, 'FORCE')
                CALL GTX(12.0, 36.0, 'APPLY')
                CALL GTX(12.0, 33.0, ' B.C.')
                CALL GTX(21.0, 36.0, 'CREATE')
                CALL GTX(21.0, 33.0, ' NODE ')
                CALL GTX(21.0, 26.0, 'DELETE')
                CALL GTX(21.0, 23.0, ' NODE ')
                CALL GTX(21.0, 16.0, ' MODIFY')
                CALL GTX(21.0, 13.0, 'POSITION')
                CALL GTX( 2.0, 14.0, 'DONE')
                CALL GTX( 2.0, 26.0, 'DELETE')
                CALL GTX( 2.0, 23.0, 'FORCE')
                CALL GTX(12.0, 26.0, 'DELETE')
                CALL GTX(12.0, 23.0, ' B.C.')
                CALL GTX(11.0, 16.0, 'DUPLICATE')
                CALL GTX(11.0, 13.0, ' NODES ')
                RETURN
                ELSEIF(SEG.EQ.BMOUSE) THEN
                        MOKB=1
                ELSEIF(SEG.EQ.BKEYB) THEN
                        MOKB=0

C
C CREATE NODE
                ELSEIF(SEG.EQ.THREE) THEN

15    NPRESN=NON+1
C
C X/Y/Z COORDS
                IF(NODIM.EQ.1) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'PROBLEM TYPE NOT DEFINED')
                CALL GTX(1.0,85.0,'  SELECT PROBLEM TYPE  ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 100
C OUT OF SPACE
                ELSEIF(NPRESN.EQ.701) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'700 NODES ALREADY EXIST')
                CALL GTX(1.0,85.0,'CONSULT PROGRAMMER
DOUG')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 100

                ELSEIF(NODIM.EQ.2) THEN
                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER X-COORDINATE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) PN(NPRESN,1)

                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER Y-COORDINATE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) PN(NPRESN,2)
                PN(NPRESN,3)=0.0

                ELSEIF(NODIM.EQ.3) THEN
                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER X-COORDINATE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) PN(NPRESN,1)

                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER Y-COORDINATE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) PN(NPRESN,2)

                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER Z-COORDINATE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) PN(NPRESN,3)

                ELSEIF(NODIM.EQ.4) THEN
                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER R-COORDINATE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) PN(NPRESN,1)

                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER Z-COORDINATE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) PN(NPRESN,2)
                PN(NPRESN,3)=0
                END IF
                PN(NPRESN,4)=PN(NPRESN,1)
                PN(NPRESN,5)=PN(NPRESN,2)
                NON=NON+1
                NAN=NAN+1
                INF(NPRESN)=1
                CALL ROTATE(0.0,0.0)

20    CALL GSTXCI(YELLOW)
                CALL GTX(2.0,85.0,'CREATE ANOTHER? Y/N')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                DUM='Y'
                BUF=BUF(1:1)
                IF (DUM.EQ.BUF) GOTO 15
                DUM='y'
                IF (DUM.EQ.BUF) GOTO 15
                GOTO 100

99    CALL GSTXCI(RED)
                CALL GTX(2.0,88.0,' ERROR IN INPUT')
                CALL GTX(2.0,85.0,'ENTER TO CONTINUE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 5
100   CONTINUE

C
C MODIFY NODE
                ELSEIF(SEG.EQ.NINE) THEN
                IF(MOKB.EQ.1) THEN
                CALL GTX(7.0,88.0,' USE MOUSE TO')
                CALL GTX(7.0,85.0,' SELECT NODE ')
                CALL GRQLC(1,2,I,NT,X(1),Y(1))
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                XX=XMN+X(1)*XDIS/0.73
                YY=YMN+Y(1)*YDIS/0.74874
                CALL FIND(XX,YY,NPRESN)
                ELSE
                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER NODE TO MOVE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,900,ERR=99) NPRESN
                END IF
                IF(INF(NPRESN).EQ.0) GOTO 95
                CALL GSMK(4)
                CALL GSPMCI(WHITE)
                CALL GSELNT (1)
                CALL GPM(1,PN(NPRESN,4),PN(NPRESN,5))
                CALL GSELNT (2)
C
C X/Y/Z COORDS
                IF(NODIM.EQ.1) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'PROBLEM TYPE NOT DEFINED')
                CALL GTX(1.0,85.0,'  SELECT PROBLEM TYPE  ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 100
```

```
            ELSEIF(NODIM.EQ.2) THEN
             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER X-COORDINATE')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(BUF,*,ERR=99) R1

             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER Y-COORDINATE')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(BUF,*,ERR=99) R2
             PN(NPRESN,3)=0.0

            ELSEIF(NODIM.EQ.3) THEN
             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER X-COORDINATE')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(BUF,*,ERR=99) R1

             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER Y-COORDINATE')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(BUF,*,ERR=99) R2

             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER Z-COORDINATE')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(BUF,*,ERR=99) R3

            ELSEIF(NODIM.EQ.4) THEN
             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER R-COORDINATE')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(BUF,*,ERR=99) R1

             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER Z-COORDINATE')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(BUF,*,ERR=99) R2
             R3=0
            END IF
            PN(NPRESN,1)=R1
            PN(NPRESN,2)=R2
            PN(NPRESN,3)=R3
            CALL ROTATE(0.0,0.0)

            GOTO 101

95   CALL GSTXCI(RED)
            CALL GTX(2.0,88.0,'NODE DOES NOT EXIST!')
            CALL GTX(2.0,85.0,' ENTER TO CONTINUE ')
            CALL GRQST(1,1,K,J,BUF)
            CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
101  CONTINUE
C
C DELETE NODE
            ELSEIF(SEG.EQ.SIX) THEN
234   IF(MOKB.EQ.1) THEN
             CALL GTX(7.0,88.0,' USE MOUSE TO')
             CALL GTX(7.0,85.0,' SELECT NODE ')
             CALL GRQLC(1,2,I,NT,X(1),Y(1))
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             XX=XMN+X(1)*XDIS/0.73
             YY=YMN+Y(1)*YDIS/0.74874
             CALL FIND(XX,YY,NPRESN)
            ELSE
             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER NODE TO DELETE')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
C
C CODE TO DELETE ALL FREE NODES
C
           IF(BUF(1:1).EQ.'A'.OR.BUF(1:1).EQ.'a') THEN
             CALL GSTXCI(ORANGE)
             CALL GTX(1.0,85.0,'DELETING FREE NODES')

             DO 34 KL=1,NON
             DO 33 J=1,NOE
             I=NEC(J,9)
             DO 33 K=1,NUMNOD(NOETY(I))
33    IF(NEC(J,K).EQ.KL.AND.IEF(J).EQ.1) GOTO 34
             INF(KL)=0
             CALL GSMK(KOMT)
             CALL GSPMCI(BLACK)
             CALL GSELNT (1)
             CALL GPM(1,PN(KL,4),PN(KL,5))
             CALL GSELNT (2)
34    CONTINUE

             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             GOTO 102
            END IF

            READ(BUF,900,ERR=99) NPRESN
            END IF

            DO 30 J=1,NOE
            I=NEC(J,9)
            DO 30 K=1,NUMNOD(NOETY(I))
            IF(NEC(J,K).EQ.NPRESN.AND.IEF(J).EQ.1) GOTO 97
30   CONTINUE
            DO 32 I=1,NBC
            IF(IDL(I).EQ.NPRESN) GOTO 332
32   CONTINUE
            IF(INF(NPRESN).EQ.0) GOTO 96
            INF(NPRESN)=0
            CALL GSMK(KOMT)
            CALL GSPMCI(BLACK)
            CALL GSELNT (1)
            CALL GPM(1,PN(NPRESN,4),PN(NPRESN,5))
            CALL GSELNT (2)
            IF(MOKB.EQ.1) GOTO 234
            GOTO 102
97   CALL GSTXCI(RED)
            CALL GTX(2.0,88.0,' NODE IS IN ELEMENT ')
            CALL GTX(2.0,85.0,'DELETE ELEMENT FIRST')
            CALL GRQST(1,1,K,J,BUF)
            CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
            GOTO 102
332  CALL GSTXCI(RED)
            CALL GTX(2.0,88.0,'NODE IS RESTRAINED')
            CALL GTX(2.0,85.0,'DELETE B.C. FIRST')
            CALL GRQST(1,1,K,J,BUF)
            CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
            GOTO 102
96   CALL GSTXCI(RED)
            CALL GTX(2.0,88.0,'NODE DOES NOT EXIST!')
            CALL GTX(2.0,85.0,' ENTER TO CONTINUE ')
            CALL GRQST(1,1,K,J,BUF)
            CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
102  CONTINUE
C
C DELETE B.C.
            ELSEIF(SEG.EQ.FIVE) THEN
            IF(MOKB.EQ.1) THEN
             CALL GTX(7.0,88.0,' USE MOUSE TO')
             CALL GTX(7.0,85.0,' SELECT NODE ')
             CALL GRQLC(1,2,I,NT,X(1),Y(1))
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             XX=XMN+X(1)*XDIS/0.73
             YY=YMN+Y(1)*YDIS/0.74874
             CALL FIND(XX,YY,NPRESN)
            ELSE
             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER NODE #')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(BUF,900,ERR=99) NPRESN
            END IF
            IF(NPRESN.EQ.0.OR.NPRESN.GT.NON) GOTO 122
            CALL GSMK(4)
            CALL GSPMCI(WHITE)
            CALL GSELNT (1)
            CALL GPM(1,PN(NPRESN,4),PN(NPRESN,5))
            CALL GSELNT (2)
            DO 13 I=1,NBC
13   IF(NPRESN.EQ.IDL(I)) GOTO 14
             CALL GSTXCI(RED)
             CALL GTX(1.0,88.0,'THIS NODE HAS NO B.C.')
             CALL GTX(1.0,85.0,'  ENTER TO CONTINUE ')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             GOTO 103
14   CALL GTX(1.0,85.0,'ENTER DOF TO FREE')
            CALL GRQST(1,1,K,J,BUF)
            CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
            BUF=BUF(1:2)
            NPDOF=0
            IF(BUF.EQ.'X '.OR.BUF.EQ.'x ') THEN
             NPDOF=1
            ELSEIF(BUF.EQ.'Y '.OR.BUF.EQ.'y ') THEN
             NPDOF=2
            ELSEIF(BUF.EQ.'Z '.OR.BUF.EQ.'z ') THEN
```

```
                NPDOF=3
               ELSEIF(BUF.EQ.'RX'.OR.BUF.EQ.'rx') THEN
                NPDOF=4
               ELSEIF(BUF.EQ.'RY'.OR.BUF.EQ.'ry') THEN
                NPDOF=5
               ELSEIF(BUF.EQ.'RZ'.OR.BUF.EQ.'rz') THEN
                NPDOF=6
               ELSEIF(BUF.EQ.'A'.OR.BUF.EQ.'a') THEN
                NPDOF=7
               END IF
               IF(NPDOF.EQ.0) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(1.0,88.0,'DOF# DOES NOT EXIST')
                 CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               ELSE
                IF(NPDOF.LT.7) THEN
                 ID(I,NPDOF)=0
                 DO 130 J=1,6
130     IF(ID(I,J).EQ.1) GOTO 135
                 DO 140 J=I,NBC-1
                 IDL(J)=IDL(J+1)
                 DO 140 K=1,6
                 ID(J,K)=ID(J+1,K)
140     ID(J+1,K)=0
                 NBC=NBC-1
                ELSE
                 ID(I,1)=0
                 ID(I,2)=0
                 ID(I,3)=0
                 ID(I,4)=0
                 ID(I,5)=0
                 ID(I,6)=0
                 DO 131 J=1,6
131     IF(ID(I,J).EQ.1) GOTO 135
                 DO 141 J=I,NBC-1
                 IDL(J)=IDL(J+1)
                 DO 141 K=1,6
                 ID(J,K)=ID(J+1,K)
141     ID(J+1,K)=0
                 NBC=NBC-1
                END IF
135   CONTINUE
               END IF
122   CALL GSTXCI(RED)
              CALL GTX(2.0,88.0,'NODE DOES NOT EXIST!')
              CALL GTX(2.0,85.0,' ENTER TO CONTINUE ')
              CALL GRQST(1,1,K,J,BUF)
              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)

103   CONTINUE
C
C CREATE B.C.
             ELSEIF(SEG.EQ.TWO) THEN
             IF(MOKB.EQ.1) THEN
              CALL GTX(7.0,88.0,' USE MOUSE TO')
              CALL GTX(7.0,85.0,' SELECT NODE ')
              CALL GRQLC(1,2,I,NT,X(1),Y(1))
              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
              XX=XMN+X(1)*XDIS/0.73
              YY=YMN+Y(1)*YDIS/0.74874
              CALL FIND(XX,YY,NPRESN)
             ELSE
              CALL GSTXCI(YELLOW)
              CALL GTX(1.0,85.0,'ENTER NODE #')
              CALL GRQST(1,1,K,J,BUF)
              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
              READ(BUF,900,ERR=99) NPRESN
             END IF
              IF(NPRESN.GT.NON.OR.NPRESN.EQ.0) THEN
               CALL GSTXCI(RED)
               CALL GTX(1.0,88.0,'NODE DOES NOT EXIST')
               CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               GOTO 104
               END IF
             CALL GSMK(4)
             CALL GSPMCI(WHITE)
             CALL GSELNT (1)
             CALL GPM(1,PN(NPRESN,4),PN(NPRESN,5))
             CALL GSELNT (2)
             CALL GSTXCI(YELLOW)
             CALL GTX(1.0,85.0,'ENTER DOF TO FIX')
             CALL GRQST(1,1,K,J,BUF)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             BUF=BUF(1:2)
             NPDOF=0
             IF(BUF.EQ.'X '.OR.BUF.EQ.'x ') THEN
                NPDOF=1
              ELSEIF(BUF.EQ.'Y '.OR.BUF.EQ.'y ') THEN
               NPDOF=2
              ELSEIF(BUF.EQ.'Z '.OR.BUF.EQ.'z ') THEN
               NPDOF=3
              ELSEIF(BUF.EQ.'RX'.OR.BUF.EQ.'rx') THEN
               NPDOF=4
              ELSEIF(BUF.EQ.'RY'.OR.BUF.EQ.'ry') THEN
               NPDOF=5
              ELSEIF(BUF.EQ.'RZ'.OR.BUF.EQ.'rz') THEN
               NPDOF=6
              ELSEIF(BUF.EQ.'A'.OR.BUF.EQ.'a') THEN
               NPDOF=7
              END IF
              IF(NPDOF.EQ.0) THEN
               CALL GSTXCI(RED)
               CALL GTX(1.0,88.0,'DOF# DOES NOT EXIST')
               CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               GOTO 104
              END IF

              DO 66 I=1,NBC
66    IF(NPRESN.EQ.IDL(I)) GOTO 67
               NBC=NBC+1
               IDL(NBC)=NPRESN
               I=NBC
67    IF(NPDOF.LT.7) THEN
               ID(I,NPDOF)=1
              ELSE
               ID(I,1)=1
               ID(I,2)=1
               ID(I,3)=1
               ID(I,4)=1
               ID(I,5)=1
               ID(I,6)=1
              END IF
104   CONTINUE

C
C CREATE FORCES
              ELSEIF(SEG.EQ.ONE) THEN
              IF(MOKB.EQ.1) THEN
               CALL GTX(7.0,88.0,' USE MOUSE TO')
               CALL GTX(7.0,85.0,' SELECT NODE ')
               CALL GRQLC(1,2,I,NT,X(1),Y(1))
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               XX=XMN+X(1)*XDIS/0.73
               YY=YMN+Y(1)*YDIS/0.74874
               CALL FIND(XX,YY,NPRESN)
              ELSE
               CALL GSTXCI(YELLOW)
               CALL GTX(1.0,85.0,'ENTER NODE #')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) NPRESN
              END IF
               IF(NPRESN.GT.NON.or.NPRESN.EQ.0) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'NODE DOES NOT EXIST')
                CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 105
                END IF
              CALL GSMK(4)
              CALL GSPMCI(WHITE)
              CALL GSELNT (1)
              CALL GPM(1,PN(NPRESN,4),PN(NPRESN,5))
              CALL GSELNT (2)
              CALL GSTXCI(YELLOW)
              CALL GTX(1.0,85.0,'ENTER DOF TO LOAD')
              CALL GRQST(1,1,K,J,BUF)
              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
              BUF=BUF(1:2)
              NPDOF=0
              IF(BUF.EQ.'X '.OR.BUF.EQ.'x ') THEN
               NPDOF=1
              ELSEIF(BUF.EQ.'Y '.OR.BUF.EQ.'y ') THEN
               NPDOF=2
              ELSEIF(BUF.EQ.'Z '.OR.BUF.EQ.'z ') THEN
               NPDOF=3
              ELSEIF(BUF.EQ.'RX'.OR.BUF.EQ.'rx') THEN
               NPDOF=4
              ELSEIF(BUF.EQ.'RY'.OR.BUF.EQ.'ry') THEN
               NPDOF=5
              ELSEIF(BUF.EQ.'RZ'.OR.BUF.EQ.'rz') THEN
               NPDOF=6
              END IF
```

```
                IF(NPDOF.EQ.0) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(1.0,88.0,'DOF# DOES NOT EXIST')
                 CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                 GOTO 105
                END IF

 77    CALL GSTXCI(YELLOW)
               CALL GTX(1.0,85.0,'ENTER FORCE VALUE')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,*,ERR=99) HOLD
               F(NPRESN,NPDOF)=HOLD

 105   CONTINUE
 C
 C DELETE FORCE
               ELSEIF(SEG.EQ.FOUR) THEN
               IF(MOKB.EQ.1) THEN
                CALL GTX(7.0,88.0,' USE MOUSE TO')
                CALL GTX(7.0,85.0,' SELECT NODE ')
                CALL GRQLC(1,2,I,NT,X(1),Y(1))
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                XX=XMN+X(1)*XDIS/0.73
                YY=YMN+Y(1)*YDIS/0.74874
                CALL FIND(XX,YY,NPRESN)
               ELSE
                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER NODE #')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,900,ERR=99) NPRESN
               END IF
                IF(NPRESN.GT.NON.or.NPRESN.EQ.0) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(1.0,88.0,'NODE DOES NOT EXIST')
                 CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                 GOTO 123
                 END IF
               CALL GSMK(4)
               CALL GSPMCI(WHITE)
               CALL GSELNT (1)
               CALL GPM(1,PN(NPRESN,4),PN(NPRESN,5))
               CALL GSELNT (2)
 114   CALL GSTXCI(YELLOW)
               CALL GTX(1.0,85.0,'ENTER DOF TO UNLOAD')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               BUF=BUF(1:2)
               NPDOF=0
               IF(BUF.EQ.'X '.OR.BUF.EQ.'x ') THEN
                NPDOF=1
               ELSEIF(BUF.EQ.'Y '.OR.BUF.EQ.'y ') THEN
                NPDOF=2
               ELSEIF(BUF.EQ.'Z '.OR.BUF.EQ.'z ') THEN
                NPDOF=3
               ELSEIF(BUF.EQ.'RX'.OR.BUF.EQ.'rx') THEN
                NPDOF=4
               ELSEIF(BUF.EQ.'RY'.OR.BUF.EQ.'ry') THEN
                NPDOF=5
               ELSEIF(BUF.EQ.'RZ'.OR.BUF.EQ.'rz') THEN
                NPDOF=6
               ELSEIF(BUF.EQ.'A'.OR.BUF.EQ.'a') THEN
                NPDOF=7

               END IF

               IF(NPDOF.EQ.0) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'DOF# DOES NOT EXIST')
                CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               ELSEIF(NPDOF.EQ.7) THEN
                F(NPRESN,1)=0.0
                F(NPRESN,2)=0.0
                F(NPRESN,3)=0.0
                F(NPRESN,4)=0.0
                F(NPRESN,5)=0.0
                F(NPRESN,6)=0.0
               ELSEIF(NPDOF.LT.7) THEN
                F(NPRESN,NPDOF)=0.0
               END IF
 123   CONTINUE
 C
 C DUPLICATE NODE
               ELSEIF(SEG.EQ.EIGHT) THEN
 C
 C START AT NON+1 TO NON+1+NDUP
               R1=0.
               R2=0.
               R3=0.
 C
 C ASK FOR NODES TO DUPLICATE
 C
               CALL GSTXCI(YELLOW)
               CALL GTX(1.0,85.0,'ENTER START NODE')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) II
               IF(INF(II).NE.1) GOTO 270
               II=II-1
               CALL GSTXCI(YELLOW)
               CALL GTX(1.0,85.0,'ENTER END NODE')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) JJ
               IF(INF(JJ).NE.1) GOTO 270
               NDUP=JJ-II
               CALL GSTXCI(YELLOW)
               CALL GTX(1.0,85.0,'ENTER INCREMENT')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) KK
               IF(KK.GT.NDUP.OR.KK.EQ.0) GOTO 275
 C ASK FOR SPACING
 C X/Y/Z COORDS
               IF(NODIM.EQ.1) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'PROBLEM TYPE NOT DEFINED')
                CALL GTX(1.0,85.0,'  SELECT PROBLEM TYPE  ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 100

               ELSEIF(NODIM.EQ.2) THEN
                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER X SPACING')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) R1

                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER Y SPACING')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) R2
                PN(NPRESN,3)=0.0

               ELSEIF(NODIM.EQ.3) THEN
                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER X SPACING')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) R1

                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER Y SPACING')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) R2

                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER Z SPACING')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) R3

               ELSEIF(NODIM.EQ.4) THEN
                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER R SPACING')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) R1

                CALL GSTXCI(YELLOW)
                CALL GTX(1.0,85.0,'ENTER Z SPACING')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,*,ERR=99) R2
                PN(NPRESN,3)=0
               END IF
               DO 260 I=1,NDUP,KK
               J=NON+I
               INF(J)=1
               PN(J,1)=PN(II+I,1)+R1
```

```
                PN(J,2)=PN(II+I,2)+R2
260     PN(J,3)=PN(II+I,3)+R3
                NON=NON+NDUP
                NAN=NON
                CALL ROTATE(0.0,0.0)
                GOTO 280
C IMPOSSIBLE NODE NUMBER
270     CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'NODE DOES NOT EXIST!')
                CALL GTX(1.0,85.0,' ENTER TO CONTINUE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 280
275     CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'IMPOSSIBLE INCREMENT')
                CALL GTX(1.0,85.0,' ENTER TO CONTINUE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
280     CONTINUE
                END IF
                GOTO 5
900     FORMAT(I6)
905     FORMAT(F8.2)
                END
C
C*************************************************************
**************
C
                SUBROUTINE LIST()
                CHARACTER*90 BUF,NAME
                INCLUDE 'NCOMMON.F'
C
C THIS SUBROUTINE LISTS THE
ELEM,NODE,BC,FORCES,GROUPS,MATERIALS
C
                CALL GSTXCI(WHITE)
C 1
                CALL GTX( 2.0, 34.0, 'NODES')
C 2
                CALL GTX(11.0, 34.0, 'ELEMENTS')
C 3
                CALL GTX(21.0, 34.0, ' B.C.')
C 4
                CALL GTX(2.0, 24.0, 'FORCES')
C 5
                CALL GTX(11.0, 24.0, 'GROUPS')
C 6
                CALL GTX(21.0,24.0, 'MATERIALS')
C 7
                CALL GSTXCI(14)
                CALL GTX( 2.0, 14.0, 'DONE')
                CALL GSELNT (1)
                CALL GSCHH(YDIS/50.0)
5       CALL GRQPK(1,2,II,SEG,JJ)
                CALL GSTXCI(WHITE)
C
C DONE
                IF(SEG.EQ.SEVEN) THEN
                CALL GSELNT (2)
                CALL GSCHH(1.5)
                CALL GSTXCI(DKBLUE)
                CALL GTX( 2.0, 34.0, 'NODES')
                CALL GTX(11.0, 34.0, 'ELEMENTS')
                CALL GTX(21.0, 34.0, ' B.C.')
                CALL GTX(2.0, 24.0, 'FORCES')
                CALL GTX(11.0, 24.0, 'GROUPS')
                CALL GTX(21.0,24.0, 'MATERIALS')
                CALL GTX( 2.0, 14.0, 'DONE')

                RETURN
C
C NODES
                ELSEIF(SEG.EQ.ONE) THEN
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='NODE#     X       Y       Z'
                X(1)=XMN+0.02*XDIS
                Y(1)=YMN+0.95*YDIS
                CALL GTX(X(1),Y(1),BUF)

                DO 10 I=1,NON+1,15
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='NODE#     X       Y       Z'
                X(1)=XMN+0.02*XDIS
                Y(1)=YMN+0.95*YDIS
                CALL GTX(X(1),Y(1),BUF)
                DO 12 J=1,15
                Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
                WRITE(BUF,95) I+J-1,(PN(I+J-1,JJ),JJ=1,3)
                IF(INF(I+J-1).EQ.0) GOTO 12
                CALL GTX(X(1),Y(1),BUF)
12      CONTINUE
                CALL GRQST(1,1,K,J,BUF)
                IF(BUF.NE.' ') GOTO 5
10      CONTINUE
C
C ELEMENTS
                ELSEIF(SEG.EQ.TWO) THEN
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='  ELEM#  GROUP#      NODES'
                X(1)=XMN+0.02*XDIS
                Y(1)=YMN+0.95*YDIS
                CALL GTX(X(1),Y(1),BUF)

                DO 20 I=1,NOE+1,15
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='  ELEM#  GROUP#      NODES'
                X(1)=XMN+0.02*XDIS
                Y(1)=YMN+0.95*YDIS
                CALL GTX(X(1),Y(1),BUF)
                DO 22 J=1,15
                Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
                IF(IEF(I+J-1).EQ.0) GOTO 22
                WRITE(BUF,96) I+J-1,NEC(I+J-1,9),(NEC(I+J-1,K),K=1,
   @                    NUMNOD(NOETY(NEC(I+J-1,9))))
                CALL GSTXCI(NEC(I+J-1,9))
                CALL GTX(X(1),Y(1),BUF)
22      CONTINUE
                CALL GRQST(1,1,K,J,BUF)
                IF(BUF.NE.' ') GOTO 5
20      CONTINUE
C
C B.C.
                ELSEIF(SEG.EQ.THREE) THEN
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='  NODE# Dx   Dy   Dz   Rx   Ry   Rz 1=
FIXED/0=
   @FREE'
                X(1)=XMN+0.02*XDIS
                Y(1)=YMN+0.95*YDIS
                CALL GTX(X(1),Y(1),BUF)

                DO 30 I=1,NBC+1,15
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='  NODE# Dx   Dy   Dz   Rx   Ry   Rz 1=
FIXED/0=
   @FREE'
                X(1)=XMN+0.02*XDIS
                Y(1)=YMN+0.95*YDIS
                CALL GTX(X(1),Y(1),BUF)
                DO 32 J=1,15
                IF((I+J-1).GT.NBC) GOTO 32
                Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
                WRITE(BUF,97)IDL(I+J-1),(ID(I+J-1,K),K=1,6)
                CALL GTX(X(1),Y(1),BUF)
32      CONTINUE
30      CALL GRQST(1,1,K,J,BUF)
C
C FORCES
                ELSEIF(SEG.EQ.FOUR) THEN
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='  NODE#  Fx     Fy     Fz     Mx     My     Mz '
                X(1)=XMN+0.02*XDIS
                Y(1)=YMN+0.95*YDIS
                CALL GTX(X(1),Y(1),BUF)

                DO 40 I=1,NON+1,15
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='  NODE#  Fx     Fy     Fz     Mx     My     Mz'
                X(1)=XMN+0.02*XDIS
                Y(1)=YMN+0.95*YDIS
                CALL GTX(X(1),Y(1),BUF)
                DO 42 J=1,15
                IF((I+J-1).GT.NON) GOTO 42
                Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
                WRITE(BUF,92) I+J-1,(F(I+J-1,K),K=1,6)
                CALL GTX(X(1),Y(1),BUF)
42      CONTINUE
                IF(KGRAV.EQ.1) CALL GTX(XMN,YMN,'GRAVITY
NOT YET GENERATED - RUN
   *CHECK ROUTINE')
                CALL GRQST(1,1,K,J,BUF)
                IF(BUF.NE.' ') GOTO 5
40      CONTINUE
C
C GROUPS
                ELSEIF(SEG.EQ.FIVE) THEN
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                BUF='    GROUP#  ELEM TYPE  #OF ELEMS
MATERIAL#'
                X(1)=XMN+0.02*XDIS
```

```
            Y(1)=YMN+0.95*YDIS
            CALL GTX(X(1),Y(1),BUF)

            DO 50 I=1,NOG+1,15
            CALL BOX(xmn,xmx,ymn,ymx,black,1)
            BUF='    GROUP#  ELEM TYPE  #OF ELEMS
 MATERIAL#'
            X(1)=XMN+0.02*XDIS
            Y(1)=YMN+0.95*YDIS
            CALL GTX(X(1),Y(1),BUF)
            DO 52 J=1,15
             IF((I+J-1).GT.NOG) GOTO 52
               IF(NOETY(I+J-1).EQ.1) THEN
                NAME='3D SPRING '
               ELSEIF(NOETY(I+J-1).EQ.2) THEN
                NAME='2D TRUSS  '
               ELSEIF(NOETY(I+J-1).EQ.3) THEN
                NAME='2D BEAM   '
               ELSEIF(NOETY(I+J-1).EQ.4) THEN
                NAME='GRID BEAM  (NOT 2D)'
               ELSEIF(NOETY(I+J-1).EQ.5) THEN
                NAME='3D TRUSS  '
               ELSEIF(NOETY(I+J-1).EQ.6) THEN
                NAME='3D BEAM   '
               ELSEIF(NOETY(I+J-1).EQ.7) THEN
                NAME='3D PLATE  '
               ELSEIF(NOETY(I+J-1).EQ.8) THEN
                NAME='3D BRICK  '
               ELSEIF(NOETY(I+J-1).EQ.9) THEN
                NAME='3D FRAME  '
               ELSEIF(NOETY(I+J-1).EQ.10) THEN
                NAME='2D PLAIN STRAIN '
               ELSEIF(NOETY(I+J-1).EQ.11) THEN
                NAME='2D PLANE STRESS  '
               ELSEIF(NOETY(I+J-1).EQ.12) THEN
                NAME='GRID PLATE (NOT 2D)'

               END IF
              Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
              WRITE(BUF,98) I+J-1,NOETY(I+J-1),NOEIG(I+J-1),I+J-1
              CALL GSTXCI(I+J-1)
              CALL GTX(X(1),Y(1),BUF)
              CALL GTX(XMN+0.7*XDIS,Y(1),NAME)
52    CONTINUE
50    CALL GRQST(1,1,K,J,BUF)
C
C MATERIALS
              ELSEIF(SEG.EQ.SIX) THEN
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='MATERIAL#  PROPERTIES'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)
              DO 60 I=1,NOG+1,15
               CALL BOX(xmn,xmx,ymn,ymx,black,1)
               BUF='MATERIAL#  PROPERTIES'
               X(1)=XMN+0.02*XDIS
               Y(1)=YMN+0.95*YDIS
               CALL GTX(X(1),Y(1),BUF)
               DO 62 J=1,15
               IF((I+J-1).GT.NOG) GOTO 62
               Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
               WRITE(BUF,92) I+J-1,(PMAT(I+J-
1,K),K=1,NUMMAT(NOETY(I+J-1)))
               CALL GSTXCI(I+J-1)
               CALL GTX(X(1),Y(1),BUF)
62    CONTINUE
60    CALL GRQST(1,1,K,J,BUF)

              END IF
              GOTO 5
92    FORMAT(I6,' ',9E9.2)
95    FORMAT(I7,3F9.3)
96    FORMAT(10I7)
97    FORMAT(7I7)
98    FORMAT(4I11)
              END
C
C*****************************************************
***************
C
              SUBROUTINE CHECK()
              INTEGER NNN(701)
              CHARACTER*20 BUF,DUM
              INCLUDE NCOMMON.F
              KSAV=0
              OPEN(UNIT=5,FILE='WARNINGS.OUT')
              NUMERR=0
C
C THIS ROUTINE COMPRESSES THE NODE AND ELEM #'s
C AND CHECKS FOR REPEATED POSITIONS AND NODES
C
C 100 FORCES  F(6 DOF FORCES)
C 100 BOUNDARY CONDITIONS  ID(6 DOF FIX), IDL(NODE WHICH
FIX)
C
C     ID    BOUNDARY CONDITION ARRAY  6dof
C     IDL   B.C. LOCATIONS
C     F     NODAL FORCE ARRAY
C     ICN   FLAG FOR COMPRESS NODES 1=YES/0=NO
C     ICE   FLAG FOR COMPRESS ELEMS 1=YES/0=NO
C     NNN   NNN(I)= NEW NODE NUMBER OF OLD NODE I
C     NAN   NUMBER OF ACTUAL NODES
C     NAE   NUMBER OF ACTUAL ELEMENTS
C     NUMERR  NUMBER OF ERRORS/WARNINGS
C
C     OPEN(UNIT=6,FILE='CHECK.OUT')
C COMPRESS NODES
C
C PROMPT FOR COMPRESS NODES OR NOT?
              CALL GSTXCI(ORANGE)
              CALL GTX(1.0,85.0,'COMPRESS NODES? Y/N')
              CALL GRQST(1,1,K,J,BUF)
              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
              DUM='Y'
              BUF=BUF(1:1)
              IF(DUM.EQ.BUF) ICN=1
              DUM='y'
              IF(DUM.EQ.BUF) ICN=1

              NAN=0
              DO 10 I=1,NON
               IF(INF(I).EQ.1) THEN
                NAN=NAN+1
                IF(ICN.EQ.1) THEN
                 INF(NAN)=1
                 DO 15 J=1,5
15      PN(NAN,J)=PN(I,J)
                 NNN(I)=NAN
                END IF
               END IF
              IF(ICN.EQ.0) NNN(I)=I
10    CONTINUE
              IF(ICN.EQ.1) THEN
               DO 13 I=NAN+1,700
13    INF(I)=0
              NON=NAN
              END IF
              WRITE(5,*)'NUMBER ACTUAL NODES',NAN
              IF(NAN.EQ.0) THEN
               NUMERR=NUMERR+1
               WRITE(5,*)

WRITE(5,*)'************************************'
              WRITE(5,*)'******  WARNING!! ERROR!!  ******'
              WRITE(5,*)'******                  ******'
              WRITE(5,*)'******      NO NODES      ******'

WRITE(5,*)'************************************'
              CALL GSTXCI(RED)
              CALL GTX(1.0,88.0,'*ERROR*  NO NODES')
              CALL GTX(1.0,85.0,'ENTER TO CONTINUE')
              CALL GRQST(1,1,K,J,BUF)
              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
              END IF
C
C REASSIGN FORCES TO NEW NODE NUMBERING
C
              DO 16 I=1,NON
              DO 16 J=1,6
16    F(I,J)=F(NNN(I),J)
              DO 66 I=NON+1,700
              DO 66 J=1,6
66    F(I,J)=0.0
C
C REASSIGN BOUNDARY CONDITIONS TO NEW NODE
NUMBERING
              DO 18 I=1,NBC
18    IDL(I)=NNN(IDL(I))
              DO 88 I=NBC+1,200
              DO 88 J=1,6
              IDL(I)=0
88    ID(I,J)=0
C
C CHECK FOR DOUBLED NODES
C
              DO 17 I=1,NON-1
              DO 17 J=I+1,NON
              IF(PN(I,1).EQ.PN(J,1).AND.PN(I,2).EQ.PN(J,2).AND.PN(I
,3).EQ.
```

```
     &    PN(J,3)) THEN
                  NUMERR=NUMERR+1
                  WRITE(5,*)

WRITE(5,*)'************************************'
                  WRITE(5,*)'******  WARNING!! ERROR!!  ******'
                  WRITE(5,*)'******                  ******'
                  WRITE(5,*)'****** REPEATED POSITIONS FOR
******'
                  WRITE(5,905) I,J

WRITE(5,*)'************************************'
                  CALL GSTXCI(ORANGE)
                  CALL GTX(1.0,88.0,'REPEATED NODE POSN.')
                  CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
                  CALL GRQST(1,1,K,J,BUF)
                  CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)

                  END IF
17    CONTINUE
C
C CHECK FOR FREE NODES (NOT IN ANY ELEMENT)
C
                 NUMNOW=NUMERR
      DO 22 I=1,NON
                 IJK=0
                 IF(INF(I).EQ.0) GOTO 22
                  DO 23 J=1,NOE
                   IF(IEF(J).EQ.0) GOTO 23
                    DO 24 K=1,8
                     IF(NEC(J,K).EQ.I) IJK=1
24       CONTINUE
23     CONTINUE
                 IF(IJK.EQ.0) THEN
                 NUMERR=NUMERR+1
                 WRITE(5,*)

WRITE(5,*)'************************************'
                 WRITE(5,*)'******  WARNING!! ERROR!!  ******'
                 WRITE(5,*)'******                  ******'
                 WRITE(5,*)'******   UNUSED FREE NODE   ******'
                 WRITE(5,902) I

WRITE(5,*)'************************************'
                 END IF
22    CONTINUE
                 IF(NUMERR.GT.NUMNOW) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(1.0,88.0,'*ERROR* FREE NODE/S')
                 CALL GTX(1.0,85.0,'ENTER TO CONTINUE')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                 END IF

                WRITE(5,*)'NUMBER OF BOUNDARY
CONDITIONS',NBC
                IF(NBC.EQ.0) THEN
                 NUMERR=NUMERR+1
                 WRITE(5,*)

WRITE(5,*)'************************************'
                 WRITE(5,*)'******  WARNING!! ERROR!!  ******'
                 WRITE(5,*)'******                  ******'
                 WRITE(5,*)'****** NO BOUNDARY CONDITIONS
******'

WRITE(5,*)'************************************'
                 CALL GSTXCI(RED)
                 CALL GTX(1.0,88.0,' *ERROR*  NO B.C.')
                 CALL GTX(1.0,85.0,'ENTER TO CONTINUE')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)

                 END IF
C
C COMPRESS ELEMENTS ( OOHH, TRICKY ! )
C
C   PROMPT IF COMPRESS ELEMENTS
                 CALL GSTXCI(ORANGE)
                 CALL GTX(1.0,85.0,'COMPRESS ELEMENTS? Y/N')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                 DUM='Y'
                 BUF=BUF(1:1)
                 IF(DUM.EQ.BUF) ICE=1
                 DUM='y'
                 IF(DUM.EQ.BUF) ICE=1

                NAE=0
                 DO 30 I=1,NOE

                  IF(ICN.EQ.1) THEN
                   DO 36 J=1,8
36       NEC(I,J)=NNN(NEC(I,J))
                  END IF

                 IF(IEF(I).EQ.1) THEN
                  NAE=NAE+1

                  IF(ICE.EQ.1) THEN
                   IEF(NAE)=I
                   DO 35 J=1,9
35       NEC(NAE,J)=NEC(I,J)
                  END IF

                 END IF
30    CONTINUE

                 IF(ICE.EQ.1) THEN
                  DO 33 I=NAE+1,700,1
                  IEF(I)=0
33    CONTINUE

                  NOE=NAE
                 END IF

                  WRITE(5,*)'NUMBER ACTUAL ELEMENTS',NAE
                 IF(NAE.EQ.0) THEN
                  NUMERR=NUMERR+1
                  WRITE(5,*)

WRITE(5,*)'************************************'
                  WRITE(5,*)'******  WARNING!! ERROR!!  ******'
                  WRITE(5,*)'******                  ******'
                  WRITE(5,*)'******     NO ELEMENTS     ******'

WRITE(5,*)'************************************'
                  CALL GSTXCI(RED)
                  CALL GTX(1.0,88.0,'*ERROR* NO ELEMENTS')
                  CALL GTX(1.0,85.0,' ENTER TO CONTINUE ')
                  CALL GRQST(1,1,K,J,BUF)
                  CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)

                  END IF
C
C CHECK FOR DOUBLE NODE USE
C
                 DO 40 I=1,NOE
                 DO 40 J=1,NUMNOD(NOETY(NEC(I,9)))-1
                 DO 40 K=J+1,NUMNOD(NOETY(NEC(I,9)))
                  IF(NEC(I,J).EQ.NEC(I,K).AND.IEF(I).EQ.1) THEN
                  NUMERR=NUMERR+1
                  WRITE(5,*)

WRITE(5,*)'************************************'
                  WRITE(5,*)'******  WARNING!! ERROR!!  ******'
                  WRITE(5,*)'******                  ******'
                  WRITE(5,*)'******   REPEATED NODE IN   ******'
                  WRITE(5,901) I
                  WRITE(5,902) NEC(I,J)

WRITE(5,*)'************************************'
                  CALL GSTXCI(RED)
                  CALL GTX(1.0,88.0,'REPEATED NODE IN ELEM.')
                  WRITE(BUF,907) I
                  CALL GTX(1.0,85.0,BUF)
                  CALL GRQST(1,1,K,J,BUF)
                  CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                  END IF
40    CONTINUE
C
C ADD WEIGHT OF ELEMENTS ( OOHH, TRICKY ! )
C
C PROMPT
                IF(KGRAV.EQ.1.AND.DUM.EQ.BUF) THEN
                 CALL GSTXCI(ORANGE)
                 CALL GTX(1.0,88.0,'DO GRAVITY FOR SOLVER')
                 CALL GTX(1.0,85.0,'FILE ONLY! NOW?  Y/N ')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                 DUM='Y'
                 BUF=BUF(1:1)
                 DO 75 I=1,NON
                 DO 75 J=1,6
75     F(I,J)=F(I,J)+10.0
                 KGRAV=0
                END IF

                 IF(NUMERR.GT.0) THEN
```

```
                    CALL GSTXCI(RED)
                    CALL GTX(1.0,88.0,'ERRORS WRITTEN
TO WARNINGS.OUT!')
                    CALL GTX(1.0,85.0,'    ENTER TO
CONTINUE  ')
                ELSE
                    CALL GSTXCI(GREEN)
                    CALL GTX(1.0,88.0,'    NO ERRORS
FOUND   ')
                    CALL GTX(1.0,85.0,'    ENTER TO
CONTINUE  ')
                END IF
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)

            CALL DRAW(1,'NEW ORDER',1)

            CALL GSTXCI(ORANGE)
C
            CLOSE(UNIT=5)

            RETURN
901   FORMAT('******  ELEMENT ',I7,'  ******')
902   FORMAT('******    NODE  ',I7,'  ******')
905   FORMAT('****** NODES ', 2I7, '  ******')
907   FORMAT(I4,' ', 'ENTER TO CONT.')
            END

C
C*************************************************************
**************
C
            SUBROUTINE ELEMSUB()
            CHARACTER*20 BUF,NAME,PROMPT(12,9)
            INTEGER I,J,K
            INCLUDE 'NCOMMON.F'
            CALL BOX(22.0,31.0,89.0,96.0,GREY,1)
            KSAV=0
C
C THIS SUBROUTINE  i) CREATES GROUPS
C           ii) SELECTS GROUPS
C          iii) CREATES ELEMENTS
C           iv) DELETES ELEMENTS
C           iiv) DE/ACTIVATES GROUPS
C          iiiv) SELECTS THE DEMO
C
C WRITING PROPERTY ASKING NAMES TO PROMPT MATRIX
C
            PROMPT(1,1)='SPRING CONSTANT'

            PROMPT(2,1)='ELASTICITY'
            PROMPT(2,2)='SECTION AREA'
            PROMPT(2,3)='DENSITY'

            PROMPT(3,1)='ELASTICITY'
            PROMPT(3,3)='MOMENT OF INERTIA'
            PROMPT(3,2)='SECTION AREA'
            PROMPT(3,4)='Cx MAXIMUM'
            PROMPT(3,5)='DENSITY'

            PROMPT(4,1)='ELASTICITY'
            PROMPT(4,2)='RIGIDITY G'
            PROMPT(4,3)='MOMENT INERTIA Ixx'
            PROMPT(4,4)='MOMENT INERTIA Izz'
            PROMPT(4,5)='Cx MAXIMUM'
            PROMPT(4,6)='R MAXIMUM'
            PROMPT(4,7)='DENSITY'

            PROMPT(5,1)='ELASTICITY'
            PROMPT(5,2)='SECTION AREA'
            PROMPT(5,3)='DENSITY'

            PROMPT(6,1)='ELASTICITY'
            PROMPT(6,2)='RIGIDITY G'
            PROMPT(6,3)='SECTION AREA'
            PROMPT(6,4)='Cx MAXIMUM'
            PROMPT(6,5)='Cy MAXIMUM'
            PROMPT(6,6)='MOMENT INERTIA Ixx'
            PROMPT(6,7)='MOMENT INERTIA Iyy'
            PROMPT(6,8)='MOMENT INERTIA Izz'
            PROMPT(6,9)='DENSITY'

            PROMPT(7,1)='ELASTICITY'
            PROMPT(7,2)='RIGIDITY G'
            PROMPT(7,3)='THICKNESS'
            PROMPT(7,4)='DENSITY'

            PROMPT(8,1)='ELASTICITY'
            PROMPT(8,2)='POISSONS RATIO'
            PROMPT(8,3)='DENSITY'

            PROMPT(9,1)='ELASTICITY'
            PROMPT(9,2)='RIGIDITY G'
            PROMPT(9,3)='SECTION AREA'
            PROMPT(9,4)='Cx MAXIMUM'
            PROMPT(9,5)='Cy MAXIMUM'
            PROMPT(9,6)='MOMENT INERTIA Ixx'
            PROMPT(9,7)='MOMENT INERTIA Iyy'
            PROMPT(9,8)='MOMENT INERTIA Izz'
            PROMPT(9,9)='DENSITY'

            PROMPT(10,1)='ELASTICITY'
            PROMPT(10,2)='POISSONS RATIO'
            PROMPT(10,3)='DENSITY'

            PROMPT(11,1)='ELASTICITY'
            PROMPT(11,2)='POISSONS RATIO'
            PROMPT(11,3)='DENSITY'

            PROMPT(12,1)='ELASTICITY'
            PROMPT(12,2)='RIGIDITY G'
            PROMPT(12,3)='THICKNESS'
            PROMPT(12,4)='DENSITY'

C   WRITING BUTTON NAMES
C
            CALL GSTXCI(GREEN)
C 1
            CALL GTX( 2.0, 36.0, 'CREATE')
            CALL GTX( 2.0, 33.0, 'GROUP')
C 2
            CALL GTX(12.0, 36.0, 'SELECT')
            CALL GTX(12.0, 33.0, 'GROUP')
C 3
            CALL GTX(21.0, 36.0, 'CREATE')
            CALL GTX(21.0, 33.0, 'ELEMENT')
C 4
            CALL GTX( 2.0, 24.0, 'DEMO')
C 6
            CALL GTX(21.0, 26.0, 'DELETE')
            CALL GTX(21.0, 23.0, 'ELEMENT')
C 7
            CALL GSTXCI(14)
            CALL GTX( 2.0, 14.0, 'DONE')
            CALL GSTXCI(GREEN)
C 5
            CALL GTX(12.0, 26.0, 'GROUP')
            CALL GTX(12.0, 23.0, 'ON/OFF')
C 8
            CALL GTX(12.0, 16.0, 'MODIFY')
            CALL GTX(12.5, 13.0, 'GROUP')
C 9
            CALL GTX(21.5, 16.0, 'MODIFY')
            CALL GTX(21.0, 13.0, 'ELEMENT')
C
C MAIN PICK
C
5   CALL GRQPK(1,2,II,SEG,JJ)
            CALL GSTXCI(GREEN)
            CALL GSCHH(1.5)

C
C DONE
            IF(SEG.EQ.SEVEN) THEN
            CALL GSTXCI(DKBLUE)
            CALL GSCHH(1.5)
            CALL GTX( 2.0, 36.0, 'CREATE')
            CALL GTX( 2.0, 33.0, 'GROUP')
            CALL GTX(12.0, 36.0, 'SELECT')
            CALL GTX(12.0, 33.0, 'GROUP')
            CALL GTX(21.0, 36.0, 'CREATE')
            CALL GTX(21.0, 33.0, 'ELEMENT')
            CALL GTX(21.0, 26.0, 'DELETE')
            CALL GTX(21.0, 23.0, 'ELEMENT')
            CALL GTX( 2.0, 14.0, 'DONE')
            CALL GTX(12.0, 26.0, 'GROUP')
            CALL GTX(12.0, 23.0, 'ON/OFF')
            CALL GTX( 2.0, 24.0, 'DEMO')
            CALL GTX(12.0, 16.0, 'MODIFY')
            CALL GTX(12.5, 13.0, 'GROUP')
            CALL GTX(21.5, 16.0, 'MODIFY')
            CALL GTX(21.0, 13.0, 'ELEMENT')
            RETURN
            ELSEIF(SEG.EQ.BMOUSE) THEN
                    MOKB=1
            ELSEIF(SEG.EQ.BKEYB) THEN
                    MOKB=0
C
C SELECT GROUP
            ELSEIF(SEG.EQ.TWO) THEN
```

```
                CALL GSTXCI(GREEN)
                CALL GTX(1.0,85.0,'ENTER GROUP #')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,900,ERR=99) JJ
                IF(JJ.GT.NOG.OR.JJ.LT.1) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(1.0,88.0,'GROUP DOES NOT EXIST!')
                 CALL GTX(1.0,85.0,'  ENTER TO CONTINUE ')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                ELSE
                NPRESG=JJ
                END IF
C
C DEMO
               ELSEIF(SEG.EQ.FOUR) THEN
                CALL DEMO()
C
C ELEMENT MODIFY
               ELSEIF(SEG.EQ.NINE) THEN
               CALL GSTXCI(GREEN)
               CALL GTX(1.0,88.0,'ENTER ELEMENT #')
               CALL GTX(1.0,85.0,'  TO MODIFY')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) JJ
                IF(JJ.LT.1.OR.JJ.GT.NOE) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(1.0,88.0,'ELEMENT DOES NOT EXIST ')
                 CALL GTX(1.0,85.0,'  ENTER TO CONTINUE  ')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                ELSE
                 NPRESE=JJ
                 NPRESG=NEC(NPRESE,9)
                 CALL GSTXCI(GREEN)
                  IF(NOETY(NPRESG).EQ.1) THEN
                   NAME='3D SPRING 2N'
                  ELSEIF(NOETY(NPRESG).EQ.2) THEN
                   NAME='2D TRUSS  2N'
                  ELSEIF(NOETY(NPRESG).EQ.3) THEN
                   NAME='2D BEAM   2N'
                  ELSEIF(NOETY(NPRESG).EQ.4) THEN
                   NAME='GRID BEAM  2N'
                  ELSEIF(NOETY(NPRESG).EQ.5) THEN
                   NAME='3D TRUSS  2N'
                  ELSEIF(NOETY(NPRESG).EQ.6) THEN
                   NAME='3D BEAM   6N'
                  ELSEIF(NOETY(NPRESG).EQ.7) THEN
                   NAME='3D PLATE  4N'
                  ELSEIF(NOETY(NPRESG).EQ.8) THEN
                   NAME='3D BRICK  8N'
                  ELSEIF(NOETY(NPRESG).EQ.9) THEN
                   NAME='3D FRAME  2N'
                  ELSEIF(NOETY(NPRESG).EQ.10) THEN
                   NAME='2D PLANE STRAIN 4N'
                  ELSEIF(NOETY(NPRESG).EQ.11) THEN
                   NAME='2D PLANE STRESS 4N'
                  ELSEIF(NOETY(NPRESG).EQ.12) THEN
                   NAME='GRID PLATE 4N'
                  END IF

                 DO 13 I=1,NUMNOD(NOETY(NPRESG))
                IF(MOKB.EQ.1) THEN
                CALL GTX(2.0,88.0,'REPICK NODE #s FOR')
                CALL GTX(2.0,85.0, NAME)
                CALL GRQLC(1,2,I,NT,X(1),Y(1))
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                XX=XMN+X(1)*XDIS/0.73
                YY=YMN+Y(1)*YDIS/0.74874
                CALL FIND(XX,YY,NEC(NPRESE,I) )
                NPRESN=NEC(NPRESE,I)
               CALL GSMK(4)
               CALL GSPMCI(WHITE)
               CALL GSELNT (1)
               CALL GPM(1,PN(NPRESN,4),PN(NPRESN,5))
               CALL GSELNT (2)
               ELSE
                CALL GTX(2.0,88.0,'REENTER NODE #s FOR')
                CALL GTX(2.0,85.0, NAME)
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(BUF,900,ERR=99) NEC(NPRESE,I)
               END IF
13    CONTINUE
                CALL ROTATE(0.0,0.0)
                END IF
C
C GROUP MODIFY
               ELSEIF(SEG.EQ.EIGHT) THEN
               CALL GSTXCI(GREEN)
               CALL GTX(1.0,88.0,'ENTER GROUP #')
               CALL GTX(1.0,85.0,'  TO MODIFY  ')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) JJ
                IF(JJ.LT.1.OR.JJ.GT.NOG) THEN
                 CALL GSTXCI(RED)
                 CALL GTX(1.0,88.0,'GROUP DOES NOT EXIST')
                 CALL GTX(1.0,85.0,'  ENTER TO CONTINUE')
                 CALL GRQST(1,1,K,J,BUF)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                ELSE
                 NPRESG=JJ
                 NGET=NOETY(NPRESG)
                  DO 62 I=1,NUMMAT(NGET)
                   CALL GTX(1.0,88.0,'INPUT MATERIAL
PROPERTY')
                   NAME=PROMPT(NGET,I)
C WRITE(NAME,910) I
                   CALL GTX(1.0,85.0,NAME)
                   CALL GRQST(1,1,K,J,BUF)
                   CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
62        READ(BUF,*,ERR=99) PMAT(NPRESG,I)
                END IF

C
C GROUP ON/OFF
              ELSEIF(SEG.EQ.FIVE) THEN
               IF(NPRESG.LT.1.OR.NPRESG.GT.NOG) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'NO GROUP SELECTED!')
                CALL GTX(1.0,85.0,'ENTER TO CONTINUE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               ELSE
                NOGAC(NPRESG)=-NOGAC(NPRESG)
                CALL ROTATE(0.0,0.0)
               END IF
C
C DELETE ELEMENT
              ELSEIF(SEG.EQ.SIX) THEN
               CALL GSTXCI(GREEN)
               CALL GTX(1.0,88.0,'ENTER ELEMENT TO')
               CALL GTX(1.0,85.0,'  BE DELETED')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) NPRESE
               IF(NPRESE.LT.1.OR.NPRESE.GT.NOE) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'ELEMENT DOES NOT EXIST')
                CALL GTX(1.0,85.0,'ENTER TO CONTINUE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               ELSE
                IEF(NPRESE)=0
                IJK=NEC(NPRESE,9)
                CALL ROTATE(0.0,0.0)
                NOEIG(IJK)=NOEIG(IJK)-1
               END IF
C
C ELEMENT CREATE
              ELSEIF(SEG.EQ.THREE) THEN
C OUT OF SPACE
             IF(NOE.EQ.700) THEN
              CALL GSTXCI(RED)
              CALL GTX(1.0,88.0,'700 ELEMS ALREADY EXIST')
              CALL GTX(1.0,85.0,'CONSULT PROGRAMMER ')
              CALL GRQST(1,1,K,J,BUF)
              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
              GOTO 5
              END IF
               IF(NPRESG.LT.1.OR.NPRESG.GT.NOG) THEN
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'NO GROUP SELECTED!')
                CALL GTX(1.0,85.0,'ENTER TO CONTINUE')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               ELSE
                CALL GSTXCI(GREEN)
                IF(NOETY(NPRESG).EQ.1) THEN
                 NAME='3D SPRING 2N'
                ELSEIF(NOETY(NPRESG).EQ.2) THEN
                 NAME='2D TRUSS  2N'
                ELSEIF(NOETY(NPRESG).EQ.3) THEN
                 NAME='2D BEAM   2N'
                ELSEIF(NOETY(NPRESG).EQ.4) THEN
                 NAME='2D GRID   2N'
                ELSEIF(NOETY(NPRESG).EQ.5) THEN
```

```
                NAME='3D TRUSS  2N'
                ELSEIF(NOETY(NPRESG).EQ.6) THEN
                NAME='3D BEAM   6N'
                ELSEIF(NOETY(NPRESG).EQ.7) THEN
                NAME='3D PLATE  4N'
                ELSEIF(NOETY(NPRESG).EQ.8) THEN
                NAME='3D BRICK  8N'
                ELSEIF(NOETY(NPRESG).EQ.9) THEN
                NAME='3D FRAME  2N'
                ELSEIF(NOETY(NPRESG).EQ.10) THEN
                NAME='2D PLANE STRAIN 4N'
                ELSEIF(NOETY(NPRESG).EQ.11) THEN
                NAME='2D PLANE STRESS 4N'
                ELSEIF(NOETY(NPRESG).EQ.12) THEN
                NAME='GRID PLATE 4N'

                END IF
               DO 10 I=1,NUMNOD(NOETY(NPRESG))
              IF(MOKB.EQ.1) THEN
               CALL GTX(2.0,88.0,'PICK NODE #s FOR')
               CALL GTX(2.0,85.0, NAME)
               CALL GRQLC(1,2,I,NT,X(1),Y(1))
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               XX=XMN+X(1)*XDIS/0.73
               YY=YMN+Y(1)*YDIS/0.74874
               CALL FIND(XX,YY,NEC(NOE+1,I) )
               NPRESN=NEC(NOE+1,I)
               IF( NPRESN.NE.0 ) THEN
                CALL GSMK(4)
                CALL GSPMCI(WHITE)
                CALL GSELNT (1)
                CALL GPM(1,PN(NPRESN,4),PN(NPRESN,5))
                CALL GSELNT (2)
               ELSE
                GOTO 124
               END IF
              ELSE
               CALL GTX(2.0,88.0,'ENTER NODE #s FOR')
               CALL GTX(2.0,85.0, NAME)
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) NEC(NOE+1,I)
              END IF
10    CONTINUE
               NEC(NOE+1,9)=NPRESG
               IEF(NOE+1)=1
               NOE=NOE+1
               NAE=NOE
               NOEIG(NPRESG)=NOEIG(NPRESG)+1
124     CALL ROTATE(0.0,0.0)
               END IF
C
C GROUP CREATE
             ELSEIF(SEG.EQ.ONE) THEN
              IF(NOG.EQ.25) THEN
C GROUPS FULL
               CALL GSTXCI(RED)
               CALL GTX(1.0,88.0,'25 GROUPS ALREADY EXIST!')
               CALL GTX(1.0,85.0,'  ENTER TO CONTINUE   ')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
C INPUT ELEM TYPE
              ELSE
55      CALL GSTXCI(GREEN)
               CALL GTX(2.0,88.0,'INPUT ELEMENT TYPE #')
               CALL GTX(2.0,85.0,'  FOR NEW GROUP   ')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               READ(BUF,900,ERR=99) NGET
               IF(NGET.LT.1.OR.NGET.GT.12) THEN
C NOT A GROUP
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'ELEMENT TYPE DOES')
                CALL GTX(1.0,85.0,'  NOT EXIST!  ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 5
                END IF
               IF(NODIM.EQ.1) THEN
C FIRST GROUP SELECTS DIM IF NODIM=1
                        NODIM=NUMDIM(NGET)
                        END IF
               IF(NODIM.NE.NUMDIM(NGET)) THEN
C NOT SAME DIMENSIONS
                CALL GSTXCI(RED)
                CALL GTX(1.0,88.0,'ELEMENT NOT SAME ')
                CALL GTX(1.0,85.0,' DIMENSION TYPE! ')
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                GOTO 5
                END IF
               DO 60 I=1,NUMMAT(NGET)
                CALL GTX(1.0,88.0,'INPUT MATERIAL
PROPERTY')
C  WRITE(NAME,910) I
                NAME=PROMPT(NGET,I)
                CALL GTX(1.0,85.0,NAME)
                CALL GRQST(1,1,K,J,BUF)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
60        READ(BUF,*,ERR=99) PMAT(NOG+1,I)
                NOG=NOG+1
                NOETY(NOG)=NGET
                NOGAC(NOG)=1
    END IF
   END IF
C LOOP END
             GOTO 5
C ERROR HANDLER
99     CALL GSTXCI(RED)
              CALL GTX(2.0,88.0,' ERROR IN INPUT')
              CALL GTX(2.0,85.0,'ENTER TO CONTINUE')
              CALL GRQST(1,1,K,J,BUF)
              CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
              GOTO 5

900    FORMAT(I6)
905    FORMAT(F8.2)
910    FORMAT(' PROPERTY #',I4)

              END
C
C*************************************************************
**************
C
              SUBROUTINE META()
              CHARACTER*20 BUF
              INCLUDE 'NCOMMON.F'
C
C THIS SUBROUTINE GENERATES A COMPUTER GRAPHICS
METAFILE
C
C INPUT TITLE OF THIS PLOT
C
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               CALL GSTXCI(GREEN)
               CALL GTX(2.0,88.0,' ENTER TITLE FOR ')
               CALL GTX(2.0,85.0,'THIS GRAPHIC PLOT')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
C
C DEACTIVATING SCREEN WORKSTATION
C ACTIVATING METAFILE WORKSTATION
C
              CALL GACWK (2)
              CALL GDAWK (1)
C
C DRAW TO METAFILE
C
              CALL GCLRWK(2,1)
              CALL DRAW(0,BUF,1)
C
C TURN METAFILE OFF / TURN SCREEN ON
C
              CALL GACWK (1)
              CALL GDAWK (2)
              RETURN
              END
C
C*************************************************************
**************
C
              SUBROUTINE DEMO()
              CHARACTER*20 BUF
              INCLUDE 'NCOMMON.F'
C
C THIS SUBROUTINE DRAWS A DEMO OF ALL THE ELEMENTS
C AND SHOWS INFORMATION ABOUT THEM.
C
C CLEAR THE GRAPHIC BOX
C AND RESET IT FOR DEMO
C
              CALL GSELNT (1)
              CALL BOX(XMN,XMX,YMN,YMX,BLACK,1)
              XMN=-2.0
              YMN=-1.0
              XMX=39.0
              YMX=31.0
              CALL GSWN (1,XMN,XMX,YMN,YMX)
              CALL BOX(XMN,XMX,YMN,YMX,DKBLUE,0)
              CALL GSCHH(0.75)
```

```
            CALL GSPMCI(YELLOW)
            CALL GSMK(KOMT)
C
C TITLE
C
            CALL GSTXCI(YELLOW)
            CALL GTX(1.0,29.0,'    ELEMENTS
MATERIAL
  &PROPERTIES    DOFs')
C
C 2D TRUSS
            CALL GSPLCI(NUMCOL(2))
            X(1)=1.0
            X(2)=6.0
            Y(1)=27.0
            Y(2)=Y(1)
            CALL GPL(2,X,Y)
            CALL GPM(2,X,Y)
            CALL GTX(-1.0,Y(1),'2')
            CALL GTX(7.0,Y(1),'2D TRUSS  2 NODES    E A
  & X Y ')
C 2D BEAM
            CALL GSPLCI(NUMCOL(3))
            X(1)=1.0
            X(2)=6.0
            Y(1)=25.0
            Y(2)=Y(1)
            CALL GPL(2,X,Y)
            CALL GPM(2,X,Y)
            CALL GTX(-1.0,Y(1),'3')
            CALL GTX(7.0,Y(1),'2D BEAM   2 NODES    E A I
  & X Y Rz')
C 2D PLANE STRAIN
            CALL GSFACI(NUMCOL(10))
            CALL BOX(1.0,6.0,20.0,23.0,NUMCOL(10),1)
            CALL BOX(1.0,6.0,20.0,23.0,WHITE,0)
            X(1)=1.0
            X(2)=6.0
            X(3)=6.0
            X(4)=1.0
            X(5)=X(1)
            Y(1)=20.0
            Y(2)=20.0
            Y(3)=23.0
            Y(4)=23.0
            CALL GPM(4,X,Y)
            CALL GTX(-1.0,Y(1),'10')
            CALL GTX(7.0,21.0,'2D PLANE STRAIN  4 NODES    E
v
  & X Y ')
C 2D PLANE STRESS
            CALL GSFACI(NUMCOL(11))
            CALL BOX(1.0,6.0,15.0,18.0,NUMCOL(11),1)
            CALL BOX(1.0,6.0,15.0,18.0,WHITE,0)
            Y(1)=15.0
            Y(2)=15.0
            Y(3)=18.0
            Y(4)=18.0
            CALL GPM(4,X,Y)
            CALL GTX(-1.0,Y(1),'11')
            CALL GTX(7.0,16.0,'2D PLANE STRESS  4 NODES    E
v
  & X Y ')
C GRID BEAM
            CALL GSPLCI(NUMCOL(4))
            X(1)=1.0
            X(2)=6.0
            Y(1)=13.0
            Y(2)=Y(1)
            CALL GPL(2,X,Y)
            CALL GPM(2,X,Y)
            CALL GTX(-1.0,Y(1),'4')
            CALL GTX(7.0,13.0,'GRID BEAM 2 NODES    E G Ixx
Izz CmaxX Rmax
  &    Z Rx Ry ')
C GRID PLATE
            CALL GSFACI(NUMCOL(12))
            CALL BOX(1.0,6.0,8.0,11.0,NUMCOL(12),1)
            CALL BOX(1.0,6.0,8.0,11.0,WHITE,0)
            Y(1)=8.0
            Y(2)=8.0
            Y(3)=11.0
            Y(4)=11.0
            CALL GPM(4,X,Y)
            CALL GTX(-1.0,Y(1),'12')
            CALL GTX(7.0,9.0,'GRID PLATE  4 NODES    E v t
  &    Z Rx Ry ')

C
C END PROMPT
C
            CALL GSTXCI(GREEN)
            CALL GTX(0.0,0.1,'              ENTER FOR NEXT
SCREEN')
            CALL GRQST(1,1,K,J,BUF)
C
C DRAW SECOND SCREEN FULL
C
            CALL BOX(XMN,XMX,YMN,YMX,BLACK,1)
            CALL BOX(XMN,XMX,YMN,YMX,DKBLUE,0)
C
C TITLE
C
            CALL GTX(1.0,29.0,'    ELEMENTS
MATERIAL
  &PROPERTIES    DOFs')

C 3D FRAME
            CALL GSPLCI(NUMCOL(9))
            X(1)=1.0
            X(2)=6.0
            Y(1)=25.0
            Y(2)=Y(1)
            CALL GPL(2,X,Y)
            CALL GPM(2,X,Y)
            CALL GTX(-1.0,Y(1),'9')
   CALL GTX(7.0,25.0,'3D FRAME 2 NODES  E G A CmaxX CmaxY
Ixx Iyy Izz
  & X Y Z Rx Ry Rz')

C 3D SPRING
            CALL GSPLCI(NUMCOL(1))
            X(1)=1.0
            X(2)=6.0
            Y(1)=23.0
            Y(2)=Y(1)
            CALL GPL(2,X,Y)
            CALL GPM(2,X,Y)
            CALL GTX(-1.0,Y(1),'1')
            CALL GTX(7.0,Y(1),'3D SPRING  2 NODES    K
  & X Y Z')
C 3D TRUSS
            CALL GSPLCI(NUMCOL(5))
            X(1)=1.0
            X(2)=6.0
            Y(1)=21.0
            Y(2)=Y(1)
            CALL GPL(2,X,Y)
            CALL GPM(2,X,Y)
            CALL GTX(-1.0,Y(1),'5')
            CALL GTX(7.0,Y(1),'3D TRUSS  2 NODES    E A
  & X Y Z')
C 3D BEAM
            CALL GSFACI(NUMCOL(6))
            X(1)=1.0
            X(2)=6.0
            X(3)=6.0
            X(4)=1.0
            X(5)=X(1)
            Y(1)=18.0
            Y(2)=Y(1)
            Y(3)=17.0
            Y(4)=Y(3)
            Y(5)=Y(1)
            CALL GSFAIS(1)
            CALL GFA(5,X,Y)
            CALL GSFAIS(0)
            CALL GSFACI(WHITE)
            CALL GFA(5,X,Y)
            CALL GPM(4,X,Y)
            Y(1)=17.5
            Y(2)=17.5
            CALL GPM(2,X,Y)
            CALL GTX(-1.0,Y(1),'6')
   CALL GTX(7.0,17.0,'3D BEAM  6 NODES  E G A CmaxX CmaxY
Ixx Iyy Izz
  & X Y Z Rx Ry Rz')
C 3D PLATE
            CALL GSFACI(NUMCOL(7))
            CALL BOX(1.0,6.0,11.0,15.0,NUMCOL(7),1)
            CALL BOX(1.0,6.0,11.0,15.0,WHITE,0)
            Y(1)=11.0
            Y(2)=11.0
            Y(3)=15.0
            Y(4)=15.0
            CALL GPM(4,X,Y)
            CALL GTX(-1.0,Y(1),'7')
            CALL GTX(7.0,11.0,'3D PLATE  4 NODES    E G v t
  & X Y Z Rx Ry ')
C 3D SOLID
```

```
                CALL BOX(1.0,5.0,3.0,7.0,NUMCOL(8),1)
                CALL BOX(1.0,5.0,3.0,7.0,WHITE,0)
                X(1)=1.0
                Y(1)=3.0
                CALL GPM(1,X,Y)
                X(2)=5.0
                X(3)=6.0
                X(4)=2.0
                X(5)=X(1)
                Y(1)=7.0
                Y(2)=Y(1)
                Y(3)=8.0
                Y(4)=Y(3)
                Y(5)=Y(1)
                CALL GSFACI(NUMCOL(8))
                CALL GSFAIS(1)
                CALL GFA(5,X,Y)
                CALL GSFAIS(0)
                CALL GSFACI(WHITE)
                CALL GFA(5,X,Y)
                CALL GPM(4,X,Y)
                X(1)=5.0
                X(2)=6.0
                X(3)=6.0
                X(4)=5.0
                X(5)=X(1)
                Y(1)=3.0
                Y(2)=4.0
                Y(3)=8.0
                Y(4)=7.0
                Y(5)=Y(1)
                CALL GSFACI(NUMCOL(8))
                CALL GSFAIS(1)
                CALL GFA(5,X,Y)
                CALL GSFAIS(0)
                CALL GSFACI(WHITE)
                CALL GFA(5,X,Y)
                CALL GPM(4,X,Y)
                CALL GTX(-1.0,Y(1),'8')
                CALL GTX(7.0,5.,'3D BRICK  8 NODES    E  V
     & X Y Z')

C
C END PROMPT
C
                CALL GSTXCI(GREEN)
                CALL GTX(0.0,0.1,'              ENTER TO RETURN')
                CALL GRQST(1,1,K,J,BUF)
                CALL SEARCH()
                CALL DRAW(1,'   ',0)
                RETURN
                END
C
C************************************************************
**************
C
                SUBROUTINE SORT()
                REAL EDTVP(700)
                INCLUDE 'NCOMMON.F'
C       OPEN(UNIT=5,FILE='SORT.DAT')
C
C THIS SUBROUTINE SORTS THE ELEMENT FROM THE CLOSEST
TO
C THE FURTHEST AND DRAWS THEM FAR TO CLOSE IN DRAW
C
C NOOSE...NUMBER OF ORDER SORTED ELEMENTS
C EDTVP...ELEMENT DISTANCE TO VIEW POINT
C
C CALCULATE VIEW POINT
C
                XV=100000.0*COS(AS)*COS(AP)
                YV=100000.0*SIN(AS)
                ZV=100000.0*COS(AS)*SIN(AP)
C       WRITE(5,*)' VIEW POINT ',XV,YV,ZV
C
C CALCULATE EDTVP's
C
C       WRITE(5,*) 'NOE=',NOE
                DO 20 I=1,NOE
                NOOSE(I)=I
C       WRITE(5,*)' ELEMENT #',I
C AVERAGE OF ELEMENT
                SX=0.0
                SY=0.0
                SZ=0.0
                DO 30 J=1,NUMNOD(NOETY(NEC(I,9)))
                SX=SX+PN(NEC(I,J),1)
                SY=SY+PN(NEC(I,J),2)
30      SZ=SZ+PN(NEC(I,J),3)
                SX=SX/NUMNOD(NOETY(NEC(I,9)))
                SY=SY/NUMNOD(NOETY(NEC(I,9)))
                SZ=SZ/NUMNOD(NOETY(NEC(I,9)))
C       WRITE(5,*)'AVERAGE POINT',SX,SY,SZ
C FIND DISTANCE
20      EDTVP(I)= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2 )**0.5
C20     WRITE(5,*)'DISTANCE=',EDTVP(I)
C
C SORTING ROUTINE (BUBBLE DOWN)
C
                DO 40 I=NOE-1,2,-1
                DO 40 J=1,I
C       WRITE(5,*) I,J
                IF(EDTVP(NOOSE(J)).LT.EDTVP(NOOSE(J+1)))
THEN
                IJK=NOOSE(J)
                NOOSE(J)=NOOSE(J+1)
                NOOSE(J+1)=IJK
                END IF
40      CONTINUE
C       WRITE(5,*)' SORTED ORDER '
C       DO 50 I=1,NOE
C50     WRITE(5,*) NOOSE(I)
C       WRITE(5,*)' DISTANCE ORDER '
C       DO 60 I=1,NOE
C60     WRITE(5,*)EDTVP(NOOSE(I))
C
C       CLOSE(UNIT=5)
                RETURN
                END
```

Appendix B4

SOLVER PROGRAM

```
      Program MHYFECS_SOLVER
C
C  THIS IS THE IN-CORE SOLVER FOR THE MHYFECS SERIES
C  OF FINITE ELEMENT PROGRAMS. IT IS DERIVED FROM
C  DR. L. M. AYARI'S BASE PROGRAM "FRAME" WITH HIS
PERMISSION.
C
C  MODIFICATIONS WRITTEN BY
C
C     i) DOUG G. SCOTT
C        M.Sc. STUDENT
C
C     ii) EILEEN A. ZHU
C        M.Sc. STUDENT
C
C*************************************************************
************
C
C  GLOBAL VARIABLES
C
C  NPOIN    NUMBER OF NODES
C  NELEM    NUMBER OF ELEMENTS
C  NGRP     NUMBER OF MATERIAL AND GEOMETRICAL
GROUPS
C  NLOAD    NUMBER OF LOAD CASES
C  ISTRTP   STRUCTURE TYPE
C  NCOOR    NUMBER OF SPATIAL COORDINATES
C  NID      NUMBER OF DEGREES OF FREEDOM PER NODE (
GLOBAL )
C  NDOF     TOTAL NUMBER OF DEGREES OF FREEDOM PER
ELEMENT (LOCAL)
C  NMATPR   NUMBER OF CONSTANTS DEFINING THE
MATERIAL AND GEOMETRICAL
C         PROPERTIES
C  NNODE    NUMBER OF NODES PER ELEMENT
C  NSIZK    SIZE OF THE STIFFNESS MATRIX IN LOCAL
COORDINATE SYSTEM
C  NTRIG    NUMBER OF DIRECTION COSINS PER ELEMENT
C  NSIZG    SIZE OF THE ARRAY IN WHICH THE GLOBAL
STIFFNESS MATRIX
C         IS STORED
C  NLODNO   NUMBER OF LOADED NODES
C  NLODEL   NUMBER OF LOADED ELEMENTS
C
C  PROPS(NGRP,NMATRP)   ARRAY OF ELEMENT PROPERTIES
C  COORD(NCOOR,NPOIN)   ARRAY OF NODAL COORDINATES
C  ID(NID,MPOIN)        ARRAY OF RESTRAINTS
C  LNODS(NNODE,NELEM)   ARRAY OF ELEMENT
CONNECTIVITY
C                (NODE-1, NODE-2)
C  MATNO(NELEM)         ARRAY OF ELEMENT PROPERTY
NUMBERS
C  ELTRIG(NTRIG,NELEM)  ARRAY OF DIRECTION COSINES FOR
ALL ELENENTS
C  STFLOC(NSIZK,NSIZK)  LOCAL STIFFNESS MATRIX FOR
EACH ELEMENT
C  MAXA(NEQ+1)          VECTOR OF THE ADDRESS IN VECTOR-
(A) OF WHICH
C                GLOBAL STIFFNESS TERMS ARE STORED ON THE
C                DIAGONAL
C  STIFGL             THAT PORTION OF THE GLOBAL STIFFESS
MATRIX
C                STORED
C*************************************************************
************
C
C
      implicit real*8(a-h,o-z)
      include 'control.cmn'
C
      include 'pointers.cmn'
C     POINTERS TO MEMORY LOCATIONS IN P VECTOR
C
      include 'ios.cmn'
C     INPUT/OUTPUT CONTROLLER
C
C       COMMON A VECTOR IN WHICH ALL DATA IS STORED
C
      common A(29000)
      mtot  = 29000
C
C OPEN INPUT/OUTPUT FILES
      call openfil()
C
C READS MANAGING DATA NON,NOE ECT.
      call manage()
C
C INPUT PROBLEM DATA
      call inputdat()
C
C ECHO INPUT DATA
C
      call echo
     * (a(Pcoord), a(Pid    ), a(Plnods ), a(Pmatno ),
     *  a(Pnelempg),a(Pmaxdof))
C
C DEVELOP ELEMENTAL STIFFNESS MATRICES
C
      call stiff
     * (a(Plnods ), a(Pcoord ), a(Pmatno ),  a(Pprops ), a(Pstfloc),
     *  a(Peltrig), a(Pnelempg), a(Piposek))
C
C ASSEMBLE LOCAL K INTO GLOBAL K
C
      call assemb
     * (a(Plnods ), a(Pid    ), a(Pstfloc), a(Peltrig) ,
     *  a(Pmaxa  ), a(Pgstif ), a(Pnelempg),a(Piposek) ,
     *  a(Pmaxdof))
C
C GO THRU LOADING STEPS
C
      do 10 iload=1,nload
                  read(input,*,ERR=99)nlodno,nlodel
                  WRITE(GRAPH,*)NLODNO,0
                  call load
     *  (a(Plnods),a(Pid)   ,a(Pcoord) ,a(Peltrig),
     *   a(Prhs)  ,a(Pfea  ),a(Pmaxdof),a(Pnelempg))
                  call sky
     *  (a(Pgstif ), a(Prhs  ), a(Pmaxa ),neq,iload)
                  call wrtdis
     *  (a(Pid    ), a(Pcoord), a(Prhs  ), a(Pdisnod), a(Pmaxdof))

                  call intfor(a(Pid),a(Plnods),a(Pfea),a(Pstfloc),a(Peltrig),
     *           a(Pdisnod),a(Pmaxdof), a(Pnelempg),a(Piposek),
     *           a(Pmatno),a(Pprops),a(Pcoord))

                  call react(a(Pid),a(Plnods),a(Pfea),a(Pstfloc),a(Peltrig),
     *          a(Pdisnod), a(Pmaxdof), a(Pnelempg),a(Piposek))
10    continue
      write(*,*)'SOLUTION FINISHED'
      STOP
C
C END OF MAIN BODY
C
99    write(output,*)
     * '***error occured while reading loadings***'
      write(*,*)
     * '***error occured while reading loadings***'

      stop
      end
C*************************************************************
*******
      subroutine openfil()
c
      character*12 WORD,DETAIL,GFILE,buf,dum,loc
      include 'ios.cmn'
      input=1
      output=2
      graph=3
c
      write(*,*)'ENTER INPUT FILENAME...'
      read(*,*) word
      open(unit=input,file=word,status="old",err=1)
      DO 700 I=1,11
      DUM=WORD(1:(13-I))
      BUF=WORD(1:(12-I))
700   IF (BUF.EQ.DUM) IPL=12-I
      IPL=IPL-3
      DETAIL=WORD(1:IPL)//'OUT'
      GFILE=WORD(1:IPL)//'STS'
      loc=word(1:ipl)//'loc'
      open(unit=output,file=DETAIL)
      open(unit=graph,file=GFILE)
      open(unit=4,file=loc)

c
      return
1     write(*,*)' DATAFILE DOES NOT EXIST! '
      write(*,*)'   PROGRAM ABORTED.    '

      stop
```

```
      end
c
C*************************************************************
*******
      subroutine manage()
c
      implicit real*8 (a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'

                  read(input,*,err=1)npoin,nan,nelem,nae,ncoor
     *              ,ngrp,nrestd,nmatrls,nload
c
c write header for graphic file
c
                  write(graph,99,err=1)npoin,nan,nelem,nae,ncoor
     *              ,ngrp,nrestd,nmatrls,nload

c     # of nodes ( user defined )
c     # of actual nodes ( compress version )
c     # of elements ( user defined )
c     # of actual elements ( compressed version )
      return
1     write(*,*)' ERROR IN READING CONTROL DATA  '
      stop
99    format(10i7)
      end
C-------------------------------------------------------------
      subroutine elematt(nelempg)
c
c Subroutine to set up the element attributes
c
      implicit real*8 (a-h,o-z)
      include 'control.cmn'
      dimension nelempg(ngrp,1)
c
c element type            \
c number in group          \
c material type             > per each group
c beginning element number  /
c ending element number    /

      istrtp= nelempg(igrp,1)
      nelemg= nelempg(igrp,2)
      matyp=  nelempg(igrp,3)
      elembgn=nelempg(igrp,4)
      elemend=nelempg(igrp,5)
c
      if(istrtp.eq.1)then
C
C 3D SPRING
C
c ncoor....number of coordinates
c nmatpr...number of material properties
c nsizk....local K matrix size
c ntrig....number of trigonemetric values
c nnode....number of nodes
c ndof.....local dofs
c nid......global dof per node
                  ncoor=3
                  nmatpr=2
                  nsizk=2
                  ntrig=3
                  nnode=2
                  ndof=nsizk
                  nid=3

      elseif(istrtp.eq.2)then
c 2D truss
                  nid=2
                  ncoor=2
                  nmatpr=3
                  nsizk=2
                  ntrig=2
                  nnode=2
                  ndof=nsizk

      elseif(istrtp.eq.3)then
C 2D frame
                  nid=3
                  ncoor=2
                  nmatpr=5
                  nsizk=6
                  ntrig=2
                  nnode=2
                  ndof=nsizk

      elseif(istrtp.eq.4)then
c grid
                  nid=3
                  ncoor=2
                  nmatpr=5
                  nsizk=6
                  ntrig=2
                  nnode=2
                  ndof=nsizk

      elseif(istrtp.eq.5)then
c 3D truss
                  nid=3
                  ncoor=3
                  nmatpr=3
                  nsizk=2
                  ntrig=3
                  nnode=2
                  ndof=nsizk

      elseif(istrtp.eq.6)then
c 3D BEAM 6 NODED WITH HEIGHT
                  nid=6
                  ncoor=3
                  nmatpr=9
                  nsizk=36
                  ntrig=3
                  nnode=6
                  ndof=nsizk

      elseif(istrtp.eq.7)then
c 3D PLATE
                  nid=6
                  ncoor=3
                  nmatpr=4
                  nsizk=24
                  ntrig=3
                  nnode=4
                  ndof=24

      elseif(istrtp.eq.8)then
c 3D BRICK
                  nid=3
                  ncoor=3
                  nmatpr=3
                  nsizk=24
                  ntrig=3
                  nnode=8
                  ndof=24

      elseif(istrtp.eq.9)then
c 3D BEAM 2 NODE
                  nid=6
                  ncoor=3
                  nmatpr=9
                  nsizk=12
                  ntrig=3
                  nnode=2
                  ndof=nsizk

      end if
c
c position of the degrees of freedom
c works out that all our elements use
c this default ordering X,Y,Z,Rx,Ry,Rz
c
      do 4 iid=1,nid
       iposid(iid)=iid
4     continue
c
      return
      end

C*************************************************************
****
      subroutine inputdat()

C---------------------------------------
C  This subroutine reads the data input
C---------------------------------------
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'pointers.cmn'
      include 'ios.cmn'
c
      common a(29000)
c-------------------------------------------------------------
c
c Coord Pointer
      Pcoord=1
      write(*,*)' READING IN DATA  '
c
```

```
c read in nodal location data
      call inputcoord(a(Pcoord))
c
c pointers
c
c pointer to material number of elements after coords
      Pmatno=Pcoord+npoin*ncoor
c
c pointer to group info et#,#/g,mat#,bgn#,end# after mat. list
      Pnelempg=Pmatno+(nelem+1)/2
c
c number of dof at each node pointer after group info
      Pmaxdof=Pnelempg+(5*ngrp+1)/2
c
c position of each elemental stiffness matrix after max dof
      Piposek=Pmaxdof+(npoin+1)/2
c
c  pointer to connectivity positions of Ke's
      Plnods=Piposek+(nelem+1)/2
c
c input the group and connectivity data
c
      call inputlnods(a(Plnods), a(Pnelempg), a(Pmatno),
     *              a(Pmaxdof),a(Piposek))
c
c find max # of N, Ke size, Trig size, int forces
c
      do 10 igrp=1,ngrp
       call elematt(a(Pnelempg))
c
c find max of #of nodes, Ke sizes, # of Trig values,
       Icountlnods=max0(Icountlnods,nnode)
       Icountstifloc=max0(Icountstifloc,nsizk)
       Icounteltrig=max0(Icounteltrig,ntrig)
       Icounterfea=max0(icounterfea,nsizk)
10     continue
c
c  pointer to boundary conditions after connectivity
      Pid=Plnods+(nelem*Icountlnods+1)/2
c
c boundary read in at Pid
c
      call inputid( a(Pid),a(Pmaxdof),nsizid )
c
c pointer to number of properties per group type after ID
      Pnumprop=Pid+(npoin*nsizid+1)/2
c
c pointer to properties after # of props per group
      Pprops=Pnumprop+(nmatrls+1)/2
c
c read in props at appropriate pointer
c
      call inputprops(a(Pprops),a(Pnumprop),nsizprop)
c
c     pointer to Ke's after properties
      Pstfloc=Pprops+nmatrls*nsizprop
c
c     pointer to trigonometry values after Ke's
      Peltrig=Pstfloc+nelem*Icountstifloc*Icountstifloc
c
c     pointer to row widths after trigs
      Pmaxa=Peltrig+nelem*Icounteltrig
c
c     pointer to global stiffnesses after row widths
      Pgstif=Pmaxa+(neq+2)/2
c
c finding column heights ( row widths )
c
      call colht(a(Plnods),a(Pid   ) , a(Pmaxa ), a(Pnelempg),
     *          a(Pmaxdof))
c
c pointer to right hand side force vector after Kglobal
      Prhs=Pgstif+nsizg
c
c pointer to solution vector after force vector
      Pfea=Prhs+neq
c
c pointer to reaction forces after U vector
      Pdisnod=Pfea+nelem*Icounterfea
c
c pointer to nodal displacements after reaction forces
      Pend=Pdisnod+npoin*nsizid
      write(*,*) ' MAX-DYNAMIC MEMORY ALLOCATED=29000 '

      if(pend.gt.29000) THEN
                  write(*,*)' PROBLEM TOO LARGE  ',pend
                  stop
      ELSE
                  write(*,*)' PROBLEM FITS IN MEMORY ',pend
      END IF
c
      return
      end
C*************************************************************
****************
      subroutine inputcoord(coord)
c
      implicit real*8 (a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension coord(npoin, 1)

c
c read in nodal locations
c
      do 10 kpoin=1,npoin
       read(input,*,err=1)
     *   ipoin,(coord(ipoin,icoor),icoor=1,ncoor)
                  write(graph,99) ipoin,(coord(ipoin,icoor),icoor=1,ncoor)
10    continue
                  return
c
c error
c

1      write(output,*)
     * '***error occured while reading coordinates***'
      write(*,*)' ERROR OCCURED WHILE READING COORDINATES '

       stop
99    format(i7,3f10.4)
      end
c*************************************************************
****************
      subroutine inputlnods(lnods,nelempg,matno,maxdof,iposek)
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension lnods(nelem,1), nelempg(ngrp,1), matno(1),
     *          maxdof(1     ), iposek(     1)

c
c read in group properties
c
      do 40 igrp=1,ngrp
       read(input,*,err=1)istrtp,nelemg,matyp
       write(graph,98)istrtp,nelemg,matyp
c
       nelempg(igrp,1)=istrtp
       nelempg(igrp,2)=nelemg
       nelempg(igrp,3)=matyp
c
c element storage pointer calculations
c
       if(igrp.eq.1)then
                  nelempg(igrp,4)=1
       else
                  nelempg(igrp,4)=nelempg(igrp-1,5)+1
       end if
       nelempg(igrp,5)=nelempg(igrp,4)+nelemg-1
       call elematt(nelempg)
c
c read in elements in group
c
       do 20 kelem=1,nelemg
                  read(input,*,err=1)
     *      i,(lnods(i,inode),inode=1,nnode)
                  write(graph,99,err=1)
     *      i,(lnods(i,inode),inode=1,nnode)

c
c find max dof at each node
c
                  do 30 inode=1,nnode
                   knode=lnods(i,inode)
                   maxdof(knode)=max0(maxdof(knode),nid)
30       continue
                  matno(i)=matyp
20     continue
40     continue
c
c
       icount=1
       do 25 igrp=1,ngrp
c
c set up element properties
c
```

```
      call elematt(nelempg)
      do 35 ielem=elembgn,elemend
c
c positions of local stiffness matrices
c
                  iposek(ielem)=icount
                  icount=icount+nsizk*nsizk
35    continue
25    continue
c...
c echo maxdof array
c
      write(output,*)'Maxdof array'
      write(output,*)'************'
      do 45 ipoin=1,npoin
       write(output,22)ipoin,maxdof(ipoin)
45    continue
22    format(i5,i7)
98    format(3i7)
99    format(9i7)
                  return
c...
c
c error
c

1     write(output,*)
     * '***error occured while reading connectivity matrix***'
      write(*,*)'ERROR OCCURED WHILE READING CONNECTIVITY
MATRIX'

      stop

      end
c*************************************************************
*****************
      subroutine inputid(id,maxdof,nsizid)
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension id(npoin,1),maxdof(1)

c
c read nodal restrains
c
      do 5 ipoin=1,nrestd
       read(input,*,err=1)
     *    jpoin,(id(jpoin,m),m=1,maxdof(jpoin))
       write(graph,99)
     *    jpoin,(id(jpoin,m),m=1,maxdof(jpoin))

5     continue
c
c recompute the id matrix
c
      neq=0
      do 10 i=1,npoin
                  nsizid=max0(nsizid,maxdof(i))
                  do 20 j=1,maxdof(i)
                    if(id(i,j).eq.0)then
                      neq=neq+1
                      id(i,j)=neq
                    else
                      id(i,j)=0
                    end if
20    continue
10    continue
99    format (7i7)
                  return
c
c error
c

1     write(output,*)
     * '***error occured while reading fixity conditions***'
      WRITE(*,*)' ERROR OCCURED WHILE READING BOUNDARY
CONDITIONS '

      stop
      end
c
c*************************************************************
*****************
      subroutine inputprops(props,numprop,nsizprop)
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension props(nmatrls,1),numprop(1)

c
      nsizprop=0
      do 10 imaterial=1,nmatrls
       read(input,*,err=1)jmaterial,nprop
       numprop(imaterial)=nprop
       read(input,*,err=1)
     *    (props(jmaterial,iprop),iprop=1,nprop)
       nsizprop=max0(nsizprop,nprop)
10    continue
                  return
c

1     write(output,*)
     * '***error occured while reading material properties***'
      write(output,*)
     * '      ------for group No.',imaterial,'------'
      WRITE(*,*) ' ERROR OCCURED WHILE READING MATERIAL
PROPERTIES '

      stop
      end
c
c*************************************************************
*******
      subroutine echo(coord,id,lnods,matno,nelempg,maxdof)
c
      implicit real*8 (a-h,o-z)
      include 'control.cmn'
      include 'pointers.cmn'
      include 'ios.cmn'
c
      dimension coord(npoin,1),lnods(nelem,1),nelempg(ngrp,1),
     *      matno(1     ),id(npoin,1),maxdof(1)
c
      write(output,100)
     *npoin,nelem,ncoor,ngrp,nrestd,nmatrls,nload
      write(output,150)
     *Pcoord  ,Pmatno ,Pnelempg,Pmaxdof,Piposek,Plnods,Pid   ,
     *Pnumprop,Pprops ,Pstfloc ,Peltrig,Pmaxa  ,Pgstif,
     *Prhs    ,Pfea   ,Pdisnod ,Pend

c
      write(output,200)
      do 5 ipoin=1,npoin
       write(output,300)ipoin,(coord(ipoin,idim),idim=1,ncoor)
5     continue
c
      write(output,320)
      do 10 ipoin=1,npoin
       write(output,350)ipoin,(id(ipoin,idof),idof=1,maxdof(ipoin))
10    continue
c
      write(output,400)
      do 15 igrp=1,ngrp
       call elematt(nelempg)
       do 20 ielem=elembgn,elemend
                  write(output,500)ielem,matno(ielem),
     *            (lnods(ielem,inode),inode=1,nnode)
20    continue
15    continue
c
      return
c------------------
c Format statements
c------------------
100   format(///,'****Control Parameters****',//,
     *      'npoin.....',i4,/,
     *      'nelem.....',i4,/,
     *      'ncoor.....',i4,/,
     *      'ngrp......',i4,/,
     *      'nrestd....',i4,/,
     *      'nmatrls...',i4,/,
     *      'nload.....',i4,/)
150   format(///,'****Pointers****',//,
     *      'Pcoord....',i4,/,
     *      'Pmatno....',i4,/,
     *      'Pnelempg..',i4,/,
     *      'Pmaxdof...',i4,/,
     *      'Piposek...',i4,/,
     *      'Plnods....',i4,/,
     *      'Pid.......',i4,/,
     *      'Pnumprop..',i4,/,
     *      'Pprops....',i4,/,
     *      'Pstfloc...',i4,/,
     *      'Peltrig...',i4,/,
     *      'Pmaxa.....',i4,/,
     *      'Pgstif....',i4,/,
```

```
     *         'Prhs.....',i4,/,
     *         'Pfea.....',i4,/,
     *         'Pdisnod.....',i4,/,
     *         'Pend.....',i4,/)
  200  format(/////,'**** Nodal Coordinates ****',//,
     *'Node No.      coord1          coord2          coord3'
     *   ,/)
  300  format(i5,3(4x,1pe12.5))
  320  format(/////,'**** Id Matrix***',//,
     *'Node No.    dof1      dof2      dof3',/)
  350  format(i5,3(5x,i6) )
  400  format(/////,'****Connectivity Matrix****',//,
     *         'Element   material  nod1  nod2',/)
  500  format(i5,6x,i5,3x,12i5)

c
      end

c*********************************************************
**********
      subroutine stiff
     *(lnods,coord,matno,props,stfloc,eltrig,nelempg,iposek)
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'

c
      dimension lnods(nelem,1) ,coord(npoin,1),matno(1),
     *         props(nmatrls,1 ), stfloc(1),iposek(1),
     *         eltrig(nelem,1),nelempg(ngrp,1)
c
                  read(input,*,ERR=99) elf
                  WRITE(*,*) 'CREATING LOCAL STIFFNESSES '


      do 5 igrp=1,ngrp
      call elematt(nelempg)
      do 10 ielem=elembgn,elemend
                  WRITE(*,*)  'CREATING ELEMENT #  -->',IELEM
                  nod1=lnods(ielem,1)
                  nod2=lnods(ielem,2)
c
c 2D elements truss, beam, grid
c
                  if(ncoor.eq.2)then
                   delx=coord(nod2,1)-coord(nod1,1)
                   dely=coord(nod2,2)-coord(nod1,2)
                   dlen=dsqrt(delx*delx+dely*dely)
                   eltrig(ielem,1)=delx/dlen
                   eltrig(ielem,2)=dely/dlen
                   elfke=dlen/2*elf
c 3D elements
                  else if(ncoor.eq.3)then
c
c spring, truss, elastic foundation 2n beam
c
                   if(istrtp.eq.1.or.istrtp.eq.5.or.istrtp.eq.9) then
                   delx=coord(nod2,1)-coord(nod1,1)
                   dely=coord(nod2,2)-coord(nod1,2)
                   delz=coord(nod2,3)-coord(nod1,3)
                   dlen=dsqrt(delx*delx+dely*dely+delz*delz)
                   eltrig(ielem,1)=delx/dlen
                   eltrig(ielem,2)=dely/dlen
                   eltrig(ielem,3)=delz/dlen
                   elfke=dlen/2*elf
c
c 3D 6n beam
c
                   elseif(istrtp.eq.6) then
                   nod1=lnods(ielem,2)
                   nod2=lnods(ielem,5)
                   delx=coord(nod2,1)-coord(nod1,1)
                   dely=coord(nod2,2)-coord(nod1,2)
                   delz=coord(nod2,3)-coord(nod1,3)
                   dlen=dsqrt(delx*delx+dely*dely+delz*delz)
                   dzx=dsqrt(delx*delx+delz*delz)

                   if(dzx.lt.0.001) then
                   eltrig(ielem,2)=0.0
                   else
                   eltrig(ielem,2)=asin(-delz/dzx)
                   if(delx.lt.0.0)
     *            eltrig(ielem,2) = acos(-1.0) - eltrig(ielem,2)
                   end if
                   cy=cos(-eltrig(ielem,2))
                   sy=sin(-eltrig(ielem,2))
                   delnx=cy*delx+sy*delz
                   delny=dely
                   delnz=-sy*delx+cy*delz


                   eltrig(ielem,3)=asin(delny/dlen)
                   if(delnx.lt.0.0)
     *            eltrig(ielem,3) = acos(-1.0) - eltrig(ielem,3)
                   cz=cos(-eltrig(ielem,3))
                   sz=sin(-eltrig(ielem,3))
c                  delmx=cz*delnx-sz*delny
                   delmy=sz*delnx+cz*delny
                   delmz=delnz

                  nod1=lnods(ielem,1)
                   nod2=lnods(ielem,2)
                   delx=coord(nod1,1)-coord(nod2,1)
                   dely=coord(nod1,2)-coord(nod2,2)
                   delz=coord(nod1,3)-coord(nod2,3)
                   dlen=dsqrt(delx*delx+dely*dely+delz*delz)

                   delnx=cy*delx+sy*delz
                   delny=dely
                   delnz=-sy*delx+cy*delz

c                  delmx=cz*delnx-sz*delny
                   delmy=sz*delnx+cz*delny
                   delmz=delnz

                   eltrig(ielem,1)=asin(delmz/dlen)
                   if(delmy.lt.0) eltrig(ielem,1)=acos(-1.0) -eltrig(ielem,1)

                  if( dabs(eltrig(ielem,1)).lt.0.00001) eltrig(ielem,1)=0.0
                  if( dabs(eltrig(ielem,2)).lt.0.00001) eltrig(ielem,2)=0.0
                  if( dabs(eltrig(ielem,3)).lt.0.00001) eltrig(ielem,3)=0.0

c
c 3D plate and brick (same rotation matrix)
c
                   elseif(istrtp.eq.7.or.istrtp.eq.8) then
                   nod1=lnods(ielem,1)
                   nod2=lnods(ielem,2)
                   delx=coord(nod2,1)-coord(nod1,1)
                   dely=coord(nod2,2)-coord(nod1,2)
                   delz=coord(nod2,3)-coord(nod1,3)
                   dlen=dsqrt(delx*delx+dely*dely+delz*delz)
                   dzx=dsqrt(delx*delx+delz*delz)

                   if(dzx.lt.0.00001) then
                   eltrig(ielem,2)=0.0
                   else
                   eltrig(ielem,2)=asin(-delz/dzx)
                   if(delx.lt.0.0)
     *            eltrig(ielem,2) = acos(-1.0) - eltrig(ielem,2)
                   end if
                   cy=cos(-eltrig(ielem,2))
                   sy=sin(-eltrig(ielem,2))
                   delnx=cy*delx+sy*delz
                   delny=dely
                   delnz=-sy*delx+cy*delz
c     write(*,*)'xy-plane x          y          z'
c                  write(*,*) delnx,delny,delnz

                   eltrig(ielem,3)=asin(delny/dlen)
                   if(delnx.lt.0.0)
     *            eltrig(ielem,3) = acos(-1.0) - eltrig(ielem,3)


                   cz=cos(-eltrig(ielem,3))
                   sz=sin(-eltrig(ielem,3))
                   delmx=cz*delnx-sz*delny
                   delmy=sz*delnx+cz*delny
                   delmz=delnz
c                  write(*,*)'x-axis  x          y          z'
c                  write(*,*) delmx,delmy,delmz
                   nod1=lnods(ielem,1)
                   nod2=lnods(ielem,4)
                   delx=coord(nod2,1)-coord(nod1,1)
                   dely=coord(nod2,2)-coord(nod1,2)
                   delz=coord(nod2,3)-coord(nod1,3)
                   dlen=dsqrt(delx*delx+dely*dely+delz*delz)

                   delnx=cy*delx+sy*delz
                   delny=dely
                   delnz=-sy*delx+cy*delz

                   delmx=cz*delnx-sz*delny
                   delmy=sz*delnx+cz*delny
                   delmz=delnz

c     write(*,*)'+/- y-axis  x          y          z'
c     write(*,*) delmx,delmy,delmz
```

```
                eltrig(ielem,1)=acos(delmy/dlen)
                if(delnz.lt.0) eltrig(ielem,1)=-eltrig(ielem,1)

               if( dabs(eltrig(ielem,1)).lt.0.00001) eltrig(ielem,1)=0.0
               if( dabs(eltrig(ielem,2)).lt.0.00001) eltrig(ielem,2)=0.0
               if( dabs(eltrig(ielem,3)).lt.0.00001) eltrig(ielem,3)=0.0
c     write(*,*) 'angles      Ax          Ay          Az'
c     write(*,*) (eltrig(ielem,j),j=1,3)

                end if
               end if

               imat=matno(ielem)
               istpos=iposek(ielem)
               if(istrtp.eq.1)then
                 call truss2d(dlen,imat,stfloc(istpos),props,istrtp)
               elseif(istrtp.eq.2)then
                 call truss2d(dlen,imat,stfloc(istpos),props,istrtp)
               elseif(istrtp.eq.3)then
                 call frame2d(dlen,imat,elfke,stfloc(istpos),props)
               elseif(istrtp.eq.4)then
                 call grid  (dlen,imat,stfloc(istpos),props)
               elseif(istrtp.eq.5)then
                 call truss2d(dlen,imat,stfloc(istpos),props,istrtp)
               elseif(istrtp.eq.6)then
                 nod1=lnods(ielem,2)
                 nod2=lnods(ielem,5)
                 delx=coord(nod2,1)-coord(nod1,1)
                 dely=coord(nod2,2)-coord(nod1,2)
                 delz=coord(nod2,3)-coord(nod1,3)
                 dlen=dsqrt(delx*delx+dely*dely+delz*delz)
                 call beam3d(dlen,imat,stfloc(istpos),props)
               elseif(istrtp.eq.7)then
                 call
plate3d(ielem,imat,stfloc(istpos),props,LNODS,COORD)
               elseif(istrtp.eq.8)then
                 call
brick3d(ielem,imat,stfloc(istpos),props,LNODS,COORD)
               elseif(istrtp.eq.9)then
                 call frame3d(dlen,imat,elfke,stfloc(istpos),props)

                end if
10    continue
5     continue
c
      return
99    write(output,*)
     *'***error occured while reading elastic foundation constant***'
      WRITE(*,*)' ERROR OCCURED READING ELASTIC
FOUNDATION CONSTANT
      STOP
      end
C*************************************************************
************
c     subroutine beam1 d (ielem,dlen,imat,stfloc,props)
c
c Stiffness of one dimensional beam element
c
c     implicit real*8(a-h,o-z)
c     include 'control.cmn'
c
c
c     dimension props(nmatrls,1),stfloc(nsizk,nsizk)

c     emod=props(imat,1)
c     dinert=props(imat,2)
c     const=emod*dinert/dlen
c     stfloc(1,1)=12*const/dlen**2
c     stfloc(1,2)= 6*const/dlen
c     stfloc(1,3)=-stfloc(1,1)
c     stfloc(1,4)= stfloc(1,2)
c     stfloc(2,2)= 4*const
c     stfloc(2,3)=-stfloc(1,2)
c     stfloc(2,4)= 2*const
c     stfloc(3,3)= stfloc(1,1)
c     stfloc(3,4)=-stfloc(1,2)
c     stfloc(4,4)= stfloc(2,2)
c
c copy lower half
c
c     do 5 l=2,nsizk
c      do 15 k=1,l-1
c       stfloc(l,k)=stfloc(k,l)
c15    continue
c5     continue
c
c     return
c     end
C*************************************************************
************
      subroutine truss2d(dlen,imat,stfloc,props,ik)
c
c Stiffness for a 2D truss
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension props(nmatrls,1),stfloc(nsizk,nsizk)
      if(ik.gt.1) then
       emod =props(imat,1)
       area =props(imat,2)
       const=emod*area/dlen
      else
       const=props(imat,1)/dlen
      end if
c
      stfloc(1,1)= const
      stfloc(1,2)=-const
      stfloc(2,1)=-const
      stfloc(2,2)= const
c
      return
      end
C*************************************************************
************
      subroutine frame2d(dlen,imat,elfke,stfloc,props)
c
c Stiffness for a 2D frame
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
c
      dimension props(nmatrls,1),stfloc(nsizk,nsizk)
c
      emod=props(imat,1)
      area=props(imat,2)
      dinert=props(imat,3)
      consta=emod*dinert/dlen
      constb=emod*area/dlen
c
      do 5 i=1,nsizk
       do 6 j=1,nsizk
                stfloc(i,j)=0.0
6      continue
5      continue
c
      stfloc(1,1)= constb
      stfloc(1,4)=-constb
      stfloc(2,2)=12*consta/dlen**2+elfke
      stfloc(2,3)= 6*consta/dlen
      stfloc(2,5)=-12*consta/dlen**2
      stfloc(2,6)= stfloc(2,3)
      stfloc(3,3)=4*consta
      stfloc(3,5)=-stfloc(2,3)
      stfloc(3,6)= 2*consta
      stfloc(4,4)= constb
      stfloc(5,5)= stfloc(2,2)
      stfloc(5,6)=-stfloc(2,3)
      stfloc(6,6)=4*consta
c
c copy lower half
c
      do 7 l=1,nsizk
              do 8 k=l+1,nsizk
               stfloc(k,l)=stfloc(l,k)
8      continue
7      continue
      return
      end
C*************************************************************
************
      subroutine grid (dlen,imat,stfloc,props)
c
c Stiffness for a grid element
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
c
      dimension props(nmatrls,1),stfloc(nsizk,nsizk)
c
      emod=props(imat,1)
      gmod=props(imat,2)
      dinert=props(imat,3)
      pinert=props(imat,4)
      consta=gmod*pinert/dlen
      constb=emod*dinert/dlen
c
```

```
      do 5 i=1,nsizk
       do 6 j=1,nsizk
                  stfloc(i,j)=0.0
 6     continue
 5     continue
 c
      stfloc(1,1)=consta
      stfloc(1,4)=-consta
      stfloc(2,2)=4*constb
      stfloc(2,3)=-6*constb/dlen
      stfloc(2,5)=2*constb
      stfloc(2,6)=-stfloc(2,3)
      stfloc(3,3)=12*constb/dlen**2
      stfloc(3,5)=stfloc(2,3)
      stfloc(3,6)=-stfloc(3,3)
      stfloc(4,4)=consta
      stfloc(5,5)=stfloc(2,2)
      stfloc(5,6)=-stfloc(2,3)
      stfloc(6,6)=stfloc(3,3)
 c
 c copy lower half
 c
      do 7 l=1,nsizk
                  do 8 k=l+1,nsizk
                    stfloc(k,l)=stfloc(l,k)
 8     continue
 7     continue
      return
      end
 c*************************************************************
 ************
      subroutine beam3d(dlen,imat,stfloc,props)
 c
 c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
 c
      dimension props(nmatrls,1),stfloc(36,36),ijk(4)
 c
                  ijk(1)=0
                  ijk(2)=6
                  ijk(3)=18
                  ijk(4)=24
 c
      emod=props(imat,1)
      gmod=props(imat,2)
      area=props(imat,3)
      xinert=props(imat,6)
      yinert=props(imat,7)
      zinert=props(imat,8)
 C
 C Cxx and Cyy USED FOR STRESSES later
 C Cxx used here for length of fictisous stiffners
 c
      do 5 i=1,nsizk
       do 5 j=1,nsizk
                  stfloc(i,j)=0.0d+00
 5     continue
 c
 c do fictisous members
 c
      dum=dlen
      dlen=props(imat,4)
      consta=1000.0*area*emod/dlen
      constb=1000.0*gmod*xinert/dlen
      constc=1000.0*emod*yinert/dlen
      constd=1000.0*emod*zinert/dlen

 c loop thru last two members
      do 15 i=3,4
 c index to proper part of stiffness matrix
                  j=ijk(i)
      stfloc(2+j,2+j)=stfloc(2+j,2+j)+consta
      stfloc(2+j,8+j)=stfloc(2+j,8+j)-consta
      stfloc(1+j,1+j)=stfloc(1+j,1+j)+12*constd/dlen**2
      stfloc(1+j,6+j)=stfloc(1+j,6+j)-6*constd/dlen
      stfloc(1+j,7+j)=stfloc(1+j,7+j)-12*constd/dlen**2
      stfloc(1+j,12+j)=stfloc(1+j,12+j)-6*constd/dlen
      stfloc(3+j,3+j)=stfloc(3+j,3+j)+12*constc/dlen**2
      stfloc(3+j,4+j)=stfloc(3+j,4+j)+6*constc/dlen
      stfloc(3+j,9+j)=stfloc(3+j,9+j)-12*constc/dlen**2
      stfloc(3+j,10+j)=stfloc(3+j,10+j)+6*constc/dlen
      stfloc(5+j,5+j)=stfloc(5+j,5+j)+constb
      stfloc(5+j,11+j)=stfloc(5+j,11+j)-constb
      stfloc(4+j,4+j)=stfloc(4+j,4+j)+4*constc
      stfloc(4+j,9+j)=stfloc(4+j,9+j)-6*constc/dlen
      stfloc(4+j,10+j)=stfloc(4+j,10+j)+2*constc
      stfloc(6+j,6+j)=stfloc(6+j,6+j)+4*constd
      stfloc(6+j,7+j)=stfloc(6+j,7+j)+6*constd/dlen
      stfloc(6+j,12+j)=stfloc(6+j,12+j)+2*constd
      stfloc(8+j,8+j)=stfloc(8+j,8+j)+consta
      stfloc(7+j,7+j)=stfloc(7+j,7+j)+12*constd/dlen**2
      stfloc(7+j,12+j)=stfloc(7+j,12+j)+6*constd/dlen
      stfloc(9+j,9+j)=stfloc(9+j,9+j)+12*constc/dlen**2
      stfloc(9+j,10+j)=stfloc(9+j,10+j)-6*constc/dlen
      stfloc(11+j,11+j)=stfloc(11+j,11+j)+constb
      stfloc(10+j,10+j)=stfloc(10+j,10+j)+4*constc
      stfloc(12+j,12+j)=stfloc(12+j,12+j)+4*constd
 15    continue

 c loop thru first two members
 c stiffness already in vertical position
 c
      do 10 i=1,2
 c index to proper part of stiffness matrix
                  j=ijk(i)
      stfloc(2+j,2+j)=stfloc(2+j,2+j)+consta
      stfloc(2+j,8+j)=stfloc(2+j,8+j)-consta
      stfloc(1+j,1+j)=stfloc(1+j,1+j)+12*constd/dlen**2
      stfloc(1+j,6+j)=stfloc(1+j,6+j)+6*constd/dlen
      stfloc(1+j,7+j)=stfloc(1+j,7+j)-12*constd/dlen**2
      stfloc(1+j,12+j)=stfloc(1+j,12+j)+6*constd/dlen
      stfloc(3+j,3+j)=stfloc(3+j,3+j)+12*constc/dlen**2
      stfloc(3+j,4+j)=stfloc(3+j,4+j)-6*constc/dlen
      stfloc(3+j,9+j)=stfloc(3+j,9+j)-12*constc/dlen**2
      stfloc(3+j,10+j)=stfloc(3+j,10+j)-6*constc/dlen
      stfloc(5+j,5+j)=stfloc(5+j,5+j)+constb
      stfloc(5+j,11+j)=stfloc(5+j,11+j)-constb
      stfloc(4+j,4+j)=stfloc(4+j,4+j)+4*constc
      stfloc(4+j,9+j)=stfloc(4+j,9+j)+6*constc/dlen
      stfloc(4+j,10+j)=stfloc(4+j,10+j)+2*constc
      stfloc(6+j,6+j)=stfloc(6+j,6+j)+4*constd
      stfloc(6+j,7+j)=stfloc(6+j,7+j)-6*constd/dlen
      stfloc(6+j,12+j)=stfloc(6+j,12+j)+2*constd
      stfloc(8+j,8+j)=stfloc(8+j,8+j)+consta
      stfloc(7+j,7+j)=stfloc(7+j,7+j)+12*constd/dlen**2
      stfloc(7+j,12+j)=stfloc(7+j,12+j)-6*constd/dlen
      stfloc(9+j,9+j)=stfloc(9+j,9+j)+12*constc/dlen**2
      stfloc(9+j,10+j)=stfloc(9+j,10+j)+6*constc/dlen
      stfloc(11+j,11+j)=stfloc(11+j,11+j)+constb
      stfloc(10+j,10+j)=stfloc(10+j,10+j)+4*constc
      stfloc(12+j,12+j)=stfloc(12+j,12+j)+4*constd
 10    continue
 c
 c do main beam
 c
      dlen=dum
      consta=area*emod/dlen
      constb=gmod*xinert/dlen
      constc=emod*yinert/dlen
      constd=emod*zinert/dlen

      stfloc(7,7)=stfloc(7,7)+consta
      stfloc(7,25)=stfloc(7,25)-consta
      stfloc(8,8)=stfloc(8,8)+12*constd/dlen**2
      stfloc(8,12)=stfloc(8,12)+6*constd/dlen
      stfloc(8,26)=stfloc(8,26)-12*constd/dlen**2
      stfloc(8,30)=stfloc(8,30)+6*constd/dlen
      stfloc(9,9)=stfloc(9,9)+12*constc/dlen**2
      stfloc(9,11)=stfloc(9,11)-6*constc/dlen
      stfloc(9,27)=stfloc(9,27)-12*constc/dlen**2
      stfloc(9,29)=stfloc(9,29)-6*constc/dlen
      stfloc(10,10)=stfloc(10,10)+constb
      stfloc(10,28)=stfloc(10,28)-constb
      stfloc(11,11)=stfloc(11,11)+4*constc
      stfloc(11,27)=stfloc(11,27)+6*constc/dlen
      stfloc(11,29)=stfloc(11,29)+2*constc
      stfloc(12,12)=stfloc(12,12)+4*constd
      stfloc(12,26)=stfloc(12,26)-6*constd/dlen
      stfloc(12,30)=stfloc(12,30)+2*constd
      stfloc(25,25)=stfloc(25,25)+consta
      stfloc(26,26)=stfloc(26,26)+12*constd/dlen**2
      stfloc(26,30)=stfloc(26,30)-6*constd/dlen
      stfloc(27,27)=stfloc(27,27)+12*constc/dlen**2
      stfloc(27,29)=stfloc(27,29)+6*constc/dlen
      stfloc(28,28)=stfloc(28,28)+constb
      stfloc(29,29)=stfloc(29,29)+4*constc
      stfloc(30,30)=stfloc(30,30)+4*constd
 c
 c copy lower half
 c
      do 7 l=1,nsizk
                  do 8 k=1,nsizk
                    stfloc(k,l)=stfloc(l,k)
 8     continue
 7     continue
 c
      return
      end
```

```
c*******************************************************
************
      subroutine partostiff(dlen,imat,stfloc,props)
c
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension props(nmatrls,1),stfloc(36,36)
c
      emod=props(imat,1)
      gmod=props(imat,2)
      area=props(imat,3)
      xinert=props(imat,6)
      yinert=props(imat,7)
      zinert=props(imat,8)
c
      do 5 i=1,nsizk
       do 5 j=1,nsizk
                stfloc(i,j)=0.0d+00
5     continue
c
c do main beam
c
      consta=area*emod/dlen
      constb=gmod*xinert/dlen
      constc=emod*yinert/dlen
      constd=emod*zinert/dlen

      stfloc(7,7)=consta
      stfloc(7,25)=-consta
      stfloc(8,8)=12*constd/dlen**2
      stfloc(8,12)=6*constd/dlen
      stfloc(8,26)=-12*constd/dlen**2
      stfloc(8,30)=6*constd/dlen
      stfloc(9,9)=12*constc/dlen**2
      stfloc(9,11)=-6*constc/dlen
      stfloc(9,27)=-12*constc/dlen**2
      stfloc(9,29)=-6*constc/dlen
      stfloc(10,10)=constb
      stfloc(10,28)=-constb
      stfloc(11,11)=4*constc
      stfloc(11,27)=6*constc/dlen
      stfloc(11,29)=2*constc
      stfloc(12,12)=4*constd
      stfloc(12,26)=-6*constd/dlen
      stfloc(12,30)=2*constd
      stfloc(25,25)=consta
      stfloc(26,26)=12*constd/dlen**2
      stfloc(26,30)=-6*constd/dlen
      stfloc(27,27)=12*constc/dlen**2
      stfloc(27,29)=6*constc/dlen
      stfloc(28,28)=constb
      stfloc(29,29)=4*constc
      stfloc(30,30)=4*constd
c
c copy lower half
c
      do 7 l=1,nsizk
                do 8 k=l,nsizk
                 stfloc(k,l)=stfloc(l,k)
8     continue
7     continue
c
      return
      end

c*******************************************************
************
      subroutine frame3d(dlen,imat,elfke,stfloc,props)
c
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension props(nmatrls,1),stfloc(nsizk,nsizk)
c
      emod=props(imat,1)
      gmod=props(imat,2)
      area=props(imat,3)
      xinert=props(imat,4)
      yinert=props(imat,5)
      zinert=props(imat,6)
c
      consta=area*emod/dlen
      constb=gmod*xinert/dlen
      constc=emod*yinert/dlen
      constd=emod*zinert/dlen
c
      do 5 i=1,nsizk
       do 6 j=1,nsizk
                stfloc(i,j)=0.0
6     continue
5     continue
c
      stfloc(1,1)=consta
      stfloc(1,7)=-consta
      stfloc(2,2)=12*constd/dlen**2
      stfloc(2,6)=6*constd/dlen
      stfloc(2,8)=-stfloc(2,2)
      stfloc(2,12)=stfloc(2,6)
      stfloc(3,3)=12*constc/dlen**2+elfke
      stfloc(3,5)=-6*constc/dlen
      stfloc(3,9)=-12*constc/dlen**2
      stfloc(3,11)=stfloc(3,5)
      stfloc(4,4)=constb
      stfloc(4,10)=-constb
      stfloc(5,5)=4*constc
      stfloc(5,9)=-stfloc(3,5)
      stfloc(5,11)=2*constc
      stfloc(6,6)=4*constd
      stfloc(6,8)=-stfloc(2,6)
      stfloc(6,12)=2*constd
      stfloc(7,7)=consta
      stfloc(8,8)=stfloc(2,2)
      stfloc(8,12)=-stfloc(2,6)
      stfloc(9,9)=stfloc(3,3)
      stfloc(9,11)=-stfloc(3,5)
      stfloc(10,10)=constb
      stfloc(11,11)=stfloc(5,5)
      stfloc(12,12)=stfloc(6,6)
c
c copy lower half
c
      do 7 l=1,nsizk
                do 8 k=l+1,nsizk
                 stfloc(k,l)=stfloc(l,k)
8     continue
7     continue
      return
      end
c*******************************************************
*****
      subroutine plate3d(ielem,imat,stfloc,props,LNODS,COORD)

c
c subroutine for 3d plate including bending and shear web
c stiffness. 5 dof per node x & y from shear web
c                z Rx Ry from bending plate
c       6th dof is 1/1000 of minimum k
c
                IMPLICIT REAL*8(A-H,O-Z)
                INCLUDE 'control.cmn'
                DIMENSION PROPS(NMATRLS,1),stfloc(24,24),
     &
LNODS(NELEM,1),COORD(NPOIN,1),BHOLD(12,12),SHOLD(12,12)
     &  ,rotinv(24,24),rotate(24,24),hhold(12,24)

c
c call bending part of stiffness
c
                call bplate(ielem,imat,BHOLD,props,LNODS,COORD)
c
c call shear part of stiffness
c
                call splate(ielem,imat,SHOLD,props,LNODS,COORD)

c
c combine stiffnesses together
c
                do 9 i=1,24
                do 9 j=1,24
                rotate(i,j)=0.0
                rotinv(i,j)=0.0
9     stfloc(i,j)=0.0
c
c relocate bending to total local
c
                rotate(1,3)=1.0
                rotate(2,5)=1.0
                rotate(3,4)=-1.0
                rotate(4,9)=1.0
                rotate(5,11)=1.0
                rotate(6,10)=-1.0
                rotate(7,15)=1.0
                rotate(8,17)=1.0
                rotate(9,16)=-1.0
                rotate(10,21)=1.0
```

```
                rotate(11,23)=1.0
                rotate(12,22)=-1.0
                 call mult (bhold,rotate,hhold,12,12,24,24,12,12,24)
                 call trans(rotate,rotinv,24,24,12,24)
                 call mult (rotinv,hhold,stfloc,24,24,12,24,24,12,24)
c
c relocate stretching to total local
c
                do 5 i=1,24
                do 5 j=1,24
5     rotate(i,j)=0.0

                rotate(1,1)=1.0
                rotate(2,2)=1.0
                rotate(3,7)=1.0
                rotate(4,8)=1.0
                rotate(5,13)=1.0
                rotate(6,14)=1.0
                rotate(7,19)=1.0
                rotate(8,20)=1.0
                 call mult (shold,rotate,hhold,12,12,24,24,12,12,24)
                 call trans(rotate,rotinv,24,24,12,24)
                 call mult (rotinv,hhold,rotate,24,24,12,24,24,12,24)
                 call addmat(stfloc,rotate,24,24,24,24,0)
c
c     FIND MINIMUM DIAGONAL STIFFNESS
c
                smin=stfloc(1,1)
                do 10 i=2,23
                if(i.eq.6.or.i.eq.12.or.i.eq.18) goto 10
                if(stfloc(i,i).lt.smin) smin=stfloc(i,i)
10     continue
c
c set 6th dofs to 1/1000 of min dof
c
                stfloc(6,6)=smin/1000.0
                stfloc(12,12)=smin/1000.0
                stfloc(18,18)=smin/1000.0
                stfloc(24,24)=smin/1000.0
                return
                end
c
C*******************************************************
***********
    SUBROUTINE
BRICK3D(IELEM,IMAT,STFLOC,PROPS,LNODS,COORD)
    IMPLICIT REAL*8(A-H,O-Z)
    include 'control.cmn'
    include 'ios.cmn'
C
C   *** THREE DIMENSIONAL SOLID ELEMENT ***
C
    DIMENSION D(1),STFLOC(NSIZK,NSIZK),props(nmatrls,1),
   &        lnods(nelem,1),coord(npoin,1),
   &        SHP(4,8),RG(8),SG(8),TG(8),WG(8)
C
    EMOD=PROPS(IMAT,1)
    AMU=PROPS(IMAT,2)
C
    D(1)=EMOD*(1.-AMU)/(1.+AMU)/(1.-2.*AMU)
    D(2)=AMU*D(1)/(1.-AMU)
    D(3)=EMOD/2.0/(1.+AMU)
C
    CALL PGAUS3(RG,SG,TG,WG)
C.....COMPUTE STIFFNESS,COMPUTE INTEGRALS OF SHAPE
FUNCTIONS
    DO 320 L=1,8
    CALL SHP3D(RG(L),SG(L),TG(L),COORD,SHP,XSJ,ielem,lnods)
    DO 100 I =1,8
    I3=3*I
    I1=I3-2
    I2=I3-1
    DO 100 J =1,8
    J3=3*J
    J1=J3-2
    J2=J3-1
    STFLOC(I1,J1) = STFLOC(I1,J1)+(D(1)*SHP(1,I)*SHP(1,J)
   &+D(3)*(SHP(2,I)*SHP(2,J)+SHP(3,I)*SHP(3,J)))*XSJ
    STFLOC(I1,J2)=STFLOC(I1,J2)+(D(2)*SHP(1,I)*SHP(2,J)
   &+D(3)*SHP(2,I)*SHP(1,J))*XSJ
    STFLOC(I1,J3)=STFLOC(I1,J3)+(D(2)*SHP(1,I)*SHP(3,J)
   &+D(3)*SHP(3,I)*SHP(1,J))*XSJ
    STFLOC(I2,J1)=STFLOC(I2,J1)+(D(2)*SHP(2,I)*SHP(1,J)
   &+D(3)*SHP(1,I)*SHP(2,J))*XSJ
    STFLOC(I2,J2) = STFLOC(I2,J2)+(D(1)*SHP(2,I)*SHP(2,J)
   &+D(3)*(SHP(1,I)*SHP(1,J)+SHP(3,I)*SHP(3,J)))*XSJ
    STFLOC(I2,J3)=STFLOC(I2,J3)+(D(2)*SHP(2,I)*SHP(3,J)
   &+D(3)*SHP(3,I)*SHP(2,J))*XSJ
    STFLOC(I3,J1)=STFLOC(I3,J1)+(D(2)*SHP(3,I)*SHP(1,J)
   &+D(3)*SHP(1,I)*SHP(3,J))*XSJ
    STFLOC(I3,J2)=STFLOC(I3,J2)+(D(2)*SHP(3,I)*SHP(2,J)
   &+D(3)*SHP(2,I)*SHP(3,J))*XSJ
100  STFLOC(I3,J3) = STFLOC(I3,J3)+(D(1)*SHP(3,I)*SHP(3,J)
   &+D(3)*(SHP(1,I)*SHP(1,J)+SHP(2,I)*SHP(2,J)))*XSJ
320  CONTINUE
C.....FORM LOWER PART BY SYMMETRY
    DO 160 I = 1,24
    DO 160 J = I,24
160  STFLOC(J,I) = STFLOC(I,J)
    write(output,*)'local stiffness matrix for element',ielem
    do 10 i=1,24
10   write(output,99) (stfloc(i,j),j=1,24)
99   format(24e10.2)

    RETURN
    END
C*******************************************************
***********
    SUBROUTINE PGAUS3(R,S,T,W)
    IMPLICIT REAL*8(A-H,O-Z)
C
    DIMENSION LR(8),LS(8),LT(8),R(8),S(8),T(8),W(8)
    DATA LR/-1,1,1,-1,-1,1,1,-1/
    DATA LS/-1,-1,1,1,-1,-1,1,1/
    DATA LT/1,1,1,1,-1,-1,-1,-1/
C.....2 X 2 X 2 INTEGRATION
    G=1.0D0/DSQRT(3.0D0)
    DO 21 I=1,8
    R(I)=G*LR(I)
    S(I)=G*LS(I)
    T(I)=G*LT(I)
 21 W(I)=1.0
    RETURN
    END
C*******************************************************
************
    SUBROUTINE SHP3D(RR,SS,TT,COORD,SHP,XSJ,ielem,lnods)
    IMPLICIT REAL*8(A-H,O-Z)
    INCLUDE 'control.cmn'
C
C   CALL SHP3D(RG(L),SG(L),TG(L),COORD,SHP,XSJ,ielem,lnods)
C.....SUBROUTINE FOR SHAPE FUNCTION OF 3-SOLIDS WITH 8
NODES
C
    DIMENSION
SHP(4,8),COORD(NPOIN,1),R(8),S(8),T(8),SX(3,3),XS(3,3),
   &        lnods(nelem,1)
    DATA R/-1.0,1.0,1.,-1.,-1.0,1.0,1.,-1./
    DATA S/-1.0,-1.,1.,1.0,-1.0,-1.,1.,1.0/
    DATA T/1.0,1.0,1.0,1.0,-1.,-1.,-1.,-1./
    DO 100 I =1,8
    SHP(4,I) =0.125*(1.0+RR*R(I))*(1.0+SS*S(I))*(1.0+TT*T(I))
    SHP(1,I) =0.125*R(I)*(1.0+SS*S(I))*(1.0+TT*T(I))
    SHP(2,I) =0.125*(1.0+RR*R(I))*S(I)*(1.0+TT*T(I))
100  SHP(3,I) =0.125*(1.0+RR*R(I))*(1.0+SS*S(I))*T(I)
C
C.....CONTSTRUCTION OF JACOBIAN AND INVERSE
C
    DO 130 I = 1,3
    DO 130 J =1,3
    XS(I,J)  = 0.0
    DO 130 K=1,8
130  XS(I,J) = XS(I,J) + COORD(LNODS(IELEM,K),J) * SHP(I,K)

    XSJ = XS(1,1)*(XS(2,2)*XS(3,3)-XS(2,3)*XS(3,2))-XS(1,2)*(XS(2,1)
   &    *XS(3,3)-XS(2,3)*XS(3,1))+XS(1,3)*(XS(2,1)*XS(3,2)-XS(2,2)*
   &    XS(3,1))

    SX(1,1) = (XS(2,2)*XS(3,3)-XS(2,3)*XS(3,2))/XSJ
    SX(2,1) =-(XS(2,1)*XS(3,3)-XS(2,3)*XS(3,1))/XSJ
    SX(3,1) = (XS(2,1)*XS(3,2)-XS(2,2)*XS(3,1))/XSJ
    SX(1,2) =-(XS(1,2)*XS(3,3)-XS(1,3)*XS(3,2))/XSJ
    SX(2,2) = (XS(1,1)*XS(3,3)-XS(1,3)*XS(3,1))/XSJ
    SX(3,2) =-(XS(1,1)*XS(3,2)-XS(1,2)*XS(3,1))/XSJ
    SX(1,3) = (XS(1,2)*XS(2,3)-XS(1,3)*XS(2,2))/XSJ
    SX(2,3) =-(XS(1,1)*XS(2,3)-XS(1,3)*XS(2,1))/XSJ
    SX(3,3) = (XS(1,1)*XS(2,2)-XS(1,2)*XS(2,1))/XSJ

C
C.....FORM GLOBAL DERIVATIVES
C
    DO 140 I=1,8
    TP1 = SHP(1,I)*SX(1,1)+SHP(2,I)*SX(2,1)+SHP(3,I)*SX(3,1)
    TP2 = SHP(1,I)*SX(1,2)+SHP(2,I)*SX(2,2)+SHP(3,I)*SX(3,2)
    SHP(3,I) = SHP(1,I)*SX(1,3)+SHP(2,I)*SX(2,3)+SHP(3,I)*SX(3,3)
    SHP(1,I) = TP1
140  SHP(2,I) = TP2

    RETURN
    END
```

```
c*************************************************************
******
      subroutine rota2(ielem,eltrig,rotate)
c
c rotation matrix for a 2D truss
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension eltrig(nelem,1),ROTATE(36,36)
c
      do 5 i=1,36
                  do 6 j=1,36
                   rotate(i,j)=0.0
6     continue
5     continue
c
      rotate(1,1)=eltrig(ielem,1)
      rotate(1,2)=eltrig(ielem,2)
      rotate(2,3)=rotate(1,1)
      rotate(2,4)=rotate(1,2)
c
c     write(output,*)'--Rotation Matrix--'
c     write(output,10)((rotate(i,j),j=1,4),i=1,2)
10    format(4(2x,1pe12.5),/)
      return
      end
c*************************************************************
*******
      subroutine rota3(ielem,eltrig,rotate)
c
c rotation matrix for a 2D frame or grid
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
c
      dimension eltrig(nelem,1),ROTATE(36,36)
c
      do 5 i=1,36
                  do 6 j=1,36
                   rotate(i,j)=0.0
6     continue
5     continue
c
      rotate(1,1)= eltrig(ielem,1)
      rotate(1,2)= eltrig(ielem,2)
      rotate(2,1)=-rotate(1,2)
      rotate(2,2)= rotate(1,1)
      rotate(3,3)= 1.0
c
      do 7 i=1,3
                  do 8 j=1,3
                    rotate(i+3,j+3)=rotate(i,j)
8     continue
7     continue
c
      return
      end
c*************************************************************
*******
      subroutine rota5(ielem,eltrig,rotate)
c
c rotation matrix for a 3D truss or spring
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
c
      dimension eltrig(nelem,1),ROTATE(36,36)
c
      do 5 i=1,36
                 do 6 j=1,36
                  rotate(i,j)=0.0
6     continue
5     continue
c
      rotate(1,1)=eltrig(ielem,1)
      rotate(1,2)=eltrig(ielem,2)
      rotate(1,3)=eltrig(ielem,3)
      rotate(2,4)=rotate(1,1)
      rotate(2,5)=rotate(1,2)
      rotate(2,6)=rotate(1,3)
c
      return
      end
c*************************************************************
**********
      subroutine rota6(ielem,eltrig,rotate)
c
c rotation matrix for a 3D frame
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension eltrig(nelem,1),ROTATE(36,36)
c
      do 5 i=1,36
                  do 6 j=1,36
                   rotate(i,j)=0.0
6     continue
5     continue
c
      rotate(1,1)=eltrig(ielem,1)
      rotate(1,2)=eltrig(ielem,2)
      rotate(1,3)=eltrig(ielem,3)
c
c check if the member is vertical
c
      if(rotate(1,1).eq.0.and.rotate(1,3).eq.0) then
                  rotate(2,1)=-rotate(1,2)
                  rotate(3,3)=1.0
      else
                  cxz=dsqrt(rotate(1,1)**2 + rotate(1,3)**2)
                  rotate(2,1)=-rotate(1,1)*rotate(1,2)/cxz
                  rotate(2,2)=cxz
                  rotate(2,3)=-rotate(1,2)*rotate(1,3)/cxz
                  rotate(3,1)=-rotate(1,3)/cxz
                  rotate(3,3)=rotate(1,1)/cxz
      end if
c
                  write(output,*)' 3d frame rotation matrix '
      do 7 i=1,3
       do 8 j=1,3
                 rotate(i+3,j+3)=rotate(i,j)
                 rotate(i+6,j+6)=rotate(i,j)
                 rotate(i+9,j+9)=rotate(i,j)
8     continue
7     continue
c
      return
      end
c*************************************************************
*******
      subroutine rota7(ielem,eltrig,rotate)
c
c rotation matrix for a grid plate
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
c
      dimension eltrig(nelem,1),ROTATE(36,36)
c
      do 5 i=1,36
                 do 6 j=1,36
                  rotate(i,j)=0.0
6     continue
5     continue
c
      rotate(2,2)= eltrig(ielem,1)
      rotate(2,3)= eltrig(ielem,2)
      rotate(3,2)= -1.0 * rotate(2,3)
      rotate(3,3)= rotate(2,2)
      rotate(1,1)= 1.0
c
      do 7 i=1,3
                 do 8 j=1,3
                   rotate(i+3,j+3)=rotate(i,j)
                   rotate(i+6,j+6)=rotate(i,j)
                   rotate(i+9,j+9)=rotate(i,j)
8     continue
7     continue
c
      return
      end
c*************************************************************
**********
      subroutine rota8(ielem,eltrig,rotate)
c
c rotation matrix for a 3D 6node beam
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension eltrig(nelem,1),ROTATE(36,36)
c
      do 5 i=1,36
                 do 6 j=1,36
```

```
                rotate(i,j)=0.0
6     continue
5     continue
c
                cx=cos(eltrig(ielem,1))
                cy=cos(eltrig(ielem,2))
                cz=cos(eltrig(ielem,3))
                sx=sin(eltrig(ielem,1))
                sy=sin(eltrig(ielem,2))
                sz=sin(eltrig(ielem,3))
c
                rotate(1,1)= cy*cz
                rotate(1,2)= sz
                rotate(1,3)= -cz*sy
                rotate(2,1)= -sz*cx*cy + sx*sy
                rotate(2,2)= cz*cx
                rotate(2,3)= sz*cx*sy + sx*cy
                rotate(3,1)= sz*sx*cy + sy*cx
                rotate(3,2)= -sx*cz
                rotate(3,3)= -sy*sz*sx + cx*cy
                write(2,*) ' GLOBAL ROTATION MATRIX '
                WRITE(2,99) ROTATE(1,1),ROTATE(1,2),ROTATE(1,3)
                WRITE(2,99) ROTATE(2,1),ROTATE(2,2),ROTATE(2,3)
                WRITE(2,99) ROTATE(3,1),ROTATE(3,2),ROTATE(3,3)
c
     do 7 i=1,3
                 do 8 j=1,3
                   rotate(i+3,j+3)=rotate(i,j)
                   rotate(i+6,j+6)=rotate(i,j)
                   rotate(i+9,j+9)=rotate(i,j)
                   rotate(i+12,j+12)=rotate(i,j)
                   rotate(i+15,j+15)=rotate(i,j)
                   rotate(i+18,j+18)=rotate(i,j)
                   rotate(i+21,j+21)=rotate(i,j)
                   rotate(i+24,j+24)=rotate(i,j)
                   rotate(i+27,j+27)=rotate(i,j)
                   rotate(i+30,j+30)=rotate(i,j)
                   rotate(i+33,j+33)=rotate(i,j)
8     continue
7     continue
c
     return
99   FORMAT(3F10.4)
     end
c*******************************************************
*****
     subroutine rota10(ielem,eltrig,rotate)
c
c local to global rotation matrix for a 3D plate or brick
c
     implicit real*8(a-h,o-z)
     include 'control.cmn'
c
     dimension eltrig(nelem,1),ROTATE(36,36)
c
     do 5 i=1,36
                 do 6 j=1,36
                  rotate(i,j)=0.0
6     continue
5     continue
c
                cx=cos(eltrig(ielem,1))
                cy=cos(eltrig(ielem,2))
                cz=cos(eltrig(ielem,3))
                sx=sin(eltrig(ielem,1))
                sy=sin(eltrig(ielem,2))
                sz=sin(eltrig(ielem,3))
c    plate
c      rotate(1,1)= cy*cz
c      rotate(1,2)= -sz
c      rotate(1,3)= cz*sy
c      rotate(2,1)= sz*cx*cy + sx*sy
c      rotate(2,2)= cz*cx
c      rotate(2,3)= sz*cx*sy - sx*cy
c      rotate(3,1)= sz*sx*cy - sy*cx
c      rotate(3,2)= sx*cz
c      rotate(3,3)= sy*sz*sx + cx*cy
c
                rotate(1,1)= cy*cz
                rotate(1,2)= sz
                rotate(1,3)= -cz*sy
                rotate(2,1)= -sz*cx*cy + sx*sy
                rotate(2,2)= cz*cx
                rotate(2,3)= sz*cx*sy + sx*cy
                rotate(3,1)= sz*sx*cy + sy*cx
                rotate(3,2)= -sx*cz
                rotate(3,3)= -sy*sz*sx + cx*cy

                write(2,*) ' GLOBAL ROTATION MATRIX '
                WRITE(2,99) ROTATE(1,1),ROTATE(1,2),ROTATE(1,3)
                WRITE(2,99) ROTATE(2,1),ROTATE(2,2),ROTATE(2,3)
                WRITE(2,99) ROTATE(3,1),ROTATE(3,2),ROTATE(3,3)

c
     do 7 i=1,3
                 do 8 j=1,3
                   rotate(i+3,j+3)=rotate(i,j)
                   rotate(i+6,j+6)=rotate(i,j)
                   rotate(i+9,j+9)=rotate(i,j)
                   rotate(i+12,j+12)=rotate(i,j)
                   rotate(i+15,j+15)=rotate(i,j)
                   rotate(i+18,j+18)=rotate(i,j)
                   rotate(i+21,j+21)=rotate(i,j)
8     continue
7     continue
c
     return
99   FORMAT(3F10.4)
     end

c*******************************************************
***********
     subroutine assemb
    *  (lnods,id,stfloc,eltrig,maxa,stifgl,nelempg,iposek,maxdof)
c
c subroutine to assemble the global stiffness matrix
c
     implicit real*8(a-h,o-z)
     include 'control.cmn'
     include 'ios.cmn'

c
     dimension ROTATE(36,36),dclstf(36,36),rotinv(36,36),dpstmt(36,36),
    *       stfelg(36,36),lm(36),maxa(1),lnods(nelem,1),
    *       id(npoin,1),stfloc(1),iposek(1),eltrig(nelem,1),
    *       nelempg(ngrp,1),maxdof(1),stifgl(1)
c
                WRITE(*,*)'ASSEMBLING GLOBAL STIFFNESS '

     do 2 isizg=1,nsizg
                 stifgl(isizg)=0.0
2     continue
c
     do 3 igrp=1,ngrp
      call elematt(nelempg)
      do 5 ielem=elembgn,elemend
      WRITE(*,*)'ASSEMBLING ELEMENT # -->',IELEM
                idof=0
                do 7 inode=1,nnode
                 node =lnods(ielem,inode)
                 do 9 iid=1,maxdof(node)
                  idof=idof+1
                  lm(idof)=id(node,iposid(iid))
9     continue
7     continue
c.....
                istpos=iposek(ielem)
c.....
c 3d spring
                 if(istrtp.eq.1)then
                  call rota5(ielem,eltrig,rotate)
c 2d truss
                 elseif(istrtp.eq.2)then
                  call rota2(ielem,eltrig,rotate)
c 2d beam
                 elseif(istrtp.eq.3)then
                  call rota3(ielem,eltrig,rotate)
c grid
                 elseif(istrtp.eq.4)then
                  call rota3(ielem,eltrig,rotate)
c 3d truss
                 elseif(istrtp.eq.5)then
                  call rota5(ielem,eltrig,rotate)
c 3d 6n beam
                 elseif(istrtp.eq.6)then
                  call rota8(ielem,eltrig,rotate)
c 3d plate or brick
                 elseif(istrtp.eq.7.or.istrtp.eq.8)then
                  call rota10(ielem,eltrig,rotate)
c 3d 2n beam
                 elseif(istrtp.eq.9)then
                  call rota6(ielem,eltrig,rotate)
                 end if
c
                 call getastif(dclstf,stfloc(istpos),nsizk)
c
                 nkrot=nid*nnode
c
c rotate to global
c
```

```
c Kg = Qt * Kl * Q

                call mult
   *  (dclstf,rotate,dpstmt,36,36,36,36,nsizk,nsizk,nkrot)
                call trans(rotate,rotinv,36,36,nsizk,nkrot)
                call mult
   *  (rotinv,dpstmt,stfelg,36,36,36,36,nkrot,nsizk,nkrot)
c
               do 12 irow=1,nkrot
                idlin=lm(irow)
                if(idlin.ne.0)then
                 do 14 ikol=1,nkrot
                  idkol=lm(ikol)
                  if(idkol.ne.0.and.idlin.le.idkol)then
                   iadres=maxa(idkol)+idkol-idlin
                   stifgl(iadres)=stifgl(iadres)+stfelg(irow,ikol)
                  end if
14      continue
               end if
12    continue
c
5     continue
3     continue
               write(2,*)

      return
      end
c*************************************************************
**
      subroutine getastif(stfelg1,stfelg2,nsizk)
      implicit real*8(a-h,o-z)
      include 'ios.cmn'
c
c YA! GET A STIF!!
c
      dimension stfelg1(36,36),stfelg2(nsizk,nsizk)
c
      do 4 isizk=1,nsizk
       do 6 jsizk=1,nsizk
               stfelg1(isizk,jsizk)=stfelg2(isizk,jsizk)
6     continue
4     continue
c
      return
      end
C*************************************************************
*****
               SUBROUTINE
BPLATE(IELEM,IMAT,stfloc,PROPS,LNODS,COORD)
C
C STIFFNESS FOR PLATE BENDING
C

               IMPLICIT REAL*8(A-H,O-Z)
               INCLUDE 'control.cmn'
               DIMENSION PROPS(NMATRLS,1),stfloc(12,12),
     &  LNODS(NELEM,1),COORD(NPOIN,1),GAUSS(2,2),WT(2,2),
     &  SF(4),GDSF(2,4),TRICK(4,2)

C
C THIS PLATE BENDING ELEMENT IS DERIVED USING GAUSS
QUADRATURE
C INTEGRATION FOR BENDING TERMS AND REDUCED G.Q.I.
FOR THE SHEAR TERMS
C
C DATA FOR GAUSS
C
               WT(1,1)=2.0D+0
               WT(1,2)=1.0D+0
               WT(2,2)=1.0D+0
               GAUSS(1,1)=0.0D+0
               GAUSS(1,2)=-.57735027D+0
               GAUSS(2,2)=.57735027D+0
C     WRITE(*,5) WT(1,1),WT(1,2),WT(2,1),WT(2,2)
C     WRITE(*,5)GAUSS(1,1),GAUSS(1,2),GAUSS(2,1),GAUSS(2,2)
5     FORMAT (4F15.3)
C
C SETUP 2D PLANE COORDINATES TRICK( N# , X,Y )
C
               TRICK(1,1)=0.
               TRICK(1,2)=0.
               H= (COORD(LNODS(IELEM,1),1)-
COORD(LNODS(IELEM,4),1))**2
   &    +(COORD(LNODS(IELEM,1),2)-
COORD(LNODS(IELEM,4),2))**2
   &    +(COORD(LNODS(IELEM,1),3)-
COORD(LNODS(IELEM,4),3))**2
               H=H**0.5D+00
               W= (COORD(LNODS(IELEM,1),1)-
COORD(LNODS(IELEM,2),1))**2
   &    +(COORD(LNODS(IELEM,1),2)-
COORD(LNODS(IELEM,2),2))**2
   &    +(COORD(LNODS(IELEM,1),3)-
COORD(LNODS(IELEM,2),3))**2
               W=W**0.5D+00
               TRICK(2,1)=W
               TRICK(2,2)=0.
               TRICK(3,1)=W
               TRICK(3,2)=H
               TRICK(4,1)=0.
               TRICK(4,2)=H
               write(*,*)' width=',w,'height=',h
C
C INITIALIZE STFLOC MATRIX
C
               DO 2 I=1,12
               DO 2 J=1,12
2     STFLOC(I,J)=0.0D+00

C
C D MATRIX TERMS
C CORRECT IS A CORRECTION TERM FOR THE SHEAR TERMS
C
               CORRECT=5.0D+0/6.0D+00
               EMOD=PROPS(IMAT,1)
               GMOD=PROPS(IMAT,2)
               THK=PROPS(IMAT,3)
               AMU=(EMOD/GMOD/2.0)-1.0
               TOP=(THK**3)/12.0D+0
               BOT=1.0D+0 - AMU**2
               D11=EMOD*TOP/BOT
               D12=AMU*D11
               D22=D11
               D33=GMOD*TOP
               D44=GMOD*THK*CORRECT
C
C  FULL GAUSS QUADRATURE ON BENDING TERMS
C
               DO 10 NI=1,2
               DO 10 NJ=1,2
               XI = GAUSS(NI,2)
               ETA= GAUSS(NJ,2)
               CALL SHAPE(XI,ETA,SF,GDSF,DET,trick)
               CONST=DET*WT(NI,2)*WT(NJ,2)
C
C ADDING STIFFNESS COMPONENTS FOR THIS PART OF GAUSS
C
    II=1
    DO 20 I=1,4
    JJ=1
    DO 30 J=1,4
    STFLOC(II+1,JJ+1) = STFLOC(II+1,JJ+1) +(
D11*GDSF(1,I)*GDSF(1,J)
   &  +D33*GDSF(2,I)*GDSF(2,J) )*CONST
    STFLOC(II+1,JJ+2) = STFLOC(II+1,JJ+2) +(
D12*GDSF(1,I)*GDSF(2,J)
   &  +D33*GDSF(2,I)*GDSF(1,J) )*CONST
    STFLOC(II+2,JJ+1) = STFLOC(II+2,JJ+1) +(
D12*GDSF(2,I)*GDSF(1,J)
   &  +D33*GDSF(1,I)*GDSF(2,J) )*CONST
    STFLOC(II+2,JJ+2) = STFLOC(II+2,JJ+2) +(
D33*GDSF(1,I)*GDSF(1,J)
   &  +D22*GDSF(2,I)*GDSF(2,J) )*CONST
30    JJ=3*J+1
20    II=3*I+1
10    CONTINUE


C
C PARTIAL GAUSS QUADRATURE ON SHEAR TERMS
C
               XI=GAUSS(1,1)
               ETA=XI
               CALL SHAPE(XI,ETA,SF,GDSF,DET,trick)
               CONST=DET*WT(1,1)*WT(1,1)
               II=1
               DO 40 I=1,4
               JJ=1
               DO 50 J=1,4
               STFLOC(II,JJ)=STFLOC(II,JJ)+(D44*GDSF(1,I)*GDSF(1,
J)+
   &               D44*GDSF(2,I)*GDSF(2,J) ) *CONST
               STFLOC(II,JJ+1)=STFLOC(II,JJ+1)+D44*GDSF(1,I)*SF(J
)*CONST
               STFLOC(II+1,JJ)=STFLOC(II+1,JJ)+D44*GDSF(1,J)*SF(I
)*CONST
               STFLOC(II+2,JJ)=STFLOC(II+2,JJ)+D44*GDSF(2,J)*SF(I
)*CONST
               STFLOC(II,JJ+2)=STFLOC(II,JJ+2)+D44*GDSF(2,I)*SF(J
)*CONST
```

```
                STFLOC(II+1,JJ+1)=STFLOC(II+1,JJ+1)+D44*SF(I)*SF(J
)*CONST
                STFLOC(II+2,JJ+2)=STFLOC(II+2,JJ+2)+D44*SF(I)*SF(J
)*CONST
50      JJ=3*J+1
40      II=3*I+1
C
C END OF SUBROUTINE
C
                RETURN
                END
C*************************************************************
******
                SUBROUTINE SHAPE(XI,ETA,SF,GDSF,DET,COORD)
C
C  THIS SUBROUTINE EVALUATES THE INTERPOLATION
FUNCTIONS SF(I)
C  IT THEN EVALUATES THE NATURAL COORDINATE
DERIVATIVES DSF(I)
C       wrt XI IF I=1  wrt ETA IF I=2
C  AND ALSO THE REAL COORDINATE DERIVATIVES GDSF(I)
C       wrt X IF I=1  wrt Y IF I=2
C  GJ IS THE JACOBIAN MATRIX
C  GJINV IS THE JACOBIAN MATRIX INVERSE
C  CNODE IS THE NATURAL COORDINATE POSITION MATRIX
C  NATURAL COORDINATES GO FROM -1 TO 1
C
                IMPLICIT REAL*8 (A-H,O-Z)
                INCLUDE 'control.cmn'
                DIMENSION
COORD(4,2),CNODE(4,2),DSF(2,4),GJ(2,2),
     &  GJINV(2,2),SF(4),GDSF(2,4)
                CNODE(1,1)=-1.0D+0
                CNODE(2,1)=1.0D+0
                CNODE(3,1)=1.0D+0
                CNODE(4,1)=-1.0D+0
                CNODE(1,2)=-1.0D+0
                CNODE(2,2)=-1.0D+0
                CNODE(3,2)=1.0D+0
                CNODE(4,2)=1.0D+0
C
C LINEAR INTERPOLATION FOR A 4 NODED ELEMENT
C
C CALCULATING NATURAL COORDINATES SF AND
C NATURAL COORDINATE DERIVATIVES DSF
C
                DO 10 I=1,4
                XP=CNODE(I,1)
                YP=CNODE(I,2)
                XIO= 1.0D+0 + XI*XP
                ETAO=1.0D+0 + ETA*YP
                SF(I)=0.25D+0 * XIO * ETAO
                DSF(1,I)=0.25D+0 * XP * ETAO
10      DSF(2,I)=0.25D+0 * YP * XIO
C
C CALCULATING JACOBIAN MATRIX AND INVERSE
C
                DO 20 I=1,2
                DO 20 J=1,2
                GJ(I,J)=0.0D+0
                DO 20 K=1,4
20      GJ(I,J)=GJ(I,J)+DSF(I,K)*COORD(K,J)
                DET=GJ(1,1)*GJ(2,2)-GJ(2,1)*GJ(1,2)
                GJINV(1,1)=GJ(2,2)/DET
                GJINV(2,2)=GJ(1,1)/DET
                GJINV(1,2)=-GJ(1,2)/DET
                GJINV(2,1)=-GJ(2,1)/DET
C
C CALCULATING REAL COORDINATE DERIVATIVES
C USING JACOBIAN INVERSE
C
                DO 30 I=1,2
                DO 30 J=1,4
                GDSF(I,J)=0.0D+0
                DO 30 K=1,2
30      GDSF(I,J)=GDSF(I,J)+GJINV(I,K)*DSF(K,J)
C
C END OF SUBROUTINE
C
                RETURN
                END
C*************************************************************
************
      SUBROUTINE
SPLATE(IELEM,IMAT,STFLOC,PROPS,LNODS,COORD)
C
C THIS SUBROUTINE CALCULATES THE MEMBRANE FORCES
FOR A
C 2D BILINEAR TRIANGLE OR THE 3D PLATE ELEMENTS
C
C ELEMENT ASSUMED RECTANGULAR AND CONSTANT
THICKNESS DUDE!
C
                IMPLICIT REAL*8 (A-H,O-Z)
                INCLUDE 'control.cmn'
                DIMENSION
LNODS(NELEM,1),COORD(NPOIN,1),PROPS(NMATRLS,1),
     &  STFLOC(12,12),D(3)
C
C RESET STIFFNESS TO ZERO
C
      DO 5 I=1,12
       DO 5 J=1,12
5       STFLOC(I,J)=0.0
C
C     SET MATERIAL PROPERTIES AND CONSTANTS
                EMOD=PROPS(IMAT,1)
                GMOD=PROPS(IMAT,2)
                T=PROPS(IMAT,3)
                AMU = (EMOD/GMOD/2.00) - 1.0
                D(1)= EMOD/(1-AMU*AMU)
                D(2)= D(1)*AMU
                D(3)= D(1)*(1-AMU)/2.0
                A=(COORD(LNODS(IELEM,1),1)-
COORD(LNODS(IELEM,2),1))**2.0
     &  +(COORD(LNODS(IELEM,1),2)-
COORD(LNODS(IELEM,2),2))**2.0
     &  +(COORD(LNODS(IELEM,1),3)-
COORD(LNODS(IELEM,2),3))**2.0
                A=(DSQRT(A))/2.0
                B=(COORD(LNODS(IELEM,1),1)-
COORD(LNODS(IELEM,4),1))**2.0
     &  +(COORD(LNODS(IELEM,1),2)-
COORD(LNODS(IELEM,4),2))**2.0
     &  +(COORD(LNODS(IELEM,1),3)-
COORD(LNODS(IELEM,4),3))**2.0
                B=(DSQRT(B))/2.0
                write(*,*)'a=',a,'b=',b
                S1=T*B*D(1)/6.0/A
                S2=T*A*D(1)/6.0/B
                S3=T*D(2)/4.0
                S4=T*A*D(3)/6.0/B
                S5=T*B*D(3)/6.0/A
                S6=T*D(3)/4.0
C
C SET STFLOC VALUES
C
                STFLOC(1,1)=2*(S1+S4)
                STFLOC(1,2)=S3+S6
                STFLOC(1,3)=-2*S1+S4
                STFLOC(1,4)=S3-S6
                STFLOC(1,5)=-S1-S4
                STFLOC(1,6)=-S3-S6
                STFLOC(1,7)=S1-2*S4
                STFLOC(1,8)=S6-S3
                STFLOC(2,2)=2*(S2+S5)
                STFLOC(2,3)=S6-S3
                STFLOC(2,4)=S2-2*S5
                STFLOC(2,5)=-S3-S6
                STFLOC(2,6)=-S2-S5
                STFLOC(2,7)=S3-S6
                STFLOC(2,8)=-2*S2+S5
                STFLOC(3,3)=2*(S1+S4)
                STFLOC(3,4)=-S3-S6
                STFLOC(3,5)=S1-2*S4
                STFLOC(3,6)=S3-S6
                STFLOC(3,7)=-S1-S4
                STFLOC(3,8)=S3+S6
                STFLOC(4,4)=2*(S2+S5)
                STFLOC(4,5)=-S3+S6
                STFLOC(4,6)=S5-2*S2
                STFLOC(4,7)=S3+S6
                STFLOC(4,8)=-S2-S5
                STFLOC(5,5)=2*(S1+S4)
                STFLOC(5,6)=S3+S6
                STFLOC(5,7)=-2*S1+S4
                STFLOC(5,8)=S3-S6
                STFLOC(6,6)=2*(S2+S5)
                STFLOC(6,7)=S6-S3
                STFLOC(6,8)=S2-2*S5
                STFLOC(7,7)=2*(S1+S4)
                STFLOC(7,8)=-S3-S6
                STFLOC(8,8)=2*(S2+S5)
C
C COPY LOWER HALF
C
                DO 7 I=1,8
                DO 7 J=1,8
7       STFLOC(J,I)=STFLOC(I,J)
C
```

```
                RETURN
                END
C*************************************************************
****************
      subroutine load(lnods,id,coord,eltrig,rhs,fea,maxdof,nelempg)
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension lnods(nelem,1), id(npoin,1),rhs(1),eltrig(nelem,1),
     *          fea(nelem,1)  , coord(npoin,1), nelempg(ngrp,1),
     *          maxdof(1)
      dimension lm(36),p(6)
c
                WRITE(*,*) ' LOADING '
      do 3 igrp=1,ngrp
       call elematt(nelempg)
       do 5 ielem=elembgn,elemend
                do 7 i=1,nsizk
                 fea(ielem,i)=0.0
7       continue
5       continue
3       continue
c
      do 9 i=1,neq
      rhs(i)=0.0
9       continue
c
      if(nlodno.ne.0)then
       write(output,10) nlodno
       write(output,15)
       do 11 ilodno=1,nlodno
                read(input,*,ERR=99)jnod,(p(i),i=1,6)
                write(output,20)jnod,(p(i),i=1,6)
                write(GRAPH,20)jnod,(p(i),i=1,6)
c
c creating RHS vector
c
                do 13 j=1,6
                  k=id(jnod,j)
                  if(k.ne.0)rhs(k)=rhs(k)+p(j)
13      continue
11      continue
      end if
c
c future use , elemental loading
c for beams and plates distributed loadings
c
c     if(nlodel.ne.0)then
c      write(output,30)
c      write(output,35)
c      do 16 ielem=1,nlodel
c       read(input,*)jelem
c       do 17 igrp=1,ngrp
c        call elematt(nelempg)
c        if(jelem.ge.elembgn.and.jelem.le.elemend)then
c         if(istrtp.eq.6)then
c          read(input,*)w,wxz
c          write(output,45)jelem,w,wxz
c         elseif(istrtp.eq.1.or.istrtp.eq.3.or.istrtp.eq.4)then
c          read(input,*)w
c          write(output,25)jelem,w
c          call endac(jelem,w,wxz,lnods,coord,eltrig,fea,nelempg)
c          idofs=0
c          do 18 inode=1,nnode
c           node =lnods(ielem,inode)
c           do 19 iid=1,nid
c            idofs=idofs+1
c            lm(idofs)=id(node,iposid(iid))
c19          continue
c18         continue
c         do 21 i=1,nsizk
c          k=lm(i)
c          if(k.ne.0)rhs(k)=rhs(k)-fea(jelem,i)
c21         continue
c        end if
c       end if
c17     continue
c16     continue
c     end if
c
      write(output,37)
      write(output,40)(i,rhs(i),i=1,neq)
c
      return
c--------------------
c Format Statements
c--------------------
10    format(//,T5,'Nodal Loads',/    ',i5)
15    format(/,T5,'Node     Loads',/)
20    format(I5,12(1pe14.6))
25    format(T5,I4,7x,1pe12.5)
30    format(//,T5,'Element Loads')
35    format(/,T5,'Element Uniform load',/)
37    format(//,T5,'R.H.S. Vector including F.E.A. ',/)
40    format(T5,i5,1pe12.5,/)
45    FORMAT(T5,i4,' x-y PLANE: ',1pe12.5,' x-z PLANE: ',1pe12.5)
c
99     write(output,*)
     * '***error occured while reading loadings***'
      WRITE(*,*)' ERROR OCCURED WHILE READING LOADS '
      STOP
      end
C*************************************************************
****************
c     subroutine endac(jelem,w,wxz,lnods,coord,eltrig,fea,nelempg)
c
c end action equivelent subroutine for future use
c
c     implicit real*8(a-h,o-z)
c     include 'control.cmn'
c
c     dimension lnods(nelem,1),eltrig(nelem,1),fea(nelem,1)  ,
c     *         coord(npoin,1),nelempg(ngrp,1)
c     dimension ROTATE(36,36),rotinv(36,36),fealoc(36),feag(36)
c
c     nod1=lnods(jelem,1)
c     nod2=lnods(jelem,2)
c     if(ncoor.eq.1)then
c      dlen=dabs(coord(nod2,1)-coord(nod1,1))
c     elseif(ncoor.eq.2)then
c      delx=coord(nod2,1)-coord(nod1,1)
c      dely=coord(nod2,2)-coord(nod1,2)
c      dlen=dsqrt(delx*delx+dely*dely)
c     else
c      delx=coord(nod2,1)-coord(nod1,1)
c      dely=coord(nod2,2)-coord(nod1,2)
c      delz=coord(nod2,3)-coord(nod1,3)
c      dlen=dsqrt(delx*delx+dely*dely+delz*delz)
c     endif
c     do 3 igrp=1,ngrp
c      call elematt(nelempg)
c      if(jelem.ge.elembgn.and.jelem.le.elemend)then
c       if(istrtp.eq.1)then
c        fea(jelem,1)=-w*dlen/2.0
c        fea(jelem,2)=-w*dlen*dlen/12.
c        fea(jelem,3)= fea(jelem,1)
c        fea(jelem,4)=-fea(jelem,2)
c       else
c        if(istrtp.eq.2)then
c         return
c        elseif(istrtp.eq.3)then
c         fealoc(1)=0.0
c         fealoc(2)=-w*dlen/2.
c         fealoc(3)=-w*dlen*dlen/12.
c         fealoc(4)=0.0
c         fealoc(5)=fealoc(2)
c         fealoc(6)=fealoc(3)
c         call rota3(eltrig(jelem,1),rotate)
c        elseif(istrtp.eq.4)then
c         fealoc(1)=0.0
c         fealoc(2)= w*dlen*dlen/12.
c         fealoc(3)=-w*dlen/2
c         fealoc(4)= 0.0
c         fealoc(5)=-fealoc(2)
c         fealoc(6)= fealoc(3)
c         call rota3(eltrig(jelem,1),rotate)
c        elseif(istrtp.eq.5)then
c         return
c        elseif(istrtp.eq.6)then
c         do 5 i=1,nsizk
c          fealoc(i)=0.0
c5       continue
c         fealoc(2 )=-w*dlen/2.
c         fealoc(6 )=-w*dlen*dlen/12.
c         fealoc(8 )= fealoc(2)
c         fealoc(12)=-fealoc(6)
c         fealoc(3 )=-wxz*dlen/2.
c         fealoc(5 )= wxz*dlen*dlen/12.
c         fealoc(9 )= fealoc(3)
c         fealoc(11)=-fealoc(5)
c         call rota6(eltrig(jelem,1),rotate)
c        end if
c        call trans(rotate,rotinv,36,36,nsizk,nsizk)
c        call mult(rotinv,fealoc,feag,36,36,36,36,nsizk,nsizk,1)
c        do 7 i=1,nsizk
c         fea(jelem,i)=feag(i)
c7       continue
c       end if
```

```
c       return
c      end if
c3     continue
c
c     return
c     end
c
c*************************************************************
*****************
      subroutine wrtdis(id,coord,v,disnod,maxdof)
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension
     * id(npoin,1), coord(npoin,1), disnod(npoin,1), v(1),maxdof(1)
c
                  WRITE(*,*) ' WRITING DISPLACEMENTS '
      do 3 ipoin=1,npoin
       do 5 idof=1,maxdof(ipoin)
                  disnod(ipoin,idof)=0.0
5      continue
3      continue
c
      do 7 ipoin=1,npoin
       do 9 idof=1,maxdof(ipoin)
                  if(id(ipoin,idof).ne.0)disnod(ipoin,idof)= v(id(ipoin,idof))
9      continue
7      continue
c
      write(output,10)
c
c     if(maxnumdof.eq.2)write(output,15)
c     if(maxnumdof.eq.3)write(output,20)
c     if(maxnumdof.eq.6)write(output,25)
      write(output,25)
      do 11 ipoin=1,npoin
                  write(output,30)
     *  ipoin,(disnod(ipoin,idof),idof=1,maxdof(ipoin))
                  write(graph,33)
     *  ipoin,(disnod(ipoin,idof),idof=1,maxdof(ipoin))

11     continue
c
      return
c------------------
c format statements
c------------------
10    format(//,T5,'Nodal Displacements')
15    format(/,T5,' Node     dof-1     dof-2',/)
20    format(/,T5,' NODE     dof-1     dof-2     dof-3',/)
25    format(/,T5,' NODE',6X,'dof-1',5X,'dof-2',5X,'dof-3',5X,'dof-4',5X,
     *          'dof-5',5X,'dof-6',/)
30    format(T5,I5,2X,6(1pe12.5))
33    format(i5,6e14.5)
c
      end
c*************************************************************
*********
      subroutine intfor(id,lnods,fea,stfloc,eltrig,disnod,maxdof,
     *                  nelempg,iposek,matno,props,coord)
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      integer nt(4),et(4)
      dimension lnods(nelem,1),id(npoin,1),fea(nelem,1),maxdof(1),
     *          stfloc(1),disnod(npoin,1),eltrig(nelem,1),
     *          nelempg(ngrp,1),iposek(1),matno(1),
     *          props(nmatrls,1),coord(npoin,1)
      dimension eldis(36),eldilo(36),forloc(36),stress(36,36),
     *          ROTATE(36,36),dlcstf(36,36),feagl(36),fealoc(36),
     *          rotran(36,36)
c
c
c setting up shape function data
c
                  data nt / -1, -1, 1, 1 /
                  data et / -1, 1, 1, -1 /
c
      WRITE(*,*)' CALCULATING STRESSES'
      do 3 igrp=1,ngrp
       call elematt(nelempg)
       do 5 ielem=elembgn,elemend
        write(*,*)' Stress for element # -->',ielem
                  do 7 i=1,36
                   do 7 j=1,36
                   stress(i,j)=0.0
                   eldis(i) =0.0
                   eldilo(i)=0.0
                   forloc(i)=0.0
7      continue
c

                   jdof=0
                   if(istrtp.eq.1)then
                    mdof=3
                   elseif(istrtp.eq.2)then
                    mdof=2
                   elseif(istrtp.eq.3)then
                    mdof=3
                   elseif(istrtp.eq.4)then
                    mdof=3
                   elseif(istrtp.eq.5)then
                    mdof=3
                   elseif(istrtp.eq.6)then
                    mdof=6
                   elseif(istrtp.eq.7)then
                    mdof=6
                   elseif(istrtp.eq.8)then
                    mdof=3
                   elseif(istrtp.eq.9)then
                    mdof=6
                   elseif(istrtp.eq.10.or.istrtp.eq.11)then
                    mdof=2
                   elseif(istrtp.eq.12)then
                    mdof=3
                   end if

                   do 9 inod=1,nnode
                   nod=lnods(ielem,inod)
                   do 11 idof=1,mdof
                    jdof=jdof+1
                    j=id(nod,idof)
                    if(j.ne.0) eldis(jdof)=disnod(nod,idof)
11     continue
9      continue
c
       if(istrtp.eq.1)then
                   call rota5(ielem,eltrig,rotate)
                  elseif(istrtp.eq.2)then
                   call rota2(ielem,eltrig,rotate)
                  elseif(istrtp.eq.3)then
                   call rota3(ielem,eltrig,rotate)
                  elseif(istrtp.eq.4)then
                   call rota3(ielem,eltrig,rotate)
                  elseif(istrtp.eq.5)then
                   call rota5(ielem,eltrig,rotate)
                  elseif(istrtp.eq.9)then
                   call rota6(ielem,eltrig,rotate)
                  elseif(istrtp.eq.6)then
                   call rota8(ielem,eltrig,rotate)
                  elseif(istrtp.eq.7.or.istrtp.eq.8)then
                   call rotal0(ielem,eltrig,rotate)
                  end if
c      call trans(rotate,rotran,36,36,36,36)
                  krot=mdof*nnode
                  call mult(rotate,eldis,eldilo,36,36,36,1,nsizk,krot,1)
c
                  write(output,*) 'local displacements for elem#',ielem
                  do 69 idis=1,nsizk
                  write(4,*) eldilo(idis)
69     write(output,*) eldilo(idis)
c
c call stiff if not 3dbeam
c
       if(istrtp.ne.6) then
                  istpos=iposek(ielem)
                  call getastif(dlcstf,stfloc(istpos),nsizk)
                  else
                   nod1=lnods(ielem,2)
                   nod2=lnods(ielem,5)
                   delx=coord(nod2,1)-coord(nod1,1)
                   dely=coord(nod2,2)-coord(nod1,2)
                   delz=coord(nod2,3)-coord(nod1,3)
                   dlen=dsqrt(delx*delx+dely*dely+delz*delz)
                   call partostiff(dlen,matno(ielem),dlcstf,props)
       end if
c
                  call mult(dlcstf,eldilo,forloc,36,36,36,1,nsizk,nsizk,1)
c
c codes for 1d beams & elemental loads
c      if(istrtp.ne.2.and.istrtp.ne.5)then
c       if(istrtp.eq.1)then
c        do 17 i=1,nsizk
c         fealoc(i)=fea(ielem,i)
c17     continue
c      else
```

```
c       do 19 i=1,nsizk
c        feagl(i)=fea(ielem,i)
c19     continue
c       call mult(rotate,feagl,fealoc,36,36,36,1,nsizk,nsizk,1)
c       end if
c       do 21 idof=1,nsizk
c        forloc(idof)=forloc(idof)+fealoc(idof)
c21     continue
c      end if
                write(output,100)ielem
                do 23 i=1,nsizk
                write(output,105)i,forloc(i)
23     continue
c
c ********************************************************
c *  code for stresses written to detailed output file  *
c *  and to end of graphic file  elem#  #of nodes     *
c *       for each node   Sx Sy Sz Txy Tyz Tzx and seq *
c ********************************************************
c
                imat=matno(ielem)
c
                 if(istrtp.eq.2)then
c 2d truss å=Force/area for each xy
                  stress(1,1)=forloc(1)/props(imat,1)
                  stress(2,2)=stress(1,1)
                 write(graph,*)ielem,2

                 elseif(istrtp.eq.1)then
c 3d spring (no stresses available)
                                write(graph,*)ielem,2

                 elseif(istrtp.eq.3)then
                                write(graph,*) ielem,2
                                AR=PROPS(IMAT,2)
                                RJ=PROPS(IMAT,3)
                                CX=PROPS(IMAT,4)
c
c  σxx = Mc/I + Faxial / area
c  σxy = Ftransv / area
c  M = forloc(3),forloc(6)
c  c = CX
c  I = RJ
c  area = AR
c  Faxial = forloc(1),forloc(4)
c  Ftrans = forloc(2),forloc(5)
c
                  IF(FORLOC(3).lt.0) THEN
                                Stress(1,1)=forloc(3)*cx/rj +
DABS(forloc(1)/ar)
                  ELSE
                                Stress(1,1)=forloc(3)*cx/rj -
DABS(forloc(1)/ar)
                  end if

                  Stress(2,2)=0.3*STRESS(1,1)
                  Stress(3,3)=forloc(2)/ar

                  IF(FORLOC(6).lt.0) THEN
                                Stress(4,4)=forloc(6)*cx/rj +
DABS(forloc(4)/ar)
                  ELSE
                                Stress(4,4)=forloc(6)*cx/rj -
DABS(forloc(4)/ar)
                  end if

                  STRESS(5,5)=0.3*STRESS(4,4)
                  Stress(6,6)=forloc(5)/ar

                 elseif(istrtp.eq.4)then
                   write(graph,*)'stress for grid '
c
                 elseif(istrtp.eq.5)then
c 3d truss å=Force/area for each xyz
                  stress(1,1)=forloc(1)/props(imat,1)
                  stress(2,2)=stress(1,1)
                 write(graph,*)ielem,2


                 elseif(istrtp.eq.6)then
C
C STRESS FOR 6 NODED BEAM
C
C by DOUG G. SCOTT
C
C MAJOR AXIAL STRESSES ASSUMED TO BE BENDING
STRESSES
C AND AXIAL STRESS
C
C σxx = Px/Area + ABS(Mx*Cxx/Izz) + ABS(My*Cyy/Iyy)
C
C  Mx = INTERNAL FORCE DOF#6
C  My = INTERNAL FORCE DOF#5
C  Px = INTERNAL FORCE DOF#1
C
C MAJOR SHEAR STRESS ASSUMED TO BE TWISTING SHEAR
AND
C LATERAL FORCE SHEAR
C
C SETTING MATERIAL PROPS
C
     AMU = (props(imat,1)/props(imat,2)/2.00) - 1.0
     AR  = PROPS(IMAT,3)
     CX  = PROPS(IMAT,4)
     CY  = PROPS(IMAT,5)
     RJ  = PROPS(IMAT,6)
     RIY = PROPS(IMAT,7)
     RIZ = PROPS(IMAT,8)
     write(graph,*)ielem,6

C
C SETTING INTERNAL FORCES FOR 2nd NODE AT NEUTRAL AXIS
C USED FOR STRESSES AT NODES 1,2,3
                 U1= -FORLOC(7)
                 U2= dabs(FORLOC(8))
                 U3= dabs(FORLOC(9))
                 V1= dabs(FORLOC(10))
                 V2= FORLOC(11)
                 V3= FORLOC(12)
                 write(output,*)' internal forces used for stress calc. #2 '
                 write(output,*)' tension U1= ',u1
                 write(output,*)' shear   U2= ',u2
                 write(output,*)' shear   U3= ',u3
                 write(output,*)' twist   V1= ',v1
                 write(output,*)' bending V2= ',v2
                 write(output,*)' bending V3= ',v3
C
C NODE 1
C
                 IF(V3.GE.0) THEN
                  Stress(1,1)=V3*CX/RIZ+DABS(V2)*CY/RIY+U1/AR
                 ELSE
                  Stress(1,1)=V3*CX/RIZ-DABS(V2)*CY/RIY+U1/AR
                 END IF
                 write(output,*)' Sx1 =',stress(1,1)
                  Stress(2,2)=amu*stress(1,1)
                  stress(3,3)=amu*stress(1,1)

Stress(1,2)=V1*DSQRT(CX**2+CY**2)/RJ+DSQRT(U2**2+U3**2)/A
R
                  Stress(2,1)=Stress(1,2)
C
C NODE 2
C
                  Stress(4,5) = DSQRT(U2**2+U3**2)/AR
                  Stress(4,4) = U1/AR
                  stress(5,5) = amu*stress(4,4)
                  stress(6,6) = amu*stress(4,4)
                  stress(5,4) = stress(4,5)
C
C NODE 3
C
                 IF(V3.GE.0) THEN
                  Stress(7,7) = -V3*CX/RIZ-DABS(V2)*CY/RIY+U1/AR
                 ELSE
                  Stress(7,7) = -V3*CX/RIZ+DABS(V2)*CY/RIY+U1/AR
                 END IF
                 write(output,*)' Sx3=',stress(7,7)
                  stress(8,8) = amu*stress(7,7)
                  stress(9,9) = amu*stress(7,7)
                  stress(7,8) = stress(1,2)
                  stress(8,7) = stress(1,2)
C
C SETTING INTERNAL FORCES FOR 5th NODE AT NEUTRAL AXIS
C USED FOR STRESSES AT NODES 4,5,6
C
                 U1= FORLOC(25)
                 U2= dabs(FORLOC(26))
                 U3= dabs(FORLOC(27))
                 V1= dabs(FORLOC(28))
                 V2= FORLOC(29)
                 V3= FORLOC(30)

                 write(output,*)' internal forces used for stress calc. #5 '
                 write(output,*)' tension U1= ',u1
                 write(output,*)' shear   U2= ',u2
                 write(output,*)' shear   U3= ',u3
                 write(output,*)' twist   V1= ',v1
```

```
                write(output,*)' bending V2= ',v2
                write(output,*)' bending V3= ',v3
C
C NODE 4
C
                IF(V3.GE.0) THEN
                 Stress(10,10)= V3*CX/RIZ+DABS(V2)*CY/RIY+U1/AR
                ELSE
                 Stress(10,10)= V3*CX/RIZ-DABS(V2)*CY/RIY+U1/AR
                END IF
                write(output,*)' Sx4=',stress(10,10)
                 stress(11,11)= amu*stress(10,10)
                 stress(12,12)= amu*stress(10,10)
                 Stress(10,11)=
V1*DSQRT(CX**2+CY**2)/RJ+DSQRT(U2**2+U3**2)/AR
                 Stress(11,10)= stress(10,11)
C
C NODE 5
C
                Stress(13,14) = DSQRT(U2**2+U3**2)/AR
                stress(14,13) = stress(13,14)
                Stress(13,13) = U1/AR
                stress(14,14) = amu*stress(13,13)
                stress(15,15) = amu*stress(13,13)
C
C NODE 6
C
                IF(V3.GE.0) THEN
                 Stress(16,16) = -V3*CX/RIZ-
DABS(V2)*CY/RIY+U1/AR
                ELSE
                 Stress(16,16) = -
V3*CX/RIZ+DABS(V2)*CY/RIY+U1/AR
                END IF
                write(output,*)' Sx3=',stress(16,16)
                 stress(17,17) = amu*stress(16,16)
                 stress(18,18) = amu*stress(16,16)
                 stress(16,17) = stress(10,11)
                 stress(17,16) = stress(10,11)

C
C END OF BEAM STRESS
C
                elseif(istrtp.eq.7)then
C
C STRESS FOR PLATE
C
C by DOUG G. SCOTT 1992
C
C STRESS FROM MEMBRANE SUPERIMPOSED ON BENDING
PLATE
C
C    Rx = ROTATION ABOUT X-AXIS
C    Ry = ROTATION ABOUT Y-AXIS
C     Z = HALF OF THICKNESS
C
C  STRAIN =   MEMBRANE    +    BENDING
C
                EMOD=PROPS(IMAT,1)
                GMOD=PROPS(IMAT,2)
                THK=PROPS(IMAT,3)
                AMU = (EMOD/GMOD/2.00) - 1.0
                D1= EMOD/(1-AMU*AMU)
                D2= D1*AMU
                D3= D1*(1-AMU)/2.0
                D4= EMOD/2/(1+AMU)
                A=(COORD(LNODS(IELEM,1),1)-
COORD(LNODS(IELEM,2),1))**2.0
   &  +(COORD(LNODS(IELEM,1),2)-
COORD(LNODS(IELEM,2),2))**2.0
   &  +(COORD(LNODS(IELEM,1),3)-
COORD(LNODS(IELEM,2),3))**2.0
                A=(DSQRT(A))/2.0
                B=(COORD(LNODS(IELEM,1),1)-
COORD(LNODS(IELEM,4),1))**2.0
   &  +(COORD(LNODS(IELEM,1),2)-
COORD(LNODS(IELEM,4),2))**2.0
   &  +(COORD(LNODS(IELEM,1),3)-
COORD(LNODS(IELEM,4),3))**2.0
                B=(DSQRT(B))/2.0
                write(graph,*)ielem,4
c
c set local displacements
c
                u1=eldilo(1)
                u2=eldilo(7)
                u3=eldilo(13)
                u4=eldilo(19)
                v1=eldilo(2)
                v2=eldilo(8)
                v3=eldilo(14)
                v4=eldilo(20)
                w1=eldilo(3)
                w2=eldilo(9)
                w3=eldilo(15)
                w4=eldilo(21)
                x1=eldilo(4)
                x2=eldilo(10)
                x3=eldilo(16)
                x4=eldilo(22)
                y1=eldilo(5)
                y2=eldilo(11)
                y3=eldilo(17)
                y4=eldilo(23)
c
                write(4,*)' strains for element 'ielem
                do 35 k=1,4
                n=(k-1)*3
c
c find strains
c
c exx
                exx=((1-nt(k))*(u2-u1)+(1+nt(k))*(u3-u4))/(4*a)
    if(exx.ge.0) then
                 exx=exx+dabs(thk*((1-nt(k))*(y2-y1)+(1+nt(k))*(y3-
y4))/(8*a))
    else
                 exx=exx-dabs(thk*((1-nt(k))*(y2-y1)+(1+nt(k))*(y3-
y4))/(8*a))
    end if
c eyy
                eyy=((1-et(k))*(v4-v1)+(1+et(k))*(v3-v2))/(4*b)
    if(eyy.ge.0) then
                 eyy=eyy+dabs(thk*((1-et(k))*(x4-x1)+(1+et(k))*(x3-
x2))/(8*b))
    else
                 eyy=eyy-dabs(thk*((1-et(k))*(x4-x1)+(1+et(k))*(x3-
x2))/(8*b))
    end if

c exy
                exy=(a*((1-et(k))*(u4-u1)+(et(k)+1)*(u3-u2))
   *    +b*((1-nt(k))*(v2-v1)+(nt(k)+1)*(v3-v4)))/4/a/b

    if(exy.ge.0) then
                 exy=exy+(-a*thk*((1-et(k))*(y4-y1)+(et(k)+1)*(y3-y2))
   *    +b*thk*((1-nt(k))*(x2-x1)+(nt(k)+1)*(x3-x4)))/8/a/b
    else
                 exy=exy-(-a*thk*((1-et(k))*(y4-y1)+(et(k)+1)*(y3-y2))
   *    +b*thk*((1-nt(k))*(x2-x1)+(nt(k)+1)*(x3-x4)))/8/a/b
    end if
c eyz
                eyz= ((1-et(k))*(w4-w1)+(1+et(k))*(w3-w2))/4/b
   *   -x1*(nt(k)-1)*(et(k)-1)/4 +x2*(nt(k)-1)*(et(k)+1)/4
   *   -x3*(nt(k)+1)*(et(k)+1)/4 +x4*(nt(k)+1)*(et(k)-1)/4
c ezx
                ezx= ((1-nt(k))*(w2-w1)+(1+nt(k))*(w3-w4))/4/a
   *   +y1*(nt(k)-1)*(et(k)-1)/4 -y2*(nt(k)-1)*(et(k)+1)/4
   *   +y3*(nt(k)+1)*(et(k)+1)/4 -y4*(nt(k)+1)*(et(k)-1)/4
                write(4,111) exx,eyy,exy,eyz,ezx
c
c find stresses
c
                stress(n+1,n+1)=D1*exx+D2*eyy
                stress(n+2,n+2)=D2*exx+D1*eyy
                stress(n+3,n+3)= amu*( stress(n+1,n+1) + stress(n+2,n+2) )
                stress(n+1,n+2)=D3*exy
                stress(n+2,n+3)=D4*eyz
                stress(n+3,n+1)=D4*ezx
                stress(n+2,n+1)=D3*exy
                stress(n+3,n+2)=D4*eyz
                stress(n+1,n+3)=D4*ezx

35    continue
                end if
c
c end of local stresses
c
c rotate stresses to global coordinates
c
                 if(istrtp.eq.1)then
                 call rota5(ielem,eltrig,rotate)
                elseif(istrtp.eq.2)then
                 call rota2(ielem,eltrig,rotate)
                elseif(istrtp.eq.3)then
                 call rota3(ielem,eltrig,rotate)
                elseif(istrtp.eq.4)then
                 call rota3(ielem,eltrig,rotate)
                elseif(istrtp.eq.5)then
                 call rota5(ielem,eltrig,rotate)
```

```
                elseif(istrtp.eq.9)then
                 call rota6(ielem,eltrig,rotate)
                elseif(istrtp.eq.6)then
                 call rota8(ielem,eltrig,rotate)
                elseif(istrtp.eq.7.or.istrtp.eq.8)then
                 call rota10(ielem,eltrig,rotate)
                end if
c
c this is done by Qt * σl * Q = σg
c
                krot=mdof*nnode
                call mult(stress,rotate,dlcstf,36,36,36,36,krot,nsizk,nsizk)
                call trans(rotate,rotran,36,36,36,36)
                call mult(rotran,dlcstf,stress,36,36,36,36,krot,nsizk,krot)
c
c
c stress now in global coords
c
c write out list of stresses
c
                do 75 inode=1,nnode
                n=(inode-1)*3
C
C Seq is Henky-Von Mises equivalent  stress
C
     seq=( (stress(n+1,n+1)-stress(n+2,n+2))**2
    *     +(stress(n+2,n+2)-stress(n+3,n+3))**2
    *     +(stress(n+3,n+3)-stress(n+1,n+1))**2
    *     +6*(stress(n+1,n+2)**2
    *       +stress(n+2,n+3)**2
    *       +stress(n+3,n+1)**2 ) )**0.5
                seq=seq/dsqrt(2.0d+00)
c
c if stress< 1.0e-09 for graphic output program
c
                if(dabs(stress(n+1,n+1)).lt.1.0e-09)
stress(n+1,n+1)=0.0d+00
                if(dabs(stress(n+2,n+2)).lt.1.0e-09)
stress(n+2,n+2)=0.0d+00
                if(dabs(stress(n+3,n+3)).lt.1.0e-09)
stress(n+3,n+3)=0.0d+00
                if(dabs(stress(n+1,n+2)).lt.1.0e-09)
stress(n+1,n+2)=0.0d+00
                if(dabs(stress(n+2,n+3)).lt.1.0e-09)
stress(n+2,n+3)=0.0d+00
                if(dabs(stress(n+3,n+1)).lt.1.0e-09)
stress(n+3,n+1)=0.0d+00
                if(seq.lt.1.0e-09) seq=0.0d+00

75    write(graph,110) stress(n+1,n+1),stress(n+2,n+2),stress(n+3,n+3),
    *       stress(n+1,n+2),stress(n+2,n+3),stress(n+3,n+1),seq

c
5     continue
3     continue
c
c
c
     return
c------------------
c Format statements
c------------------
100   format(//,T5,'Element Internal Forces for element: ',I5,/,T5,
    *     'dof  force')
105   FORMAT(T5,I5,1pe12.5)
110   format(7e14.5)
111   format(5e14.5)
c
     end
c*************************************************************
******
     subroutine react(id,lnods,fea,stfloc,eltrig,disnod,maxdof,
    *               nelempg,iposek)
c
     implicit real*8(a-h,o-z)
     include 'control.cmn'
     include 'ios.cmn'
c
     dimension lnods(nelem,1) , id(npoin,1)    ,fea(nelem,1)   ,
    *         stfloc(1), disnod(npoin,1),eltrig(nelem,1),
    *         nelempg(ngrp,1), maxdof(1),iposek(1)
c
     dimension
    *         ROTATE(36,36)  ,dpstmt(36,36)  ,rotinv(36,36)  ,
    *         stfelg(36,36),eldis(36),forglb(36),dlcstf(36,36)
c    *         ,reactfor(nelem,6)
c
c
                write(*,*)' CALCULATING REACTION FORCES'

     do 3 igrp=1,ngrp
      call elematt(nelempg)
      do 5 ielem=elembgn,elemend
       write(*,*)'Reaction for element # -->',ielem
       nkrot=nid*nnode
                ireact=0
                do 7 inod=1,nnode
                 nod=lnods(ielem,inod)
                 do 9 idof=1,maxdof(nod)
                  k=id(nod,idof)
                  if(k.eq.0)ireact=1
9     continue
7     continue
                if(ireact.eq.1)then
                 write(output,87)ielem
                 if(istrtp.eq.1)then
                  call rota5(ielem,eltrig,rotate)
                 elseif(istrtp.eq.2)then
                  call rota2(ielem,eltrig,rotate)
                 elseif(istrtp.eq.3)then
                  call rota3(ielem,eltrig,rotate)
                 elseif(istrtp.eq.4)then
                  call rota3(ielem,eltrig,rotate)
                 elseif(istrtp.eq.5)then
                  call rota5(ielem,eltrig,rotate)
                 elseif(istrtp.eq.6)then
                  call rota8(ielem,eltrig,rotate)
                 elseif(istrtp.eq.9)then
                  call rota6(ielem,eltrig,rotate)
                 elseif(istrtp.eq.7.or.istrtp.eq.8)then
                  call rota10(ielem,eltrig,rotate)
                 end if
                 istpos=iposek(ielem)
                 call getastif(dlcstf,stfloc(istpos),nsizk)
                 call
mult(dlcstf,rotate,dpstmt,36,36,36,36,nsizk,nsizk,nkrot)
                 call trans(rotate,rotinv,36,36,nsizk,nkrot)
                 call
mult(rotinv,dpstmt,stfelg,36,36,36,36,nkrot,nsizk,nkrot)
c  old       elseif(istrtp.eq.1)then
c  1D        istpos=iposek(ielem)
c  beam       call getastif(stfelg,stfloc(istpos),nsizk)
c  code      end if
                 do 11 i=1,nkrot
                  eldis(i)=0.0
                  forglb(i)=0.0
11    continue
                 jdof=0
                 do 13 inod=1,nnode
                  nod=lnods(ielem,inod)
                  do 15 idof=1,maxdof(nod)
                   jdof=jdof+1
                   k=id(nod,idof)
                   if(k.ne.0)eldis(jdof)=disnod(nod,idof)
15    continue
13    continue
                 call mult(stfelg,eldis,forglb,36,36,36,36,nkrot,nkrot,1)
c code for       if(istrtp.ne.2.and.istrtp.ne.5)then
c element        do 17 i=1,nkrot
c loadings       forglb(i)=forglb(i)+fea(ielem,i)
c17             continue
c              end if
                 jdof=0
                 do 19 inod=1,nnode
                  nod=lnods(ielem,inod)
                  do 21 idof=1,maxdof(nod)
                   jdof=jdof+1
                   k=id(nod,idof)
                   if(k.eq.0)disnod(nod,idof)=disnod(nod,idof)+forglb(jdof)
21    continue
19    continue
                end if
5     continue
3     continue
c
     write(output,100)
     do 23 ipoin=1,npoin
      do 25 idof=1,maxdof(ipoin)
                if(id(ipoin,idof).eq.0)then
                 write(output,110) ipoin,idof,disnod(ipoin,idof)
                end if
25    continue
23    continue
c
     return
c------------------
c Format Statements
c------------------
87    format(//T5,'Element: ',I4,' has reactions')
100   format(//,T5,'Reactions'/,T5,'Node  Dof     Reaction'/)
```

```
110  FORMAT(T5,i4,i5,1pe12.5)
c
     end
c*************************************************************
*******
c
     subroutine addmat(a,b,nsizl,nsizk,nl,nk,kode)
c
c  This subroutine adds or subtracts matrices a and b and stores the
c  result in matrix a. nsizl and nsizk dimension the arrays in the
c  subroutine nl and nk is the actual size of the matrices.kode is
c  "0" for addition and "1" for subtraction.
c  written by: Kent Montgomery
c  called from: Main
c  subroutines called: none
c
     implicit real*8(a-h,o-z)
     dimension a(nsizl,nsizk),b(nsizl,nsizk)
c
c
c
     do 3 i=1,nl
                  do 5 j=1,nk
                    if(kode.eq.0) then
                      a(i,j)=a(i,j)+b(i,j)
                    else if(kode.eq.1) then
                      a(i,j)=a(i,j)-b(i,j)
                    endif
5      continue
3      continue
     return
     end
c*************************************************************
***
     subroutine trans(a,b,nsizl,nsizk,nl,nk)
c
c  This subroutine calculates the transpose of matrix a and
c  stores it in matrix b.nsizl and nsizk dimension the arrays
c  in the subroutine.nl and nk is the actual size of the matrices.
c  written by: Kent Montgomery
c  corrected by: Doug G. Scott (Jan/1992)
c  called from: Main
c  subroutines called: none
c
     implicit real*8(a-h,o-z)
     dimension a(nsizl,nsizk),b(nsizk,nsizl)
c
c
     do 3 i=1,nl
                  do 5 j=1,nk
                    b(j,i)=a(i,j)
5      continue
3      continue
     return
     end
c
c*************************************************************
******
c
     subroutine mult(a,b,c,nra,nca,nrb,ncb,nl,nkl,nk)
c
c  This subroutine multiplies matrices a and b ([a][b]) and
c  stores the result in matrix c. [nra,nca] & [nrb,ncb]
c  dimension the arrays in the subroutine.nl,nkl,and nk is the
c  actual size of the arrays.(nl,nkl)*(nkl,nk)=(nl,nk)
c  written by: Kent Montgomery
c  changed by: Doug G. Scott
c  called from: Main
c  subroutines called: none
c
     implicit real*8(a-h,o-z)
     dimension a(nra,nca),b(nrb,ncb),c(nra,ncb)
c
c  Initialize matrix c
c
     do 3 i=1,nl
                  do 5 j=1,nk
                    c(i,j)=0.0
5      continue
3      continue
c
c  Multiply the matrices.
c
     do 7 i=1,nl
                 do 9 j=1,nkl
                   do 11 k=1,nkl
                     c(i,j)=a(i,k)*b(k,j)+c(i,j)
11     continue
9      continue
7      continue
     return
     end

C*************************************************************
**********************
     SUBROUTINE sky(A,V,MAXA,NN,KODE)
C*************************************************************
**********************
     IMPLICIT REAL*8(A-H,O-Z)
     include 'ios.cmn'
C
     DIMENSION MAXA(1),A(1),V(1)
     WRITE(*,*) ' INCORE SOLVER WORKING '
C-----------------------------------------
C READ UDU,MAXA WHEN KODE.GT.1 ,  AND RETURN
C-----------------------------------------
C
     IF(KODE.GT.1)GO TO 150
C-----------------------------------------------------
C PERFORM L*D*L(T) FACTORIZATION OF STIFFNESS MATRIX
C-----------------------------------------------------
40   CONTINUE
     DO 140 N=1,NN
     write(*,*) 'working on equation # -->',n
     KN=MAXA(N)
     KL=KN+1
     KU=MAXA(N+1)-1
     KH=KU-KL
     IF(KH)110,90,50
50   K=N-KH
     IC=0
     KLT=KU
     DO 80 J=1,KH
     IC=IC+1
     KLT=KLT-1
     KI=MAXA(K)
     ND=MAXA(K+1)-KI-1
     IF(ND)80,80,60
60   KK=MIN0(IC,ND)
     C=0.0
     DO 70 L=1,KK
70   C=C+A(KI+L)*A(KLT+L)
     A(KLT)=A(KLT)-C
80   K=K+1
90   K=N
     B=0.0
     DO 100 KK=KL,KU
     K=K-1
     KI=MAXA(K)
     C=A(KK)/A(KI)
     B=B+C*A(KK)
100  A(KK)=C
     A(KN)=A(KN)-B
110  IF(A(KN))130,120,140

120  WRITE(*,*)' STOP!! -  STIFFNESS MATRIX NOT POSITIVE
DEFINITE'
     WRITE(*,2001)N
     WRITE(*,2002)A(KN)
     WRITE(*,*)' SOLUTION ABORTED!!'
     WRITE(*,*)' CHECK MATERIAL PROPERTIES ? '

     STOP

130  WRITE(*,*)'STOP!! -  STIFFNESS HAS ZERO VALUE '
     WRITE(*,2001)N
     WRITE(*,2002)A(KN)
     WRITE(*,*)'SOLUTION ABORTED!!'
     WRITE(*,*)'CHECK BOUNDARY CONDITIONS!! '

     STOP

140  CONTINUE
C-----------------------
C REDUCE V & LOAD VECTOR
C-----------------------
150  continue
     DO 180 N=1,NN
     KL=MAXA(N)+1
     KU=MAXA(N+1)-1
     IF(KU-KL)180,160,160
160  K=N
     C=0.0
     DO 170 KK=KL,KU
     K=K-1
170  C=C+A(KK)*V(K)
     V(N)=V(N)-C
180  CONTINUE
C----------------
C BACK-SUBSTITUTE
```

```
C----------------
      DO 200 N=1,NN
      K=MAXA(N)
200   V(N)=V(N)/A(K)
      IF(NN.EQ.1)RETURN
      N=NN
      DO 230 L=2,NN
      KL=MAXA(N)+1
      KU=MAXA(N+1)-1
      IF(KU-KL)230,210,210
210   K=N
      DO 220 KK=KL,KU
      K=K-1
220   V(K)=V(K)-A(KK)*V(N)
230   N=N-1
                  RETURN


2001  FORMAT('      NONPOSITIVE PIVOT FOR EQUATION  ',I4)
2002  FORMAT('             PIVOT =',D20.12)


      END

c**********************************************************
**********
      subroutine colht(lnods,id,maxa,nelempg,maxdof)
c
      implicit real*8(a-h,o-z)
      include 'control.cmn'
      include 'ios.cmn'
c
      dimension lnods(nelem,1),id(npoin,1),nelempg(ngrp,1),
     *          maxdof(1),maxa(1)
      dimension lm(36)
c
      lrgnum=10*neq
      do 5 ieq=1,neq+1
       maxa(ieq)=lrgnum
5     continue
      do 7 igrp=1,ngrp
       call elematt(nelempg)
       do 9 ielem=elembgn,elemend
                  idof=0
                  do 11 inod=1,nnode
                   nod=lnods(ielem,inod)
                   do 13 i=1,maxdof(nod)
                    idof=idof+1
                    lm(idof)=id(nod,i)
13                 continue
11                 continue
                  nevab=idof
                  mindof=neq
                  do 15 idof=1,nevab
                   jdof=lm(idof)
                   if(jdof.ne.0)mindof=min0(mindof,jdof)
15                 continue
                  do 17 idof=1,nevab
                   jdof=lm(idof)
                   if(jdof.ne.0)maxa(jdof)=min0(maxa(jdof),mindof)
17                 continue
9                  continue
7                  continue
c
      nsizg=0
      do 19 ieq=1,neq
       nsizg=nsizg+ieq-maxa(ieq)+1
19    continue
      maxa(neq+1)=nsizg+1
c
      do 21 ieq=neq,1,-1
                  maxa(ieq)=maxa(ieq+1)-(ieq-maxa(ieq)+1)
21    continue
c
      write(2,*)' maxa array'
      write(2,33)(ieq,maxa(ieq),ieq=1,neq+1)
33    format(2i10)
c
      return
      end
```

Appendix B5

OUTPUT PROGRAM

```
                PROGRAM MHYFEC_OUTPUT
C
C THIS IS THE POSTPROCESSOR FOR Manitoba HYdro Finite
Element Code
C
C     WRITTEN BY:
C          DOUGLAS G. SCOTT
C          MECH. ENG. GRADUATE STUDENT
C          FEBURARY/MARCH 1992
C
C     IT IS DESIGNED TO SHOW THE DEFLECTIONS AND
STRESSES
C     IN THE FINITE ELEMENT MODELS OF 2 AND 3
DIMENSIONAL
C     STRUCTURES AS WELL AS AXIS-SYMMETRIC OBJECTS.
C
C
C     VARIABLE LIST
C
C     COUNTERS
C
C     NOG    NUMBER OF GROUPS
C     NOE    NUMBER OF ELEMENTS
C     NON    NUMBER OF NODES
C     NBC    NUMBER OF BOUNDARY CONDITIONS
C
C     POINTERS
C
C     NPRESG    GROUP#
C     NPRESE    ELEMENT#
C     NPRESN    NODE#
C     NODIM     NUMBER OF DIMENSIONS
C
C     ELEMENT PROPERTIES
C
C     NUMNOD  NUMBER OF NODES IN ELEMENT  (TYPE#)
C     NUMCOL  NUMBER OF COLOUR OF ELEMENT (TYPE#)
C     NEC     ELEMENT CONECTIVITY,G# (9)
C
C     GROUP
C
C     NOEIG   NUMBER OF ELEMENTS IN THIS GROUP
C     NOETY   NUMBER OF ELEMENT TYPE IN THIS GROUP
C     NOGAC   ACTIVITY STATUS OF GROUP (SHOW=1/NO=-1)
C
C     GRAPHIC FLAGS
C
C     KNN     FLAG FOR NODE NUMBER 1=ON -1=OFF
C     KEN     FLAG FOR ELEM NUMBER 1=ON -1=OFF
C     KNS     FLAG FOR NODE GRAPH  1=ON -1=OFF
C     KES     FLAG FOR ELEM GRAPH  1=ON -1=OFF
C     NOTE: K*S MUST=1 FOR NUMBERING TO FUNCTION
C     KSOW    FLAG SOLID OR WIREFRAME 0=WIRE,1=SOLID
C     KBC     BOUNDARY CONDITIONS SHOW=1/NO=-1
C     KFOR    FORCES SHOW=1/NO=-1
C
C     BOUNDARY CONDITIONS
C
C     ID     BOUNDARY CONDITION ARRAY  6dof
C     IDL    B.C. LOCATIONS
C
C     OTHERS
C     PN     NODAL POSITIONS(5) (3D + 2 FOR GRAPHIC)
C     F      NODAL FORCE ARRAY
C     INF    NODAL FLAG IF EXISTS 0=NO
C     IEF    ELEMT FLAG IF EXISTS 1=YES
C     STS    STRESSES
C
C    STS   STRESS MATRIX STS(800,4,7)
C 800 DISP  RDISP(800,6)
C 800 NODES  PN(800,5) (X,Y,Z,XX,YY)
C               X Y Z ARE REAL COORDINATES
C               XX YY ARE MAPPED ONTO SCREEN COORDS.
C 800 ELEMENTS   NEC(800,9) (8 NODES, GROUP#)
C 25 GROUPS #ELM IN GROUP, ELEMTYPE, ACTIVITY
C 800 FORCES  F(node#, 6 DOF FORCES)
C 200 BOUNDARY CONDITIONS  ID(6 DOF FIX), IDL(NODE WHICH
FIX)

C
C     DECLARATIONS
C
              CHARACTER*20 TITLE
              INCLUDE 'COMMON.F
C
C     OPEN GKS ENVIROMENT
C
              XMN=-10.0
              YMN=-7.4
              XMX=20.0
              YMX=14.8
              XDIS=XMX-XMN
              YDIS=YMX-YMN
              CALL OPEN(XMN,YMN,XMN,YMX)
              CALL GSELNT (2)
C
C     INCLUDE COLOR,BUTTON,INFO
              INCLUDE 'COLOUR.F'
C
C     DRAW BACKGROUND
C
              CALL BOX(-1.0,31.0,-21.0,100.0,GREY,1)

              INCLUDE 'POPBUT.F'
              CALL GSELNT (2)
              INCLUDE 'NEWINFO.F'
              CALL GSTXCI(14)
              CALL GSCHH(3.0)
              CALL GTX(0.0,-18.0,'MHYFECS OUTPUT')

C
C     SET DEFAULTS
C
              TITLE='UNTITLED'
              KNN=-1
              KEN=-1
              KNS=1
              KES=1
              AS=0.0
              AP=1.57079
              KSOW=1
              KBC=-1
              KFOR=-1
              NODIM=1
C
C TITLE AND ICON HIGHLIGHT
C
45    CALL GSLWSC(3.0)
              CALL GSPLCI(PURPLE)
              CALL GSCHH(3.0)
              CALL GSTXCI(ORANGE)
              CALL GTX(1.0,95.0,TITLE)
              CALL GSMK(1)
              CALL GSCHH(1.5)
C
C *** START OF MAIN ROUTINE ***
C
50    CALL GRQPK(1,2,II,SEG,JJ)
C
C     MAIN PICK ROUTINE
C
C
              CSC=1
              IF(SEG.EQ.BVAL) THEN
                        CALL SEARCH()
                        CALL DRAW(1,'  ',0)
              ELSEIF(SEG.EQ.BU) THEN
                        CALL ROTATE(0.0,-1.0)
                        CALL DRAW(1,'  ',1)
              ELSEIF(SEG.EQ.BD) THEN
                        CALL ROTATE(0.0,1.0)

                        CALL DRAW(1,'  ',1)
              ELSEIF(SEG.EQ.BR) THEN
                        CALL ROTATE(1.0,0.0)

                        CALL DRAW(1,'  ',1)
              ELSEIF(SEG.EQ.BL) THEN
                        CALL ROTATE(-1.0,0.0)

                        CALL DRAW(1,'  ',1)
              ELSEIF(SEG.EQ.REDRAW) THEN

                        CALL DRAW(1,'  ',1)
              ELSEIF(SEG.EQ.BPC) THEN
                        CALL CENTER()
              ELSEIF(SEG.EQ.BZI) THEN
                        CALL ZOOMIN()
              ELSEIF(SEG.EQ.BZO) THEN
                        CALL ZOOMOUT()
```

```
                ELSEIF(SEG.EQ.BRES) THEN
                        CALL RESET()
                ELSEIF(SEG.EQ.BEON) THEN
                        KES=-KES
                ELSEIF(SEG.EQ.BENON) THEN
                        KEN=-KEN
                ELSEIF(SEG.EQ.BNON) THEN
                        KNS=-KNS
                ELSEIF(SEG.EQ.BSOW) THEN
                    IF(KSOW.EQ.0) THEN
                        KSOW=1
                        ELSE
                        KSOW=0
                        END IF
                ELSEIF(SEG.EQ.BNNON) THEN
                        KNN=-KNN
                ELSEIF(SEG.EQ.BBC) THEN
                        KBC=-KBC
                ELSEIF(SEG.EQ.BFOR) THEN
                        KFOR=-KFOR
                ELSEIF(SEG.EQ.BLIST) THEN
                        CALL LIST()
                ELSEIF(SEG.EQ.FILE) THEN
                        CALL FILESUB(TITLE)
                ELSEIF(SEG.EQ.DISP) THEN
                        CALL DISPLACE()
                ELSEIF(SEG.EQ.STRESS) THEN
                        CALL PICKSTRESS()

                END IF
                GOTO 50
C
C *** END OF MAIN ROUTINE ***
C
                END
C*******************************************************
*******

    SUBROUTINE CLOSE

*   CLOSE GKS
    CALL GCLRWK(1,1)
    CALL GDAWK (0)
    CALL GDAWK (1)
    CALL GDAWK (2)
    CALL GCLWK (2)
    CALL GCLWK (0)
    CALL GCLWK (1)
    CALL GCLKS ()
    CLOSE (14)
    RETURN
    END


C*******************************************************
*******
    SUBROUTINE OPEN(XMN,YMN,XMX,YMX)
C   OPEN GKS AND SET VIEWPORT AND WINDOW
    INTEGER*2 ERR,DCUNIT,XRAS,YRAS
    REAL
XDCMX,YDCMX,SCALE,XNDC,YNDC,XMN,XMX,YMN,YMX
    CHARACTER*15 PROMPT
    prompt='>               '
    OPEN (14,FILE='ERRORS')
    CALL GOPKS (14,1024)
    CALL GOPWK (0,0,0)
    CALL GOPWK (1,0,1)
    CALL GOPWK (2,0,2)
    CALL GACWK (0)
    CALL GACWK (1)
    CALL GACWK (2)
    CALL GQDSP (1,ERR,DCUNIT,XDCMX,YDCMX,XRAS,YRAS)
    IF (XDCMX.GT.YDCMX) THEN
                SCALE=XDCMX
    ELSE
                SCALE=YDCMX
    END IF
    XNDC=XDCMX/SCALE
    YNDC=YDCMX/SCALE
    CALL GSWKWN (1,0.0,XNDC,0.0,YNDC)
    CALL GSWKVP (1,0.0,XDCMX,0.0,YDCMX)

C
C TRANSFORMATION #1
C
    CALL GSWN (1,XMN,XMX,YMN,YMX)
    CALL GSVP (1,0.0,0.745*XNDC,0.0,YNDC)
C
C TRANSFORMATION #2
C
    CALL GSWN (2,-1.0,31.0,-21.0,100.0)
    CALL GSVP (2,0.75*XNDC,XNDC,0.0,YNDC)
C
C TRANSFORMATION #3
C
    CALL GSWN (3,0.0,20.0,0.0,20.0)
    CALL GSVP (3,0.1*XNDC,0.7*XNDC,0.1*YNDC,0.9*YNDC)
C
C TEST FONT
    CALL GSTXFP (-101,2)
C LOCATOR MODE
    CALL GSLCM(1,2,0,1)
C INITIALIZE STRING
    CALL
GINST(1,1,15,PROMPT,1,0.75*XDCMX,XDCMX,0.8347*YDCMX,0.9*
YDCMX
   & ,20,1,10,PROMPT)
C INITIALIZE PICK
    CALL GINPK(1,2,1,1,1,1,0.0,XDCMX,0.0,YDCMX,10,PROMPT)
    CALL GDAWK (2)
    RETURN
    END
C
C*******************************************************
*********
C
                subroutine box (xmin,xmax,ymin,ymax,IC,IS)
                real xmin, ymin, xmax, ymax,x(5), y(5)
                INTEGER IC,IS
c
                x(1) = xmin
                y(1) = ymin
                x(2) = xmax
                y(2) = ymin
                x(3) = xmax
                y(3) = ymax
                x(4) = xmin
                y(4) = ymax
                x(5) = xmin
                y(5) = ymin
                CALL GSFAIS(IS)
                CALL GSFACI(IC)
                CALL GFA (5, x, y)
                return
                end
C
C*******************************************************
**************
C
                SUBROUTINE FILESUB(TITLE)
                CHARACTER*20 TITLE,BUF,DUM,PATH
                CHARACTER*40 TOTAL
                DIMENSION SUMS(7)
                INCLUDE 'COMMON.F'


C
C THIS SUBROUTINE  i) OPENS & READS A DATA FILE
C           ii) SETS THE PATH FOR FILE USAGE
C           iii) QUITS THE MAIN PROGRAM
C           iv) NODAL STRESS AVERAGING IN COMMON
C              NODES BETWEEN SAME TYPE OF ELEMENTS
C
C     LOCAL VARIABLES
C
C     IPL...LENGTH OF PATH NAME
C     USED TO JOIN TOGETHER TO MAKE TOTAL
C     WHICH IS USED TO READ/WRITE FILES
C
C     TITLE...NAME OF MODEL/FILE
C     PATH....PATH TO DIRECTORY USED FOR
READING/WRITING
C     TOTAL...PATH & TITLE JOINED TOGETHER FOR OPEN A
FILE
C
C     BUF.....CHARACTER BUFFER THAT ALL INPUT IS READ
INTO
C     DUM.....DUMMY CHARACTER FOR COMPARING INPUT
Y/N?
C
C     DEFAULT PATH
                IF(NOE.EQ.0) PATH='.    '
                IPL=1
C     WRITING BUTTON NAMES
C
                CALL GSTXCI(ORANGE)
                CALL GTX( 2.0, 34.0, 'OPEN')
                CALL GTX(12.0, 34.0,  'PATH')
                CALL GSTXCI(14)
                CALL GTX( 2.0, 24.0, 'DONE')
                CALL GSTXCI(DKRED)
```

```
              CALL GTX(12.0, 24.0, 'QUIT')
              CALL GSTXCI(ORANGE)
C
C  MAIN PICK
C
5     CALL GRQPK(1,2,II,SEG,JJ)
              CALL GSCHH(1.5)
              CALL GSTXCI(ORANGE)
C QUIT
              IF(SEG.EQ.FIVE) THEN
               CALL CLOSE
               STOP
C PATH
              ELSEIF(SEG.EQ.TWO) THEN
               CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
               CALL GTX(2.0,85.0,'ENTER FILE PATH')
               CALL GRQST(1,1,I,J,PATH)
               DO 700 I=1,15
               DUM=PATH(1:(17-I))
               BUF=PATH(1:(16-I))
700      IF (BUF.EQ.DUM) IPL=16-I
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
C DONE
              ELSEIF(SEG.EQ.FOUR) THEN
               CALL GSTXCI(DKBLUE)
              CALL GTX( 2.0, 34.0, 'OPEN')
              CALL GTX(12.0,34.0, 'PATH')
              CALL GTX( 2.0, 24.0, 'DONE')
              CALL GTX(12.0, 24.0, 'QUIT')
               RETURN
C
C OPEN
              ELSEIF(SEG.EQ.ONE) THEN
              CALL GSTXFP (1,2)
              CALL GSTXCI(WHITE)
              XMN=0.0
              YMN=0.0
              XMX=30.0
              YMX=22.0
              XDIS=XMX
              YDIS=YMX
              CALL GSWN (1,XMN,XMX,YMN,YMX)
              CALL GSELNT (1)
              CALL GSCHH ( YDIS/27.0 )
              OPEN(UNIT=8,FILE='TEMP.LST')
              X(1)=XMN+0.05*XDIS
78    CALL BOX(xmn,xmx,ymn,ymx,black,1)
              DO 77 I=1,20
              READ(8,*,ERR=77,END=79) BUF
              Y(1)=YMN+0.97*YDIS-I*0.048*YDIS
              CALL GTX(X(1),Y(1),BUF)
77    CONTINUE
              Y(1)=YMN+0.97*YDIS-I*0.048*YDIS
              CALL GTX(X(1),Y(1),'ENTER FOR MORE')
              CALL GRQST(1,1,I,J,DUM)
              GOTO 78
79    Y(1)=YMN+0.97*YDIS-(I+1)*0.048*YDIS
              CALL GTX(X(1),Y(1),'END OF DATA FILES [ENTER]')
              CALL GSTXFP (-101,2)
              CALL GSELNT (2)
               CALL GSTXCI(ORANGE)
               CALL GSCHH(1.5)
               CLOSE(UNIT=8)
               CALL BOX(-1.0,31.0,80.4,100.0,GREY,1)
               CALL GTX(2.0,85.0,'ENTER FILE NAME')
               CALL GRQST(1,1,I,J,TITLE)
                CALL GSELNT (1)
                CALL BOX(xmn,xmx,ymn,ymx,black,1)
                CALL GSELNT (2)
               CALL BOX(-1.0,31.0,80.4,90.7,GREY,1)
               TOTAL=PATH(1:IPL)//\/\//TITLE
               OPEN(UNIT=4,FILE=TOTAL,STATUS='OLD',ERR=799)
C
C READ CONTROL DATA
              READ(4,*,ERR=990)NON,NAN,NOE,NAE,NODIM,NOG
,NBC,I,J
C READ NODES
              DO 60 I=1,NAN
               READ(4,*,ERR=991) K,(PN(K,J),J=1,NODIM)
                INF(K)=1
                PN(K,4)=PN(K,1)
                PN(K,5)=PN(K,2)
                PN(K,6)=PN(K,1)
60       PN(K,7)=PN(K,2)
C READ GROUPS AND CONNECTIVITY
              DO 65 I=1,NOG
               NOGAC(I)=1
               READ(4,*,ERR=992) NOETY(I),NOEIG(I),J
              DO 65 J=1,NOEIG(I)
                    READ(4,*,ERR=992) K,(
NEC(K,IJ),IJ=1,NUMNOD(NOETY(I)) )
                    IEF(K)=1
65       NEC(K,9)=I

C READ BOUNDARY CONDITIONS
                    DO 70 I=1,NBC
70       READ(4,*,ERR=993) IDL(I),( ID(I,J),J=1,6)
C READ FORCES
                   READ(4,*,ERR=994) j,I
                   DO 80 I=1,NON
80       READ(4,*,ERR=994) j,F(I,1),F(I,2),F(I,3)
   @                    ,F(I,4),F(I,5),F(I,6)
                   CALL SEARCH()
                   CALL DRAW(1,'  ',1)

C READ DISPLACEMENTS
                   DO 85 I=1,NAN
85       READ(4,*,ERR=995) J,(RDISP(J,K),K=1,6)
C READ STRESSES
C

                   DO 90 I=1,NAE
                    READ(4,*,ERR=996) NPRESE,NSN
                    write(buf,*) i
                    CALL GTX(2.0,85.0,buf)
                    DO 90 K=1,NSN
90        READ(4,*,ERR=996) ( STS(NPRESE,K,J),J=1,7 )

                   CLOSE(UNIT=4)
C
C DRAWING OPENED FILE
                   CALL GSTXCI(ORANGE)
                   CALL GSCHH(3.0)
                   CALL GTX(1.0,95.0,TITLE)
                   CALL GSCHH(1.5)
                   GOTO 5
C ERROR IN DATA
990       CALL ERROR(102)
                   TITLE='UNTITLED'
                   CALL GSTXCI(ORANGE)
                   CALL GSCHH(3.0)
                   CALL GTX(1.0,95.0,TITLE)
                   CALL GSCHH(1.5)
                   GOTO 5
991       CALL ERROR(106)
                   TITLE='UNTITLED'
                   CALL GSTXCI(ORANGE)
                   CALL GSCHH(3.0)
                   CALL GTX(1.0,95.0,TITLE)
                   CALL GSCHH(1.5)
                   GOTO 5
992       CALL ERROR(107)
                   TITLE='UNTITLED'
                   CALL GSTXCI(ORANGE)
                   CALL GSCHH(3.0)
                   CALL GTX(1.0,95.0,TITLE)
                   CALL GSCHH(1.5)
                   GOTO 5
993       CALL ERROR(108)
                   TITLE='UNTITLED'
                   CALL GSTXCI(ORANGE)
                   CALL GSCHH(3.0)
                   CALL GTX(1.0,95.0,TITLE)
                   CALL GSCHH(1.5)
                   GOTO 5
994       CALL ERROR(109)
                   TITLE='UNTITLED'
                   CALL GSTXCI(ORANGE)
                   CALL GSCHH(3.0)
                   CALL GTX(1.0,95.0,TITLE)
                   CALL GSCHH(1.5)
                   GOTO 5
995       CALL ERROR(110)
                   TITLE='UNTITLED'
                   CALL GSTXCI(ORANGE)
                   CALL GSCHH(3.0)
                   CALL GTX(1.0,95.0,TITLE)
                   CALL GSCHH(1.5)
                   GOTO 5
996       CALL ERROR(111)
                   TITLE='UNTITLED'
                   CALL GSTXCI(ORANGE)
                   CALL GSCHH(3.0)
                   CALL GTX(1.0,95.0,TITLE)
                   CALL GSCHH(1.5)
                   GOTO 5

C ERROR IN FILE/PATH
799       CALL ERROR(103)
```

```
            END IF
            GOTO 5
            END
C
C*************************************************************
**************
C
            SUBROUTINE ERROR(MNM)
            INCLUDE 'COMMON.F'
            INTEGER MNM


            CALL GSPLCI(0)
            CALL GSELNT (3)
            CALL GSVIS(101,1)
            CALL GSTXCI(YELLOW)
            CALL GSCHH(0.6)

            IF(MNM.EQ.102) THEN
             CALL GTX( 6.5,13.5,'ERROR READING CONTROL
DATA ')
             CALL GTX( 6.5,12.0,'  CLICK HERE TO CONTINUE ')

            ELSEIF(MNM.EQ.103) THEN
             CALL GTX( 6.5,13.5,'ERROR IN PATH OR FILENAME')
             CALL GTX( 6.5,12.0,' CLICK HERE TO CONTINUE ')

            ELSEIF(MNM.EQ.104) THEN
             CALL GTX( 6.5,12.0,' CLICK HERE TO CONTINUE ')
             CALL GTX( 6.5,13.5,'  GROUP DOES NOT EXIST  ')

            ELSEIF(MNM.EQ.105) THEN
             CALL GTX( 6.5,13.5,'     INVALID NUMBER ')
             CALL GTX( 6.5,12.0,'  CLICK HERE TO CONTINUE ')

            ELSEIF(MNM.EQ.106) THEN
             CALL GTX( 6.5,13.5,'  ERROR READING IN NODES')
             CALL GTX( 6.5,12.0,'  CLICK HERE TO CONTINUE ')

            ELSEIF(MNM.EQ.107) THEN
             CALL GTX( 6.5,13.5,'ERROR READING IN
ELEMENTS')
             CALL GTX( 6.5,12.0,'  CLICK HERE TO CONTINUE ')

            ELSEIF(MNM.EQ.108) THEN
             CALL GTX( 6.5,13.5,'ERROR READING BOUNDARY
COND')
             CALL GTX( 6.5,12.0,'  CLICK HERE TO CONTINUE ')

            ELSEIF(MNM.EQ.109) THEN
             CALL GTX( 6.5,13.5,' ERROR IN READING FORCES')
             CALL GTX( 6.5,12.0,' CLICK HERE TO CONTINUE ')

            ELSEIF(MNM.EQ.110) THEN
             CALL GTX( 6.5,13.5,'ERROR READING
DISPLACEMENTS')
             CALL GTX( 6.5,12.0,'  CLICK HERE TO CONTINUE ')

            ELSEIF(MNM.EQ.111) THEN
             CALL GTX( 6.5,13.5,'ERROR IN READING STRESSES')
             CALL GTX( 6.5,12.0,' CLICK HERE TO CONTINUE ')

            END IF

5    CALL GRQPK(1,2,II,IJ,JJ)
            IF(IJ.NE.101) GOTO 5
            CALL GSVIS(101,0)
            call draw(1,'  ',1)
            CALL GSELNT (2)
            CALL GSPLCI(PURPLE)
            RETURN
            END
C
C*************************************************************
**************
C
            SUBROUTINE DISPLACE()
            INCLUDE 'COMMON.F'
            CHARACTER*10 DUM

C
C THIS SUBROUTINE  i) SHOWS EXAGERATED DISPLACMENT
C              SOLID OR WIREFRAME
C           ii) SHOWS SUPERIMPOSED DISPLACEMENT
C              DISPLACED SHAPE IN WIRE, ORIGINAL IN SOLID
C          iii) TURNS GROUPS ON/OFF
C
C
C
C     BUF.....CHARACTER BUFFER THAT ALL INPUT IS READ
INTO

C     WRITING BUTTON NAMES
C
            CALL GSTXCI(ORANGE)
C 1
            CALL GTX( 2.5, 35.0, 'SOLID ')
            CALL GTX( 2.0, 31.0, 'DISPL.')
C 2
            CALL GTX( 12.0, 37.0, ' WIRE ')
            CALL GTX( 12.5, 34.0, 'FRAME ')
            CALL GTX( 12.5, 31.0, 'DISPL.')
C 3
            CALL GTX(22.0, 37.0, ' SUPER ')
            CALL GTX(22.0, 34.0, 'IMPOSED')
            CALL GTX(22.5, 31.0, 'DISPL.')
C 9
            CALL GTX( 22.0, 15.0, 'GROUPS')
            CALL GTX( 22.0, 11.0, 'ON/OFF')
            CALL GSTXCI(14)
C 7
            CALL GTX( 2.0, 14.0, 'DONE')
            CALL GSTXCI(ORANGE)
            IPLAN=0
C
C  MAIN PICK
C
5    CALL GRQPK(1,2,II,SEG,JJ)
            CALL GSCHH(1.5)
            CALL GSTXCI(ORANGE)
C DONE
            IF(SEG.EQ.7) THEN
            CALL GSTXCI(DKBLUE)
            CALL GTX( 2.5, 35.0, 'SOLID ')
            CALL GTX( 2.0, 31.0, 'DISPL.')
            CALL GTX( 12.0, 37.0, ' WIRE ')
            CALL GTX( 12.5, 34.0, 'FRAME ')
            CALL GTX( 12.5, 31.0, 'DISPL.')
            CALL GTX(22.0, 37.0, ' SUPER ')
            CALL GTX(22.0, 34.0, 'IMPOSED')
            CALL GTX(22.5, 31.0, 'DISPL.')
            CALL GTX( 22.0, 15.0, 'GROUPS')
            CALL GTX( 22.0, 11.0, 'ON/OFF')
            CALL GTX( 2.0, 14.0, 'DONE')
            XG=0
            RETURN
C GROUP CONTROL
            ELSEIF(SEG.EQ.9) THEN
             CALL GTX(2.0,88.0,'ENTER GROUP NUMBER')
             CALL GTX(2.0,85.0,' TO TOGGLE ON/OFF ')
             CALL GRQST(1,1,I,J,DUM)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(DUM,71,ERR=99) NPRESG
             IF(NPRESG.LT.1.OR.NPRESG.GT.NOG) GOTO 88
             NOGAC(NPRESG)=-NOGAC(NPRESG)
C META DUMP
            ELSEIF(SEG.EQ.BMETA) THEN
             CALL META()
C EXAGERATED DISP SOLID
            ELSEIF(SEG.EQ.1) THEN
             IPLAN=1
             CALL GTX(2.0,85.0,'ENTER EXAG FACTOR')
             CALL GRQST(1,1,I,J,DUM)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(DUM,70,ERR=99) EXAG
             CALL ROTATE(0.0,0.0)
             XG=2
             CSC=1
             KSOW=1

             CALL DRAW(1,'  ',0)

C EXAGERATED DISP WIRE
            ELSEIF(SEG.EQ.2) THEN
             IPLAN=2
             CALL GTX(2.0,85.0,'ENTER EXAG FACTOR')
             CALL GRQST(1,1,I,J,DUM)
             CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
             READ(DUM,70,ERR=99) EXAG
             CALL ROTATE(0.0,0.0)
             XG=2
             CSC=1
             KSOW=0

             CALL DRAW(1,'  ',0)

C SUPER IMPOSED WIRE OVER SOLID
            ELSEIF(SEG.EQ.3) THEN
             IPLAN=3
```

```
                CALL GTX(2.0,85.0,'ENTER EXAG FACTOR')
                CALL GRQST(1,1,I,J,DUM)
                CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                READ(DUM,70,ERR=99) EXAG
                CALL ROTATE(0.0,0.0)
                XG=0
                CSC=1
                KSOW=1

                CALL DRAW(1,'  ',0)
                XG=2
                CSC=0
                KSOW=0
                SEG=3
                CALL DRAW(1,'  ',0)

               END IF
               GOTO 5
70  FORMAT(F10.3)
71  FORMAT(I2)
88    CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                CALL ERROR(104)
                GOTO 5

C
C ERROR HANDLER
C
99    CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                CALL ERROR(105)
                GOTO 5
                END
C
C************************************************************
**************
C
                SUBROUTINE PICKSTRESS()
                INCLUDE 'COMMON.F'
                CHARACTER*10 DUM

C
C THIS SUBROUTINE  i) ALLOWS PICKING OF STRESSES
C                  TO BE SHOWN ON THE SCREEN
C                 ii) METAFILE PLOTS OF STRESSES
C                iii) TOGGLE GROUPS ON OFF
C
C     BUF.....CHARACTER BUFFER THAT ALL INPUT IS READ
INTO
C
C     WRITING BUTTON NAMES
C
                CALL GSTXCI(GREEN)
C 1
                CALL GTX( 2.5, 35.0, 'STRESS')
                CALL GTX( 3.0, 31.0, ' X  ')
C 2
                CALL GTX(12.5, 35.0, 'STRESS')
                CALL GTX(13.0, 31.0, ' Y  ')
C 3
                CALL GTX(22.5, 35.0, 'STRESS')
                CALL GTX(23.0, 31.0, ' Z  ')
C 4
                CALL GTX( 2.5, 25.0, 'STRESS')
                CALL GTX( 3.0, 21.0, ' XY ')
C 5
                CALL GTX(12.5, 25.0, 'STRESS')
                CALL GTX(13.0, 21.0, ' YZ ')
C 6
                CALL GTX(22.5, 25.0, 'STRESS')
                CALL GTX(23.0, 21.0, ' ZX ')
C 8
                CALL GTX(12.0, 11.0, 'STRESS')
                CALL GTX(10.5, 15.0, 'Von-Mises')
C 9
                CALL GTX( 22.0, 15.0, 'GROUPS')
                CALL GTX( 22.0, 11.0, 'ON/OFF')
                CALL GSTXCI(14)
C 7
                CALL GTX( 2.0, 14.0, 'DONE')
                CALL GSTXCI(GREEN)
                IPLAN=0
C
C  MAIN PICK
C
5     CALL GRQPK(1,2,II,SEG,JJ)
                CALL GSCHH(1.5)
                CALL GSTXCI(GREEN)
C DONE
                IF(SEG.EQ.7) THEN
                CALL GSTXCI(DKBLUE)
                CALL GTX( 2.5, 35.0, 'STRESS')
                CALL GTX( 3.0, 31.0, ' X  ')
                CALL GTX(12.5, 35.0, 'STRESS')
                CALL GTX(13.0, 31.0, ' Y  ')
                CALL GTX(22.5, 35.0, 'STRESS')
                CALL GTX(23.0, 31.0, ' Z  ')
                CALL GTX( 2.5, 25.0, 'STRESS')
                CALL GTX( 3.0, 21.0, ' XY ')
                CALL GTX(12.5, 25.0, 'STRESS')
                CALL GTX(13.0, 21.0, ' YZ ')
                CALL GTX(22.5, 25.0, 'STRESS')
                CALL GTX(23.0, 21.0, ' ZX ')
                CALL GTX(12.0, 11.0, 'STRESS')
                CALL GTX(10.5, 15.0, 'Von-Mises')
                CALL GTX( 22.0, 15.0, 'GROUPS')
                CALL GTX( 22.0, 11.0, 'ON/OFF')
                CALL GTX( 2.0, 14.0, 'DONE')
                RETURN
C GROUP CONTROL
                ELSEIF(SEG.EQ.9) THEN
                 CALL GTX(2.0,88.0,'ENTER GROUP NUMBER')
                 CALL GTX(2.0,85.0,' TO TOGGLE ON/OFF ')
                 CALL GRQST(1,1,I,J,DUM)
                 CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                 READ(DUM,71,ERR=99) NPRESG
                 IF(NPRESG.LT.1.OR.NPRESG.GT.NOG) GOTO 88
                 NOGAC(NPRESG)=-NOGAC(NPRESG)

C META DUMP
                ELSEIF(SEG.EQ.BMETA) THEN
                 CALL META()
C STRESS X
                ELSEIF(SEG.EQ.1) THEN
                 IPLAN=4

                 CALL CONT(1,'  ')
C STRESS Y
                ELSEIF(SEG.EQ.2) THEN
                 IPLAN=5

                 CALL CONT(1,'  ')
C STRESS Y
                ELSEIF(SEG.EQ.3) THEN
                 IPLAN=6

                 CALL CONT(1,'  ')
C STRESS XY
                ELSEIF(SEG.EQ.4) THEN
                 IPLAN=7

                 CALL CONT(1,'  ')
C STRESS YZ
                ELSEIF(SEG.EQ.5) THEN
                 IPLAN=8

                 CALL CONT(1,'  ')
C STRESS ZX
                ELSEIF(SEG.EQ.6) THEN
                 IPLAN=9

                 CALL CONT(1,'  ')
C STRESS EQUIVELENT
                ELSEIF(SEG.EQ.8) THEN
                 IPLAN=10

                 CALL CONT(1,'  ')

                END IF
                GOTO 5


88    CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                CALL ERROR(104)
                CALL GSTXCI(ORANGE)
                GOTO 5

C
C ERROR HANDLER
C
99    CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
                CALL ERROR(105)
                CALL GSTXCI(ORANGE)
                GOTO 5
70  FORMAT(F10.3)
71  FORMAT(I2)

                END
C
C************************************************************
**************
C
```

```
      SUBROUTINE CONT(IKJ,BUF)
C
C     THIS ROUTINE DRAWS THE STRESS CONTOUR GRAPHICS
C
C     IKJ.... = 0 FOR META DRAW, 1 FOR SCREEN
C     BUF.... IS TITLE FOR META DRAW
C
C
      CHARACTER*20 BUF
      INTEGER IKJ,CCN,CA,CNN
      INCLUDE 'COMMON.F
      DIMENSION CA(10),CNN(4)
      CA(1)=12
      CA(2)=7
      CA(3)=9
      CA(4)=4
      CA(5)=8
      CA(6)=3
      CA(7)=5
      CA(8)=10
      CA(9)=2
      CA(10)=11
      CCN=10
      IF(CORG.EQ.0) THEN
C GREY SCALE
       CALL GSCR(2,12,0.05,0.05,0.05)
       CALL GSCR(2, 7,0.15,0.15,0.15)
       CALL GSCR(2, 9,0.25,0.25,0.25)
       CALL GSCR(2, 4,0.35,0.35,0.35)
       CALL GSCR(2, 8,0.55,0.55,0.55)
       CALL GSCR(2, 3,0.65,0.65,0.65)
       CALL GSCR(2, 5,0.75,0.75,0.75)
       CALL GSCR(2,10,0.85,0.85,0.85)
       CALL GSCR(2, 2,0.95,0.95,0.95)
       CALL GSCR(2,11,1.0 ,1.0 ,1.0 )

      ELSE
C COLOUR
      CALL GSCR(2,2,1.0,0.0,0.0)
      CALL GSCR(2,7,0.5,0.0,1.0)
      CALL GSCR(2,3,0.0,1.0,0.0)
      CALL GSCR(2,0,0.0,0.0,0.0)
      CALL GSCR(2,9,0.0,0.0,1.0)
      CALL GSCR(2,4,0.0,0.0,0.5)
      CALL GSCR(2,1,1.0,1.0,1.0)
      CALL GSCR(2,6,0.5,0.5,0.5)
      CALL GSCR(2,8,0.0,0.5,0.0)
      CALL GSCR(2,11,0.5,0.0,0.0)
      CALL GSCR(2,5,1.0,1.0,0.0)
      CALL GSCR(2,10,1.0,0.5,0.0)
      CALL GSCR(2,12,1.0,0.0,1.0)

      END IF

C
C GOING TO TRANSFORMATION #1
C
      CALL GSELNT (1)
      CALL GSLWSC(0.5)
C
C CODE FOR METAFILE OUTPUT OR SCREEN DRAW
C
      IF(IKJ.EQ.0) THEN
        CALL GSTXCI(0)
        CALL GSPLCI(0)
        CALL GTX(XMN+0.37*XDIS,YMN+0.37*YDIS,BUF)
      ELSE
       CALL GSTXCI(1)
       CALL GSPLCI(6)
      END IF
       CALL GSCHH(YDIS/55.0)
       KSOW=1
       NSTS=IPLAN-3
       YDIS=YMX-YMN
       CALL GSLWSC(0.5)
C
C clear screen
      call box(xmn,xmx,ymn,ymx,0,1)
C
C *** PLOTTING ROUTINE ***
C
C SEARCH FOR MAX
C
      SMAX=-10.0E13
      SMIN=+10.0E13
      DO 3 I=1,NOE
      DO 3 J=1,NUMNOD(NOETY(NEC(I,9)))
      IF(STS(I,J,NSTS).GT.SMAX) SMAX=STS(I,J,NSTS)
      IF(STS(I,J,NSTS).LT.SMIN) SMIN=STS(I,J,NSTS)
3     CONTINUE
      CALL GSFAIS(1)

C
C SET SCALE
C
      SSCA=(SMAX-SMIN)/FLOAT(CCN)
      IF(SSCA.EQ.0.0) CCN=1
C
C LOOP THRU ELEMENTS
C
C
C     LOOP THRU ELEMENTS
      DO 5 JJ=1,NOE
      I=NOOSE(JJ)
C CHECK IF ELEMENT IS ACTIVE
      IF(IEF(I).NE.0) THEN
C CHECK IF ELEMENTS GROUP IS ACTIVE
      IF(NOGAC(NEC(I,9)).EQ.1) THEN
C CHECK IF BRICK
      IF(NOETY(NEC(I,9)).EQ.8) THEN
C
C VIEW POINT
      XV=100000.0*COS(AS)*COS(AP)
      YV=100000.0*SIN(AS)
      ZV=100000.0*COS(AS)*SIN(AP)
      DO 15 KK=1,3
       IF(KK.EQ.1) THEN
C TOP BOTTOM PICK

SX=PN(NEC(I,1),1)+PN(NEC(I,2),1)+PN(NEC(I,3),1)+PN(NEC(I,4),1)/4.

SY=PN(NEC(I,1),2)+PN(NEC(I,2),2)+PN(NEC(I,3),2)+PN(NEC(I,4),2)/4.

SZ=PN(NEC(I,1),3)+PN(NEC(I,2),3)+PN(NEC(I,3),3)+PN(NEC(I,4),3)/4.
      EDTVPA= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(I,5),1)+PN(NEC(I,6),1)+PN(NEC(I,7),1)+PN(NEC(I,8),1)/4.

SY=PN(NEC(I,5),2)+PN(NEC(I,6),2)+PN(NEC(I,7),2)+PN(NEC(I,8),2)/4.

SZ=PN(NEC(I,5),3)+PN(NEC(I,6),3)+PN(NEC(I,7),3)+PN(NEC(I,8),3)/4.
      EDTVPB= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
    IF(EDTVPA.LT.EDTVPB) THEN
       CNN(1)=1
       CNN(2)=2
       CNN(3)=3
       CNN(4)=4
      ELSE
       CNN(1)=5
       CNN(2)=6
       CNN(3)=7
       CNN(4)=8
      END IF
       ELSEIF(KK.EQ.2) THEN
C LEFT RIGHT PICK

SX=PN(NEC(I,1),1)+PN(NEC(I,2),1)+PN(NEC(I,6),1)+PN(NEC(I,5),1)/4.

SY=PN(NEC(I,1),2)+PN(NEC(I,2),2)+PN(NEC(I,6),2)+PN(NEC(I,5),2)/4.

SZ=PN(NEC(I,1),3)+PN(NEC(I,2),3)+PN(NEC(I,6),3)+PN(NEC(I,5),3)/4.
      EDTVPA= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(I,4),1)+PN(NEC(I,3),1)+PN(NEC(I,8),1)+PN(NEC(I,7),1)/4.

SY=PN(NEC(I,4),2)+PN(NEC(I,3),2)+PN(NEC(I,8),2)+PN(NEC(I,7),2)/4.

SZ=PN(NEC(I,4),3)+PN(NEC(I,3),3)+PN(NEC(I,8),3)+PN(NEC(I,7),3)/4.
      EDTVPB= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
    IF(EDTVPA.LT.EDTVPB) THEN
       CNN(1)=1
       CNN(2)=2
       CNN(3)=6
       CNN(4)=5
      ELSE
       CNN(1)=3
       CNN(2)=4
       CNN(3)=8
       CNN(4)=7
       END IF
       ELSEIF(KK.EQ.3) THEN
C FRONT BACK PICK

SX=PN(NEC(I,1),1)+PN(NEC(I,4),1)+PN(NEC(I,8),1)+PN(NEC(I,5),1)/4.

SY=PN(NEC(I,1),2)+PN(NEC(I,4),2)+PN(NEC(I,8),2)+PN(NEC(I,5),2)/4.
```

```
SZ=PN(NEC(I,1),3)+PN(NEC(I,4),3)+PN(NEC(I,8),3)+PN(NEC(I,5),3)/4.
              EDTVPA= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(I,2),1)+PN(NEC(I,3),1)+PN(NEC(I,7),1)+PN(NEC(I,6),1)/4.

SY=PN(NEC(I,2),2)+PN(NEC(I,3),2)+PN(NEC(I,7),2)+PN(NEC(I,6),2)/4.

SZ=PN(NEC(I,2),3)+PN(NEC(I,3),3)+PN(NEC(I,7),3)+PN(NEC(I,6),3)/4.
              EDTVPB= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
               IF(EDTVPA.LT.EDTVPB) THEN
                CNN(1)=1
                CNN(2)=4
                CNN(3)=8
                CNN(4)=5
               ELSE
                CNN(1)=2
                CNN(2)=3
                CNN(3)=7
                CNN(4)=6
               END IF
               END IF
C
C END OF SIDE SELECTION
C
C PLOTTING CONTOURS
C
              DO 161 J=1,CCN
              CALL GSFACI(CA(J))
              K=0
              SH=SMIN + SSCA*FLOAT(J)
              SL=SH-SSCA
C
C LOOP THRU SIDES OF ELEMENT
C
              DO 171 L=1,4
C SET NODE NUMBERS
               NA=NEC(I,CNN(L))
               PA=STS(I,CNN(L),NSTS)
               IF(L.EQ.4) THEN
                 NB=NEC(I,CNN(1))
                 PB=STS(I,CNN(1),NSTS)
               ELSE
                 NB=NEC(I,CNN(L+1))
                 PB=STS(I,CNN(L+1),NSTS)
               END IF
C
C IF STRUCTURE FOR FINDING VALUES
C
C IF BOTH OUT OF RANGE
              IF(PA.LT.SL.AND.PB.LT.SL) THEN

              ELSE IF(PB.GT.SH.AND.PA.GT.SH) THEN

C IF BOTH IN RANGE
              ELSE
IF(PA.LE.SH.AND.PA.GE.SL.AND.PB.LE.SH.AND.PB.GE.SL) THEN

               K=K+1
               X(K)=PN(NA,6)
               Y(K)=PN(NA,7)
C IF THRU RANGE 1
              ELSE IF(PA.LT.SL.AND.PB.GT.SH) THEN

               K=K+1
               CALL INPL(PA,SL,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
               K=K+1
               CALL INPL(PA,SH,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
C IF THRU RANGE 2
              ELSE IF(PB.LT.SL.AND.PA.GT.SH) THEN

               K=K+1
               CALL INPL(PA,SH,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
               K=K+1
               CALL INPL(PA,SL,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
C IF PA IN BUT PB OUT
              ELSE IF(PA.GE.SL.AND.PA.LE.SH) THEN

               K=K+1
               X(K)=PN(NA,6)
               Y(K)=PN(NA,7)
                ST=SL
                IF(PB.GE.SH) ST=SH
                K=K+1
                CALL INPL(PA,ST,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))

C IF PB IN BUT PA OUT
              ELSE IF(PB.GE.SL.AND.PB.LE.SH) THEN

               K=K+1
                ST=SL
                IF(PA.GE.SH) ST=SH
               CALL INPL(PA,ST,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
              END IF
C     NEXT EDGE OF ELEMENT
171    CONTINUE
C     INSURE CLOSED POLYGON??
               IF(K.GT.0.AND.K.LE.9) THEN
                K=K+1
                X(K)=X(1)
                Y(K)=Y(1)
                END IF
C
C PLOTTING A CONTOUR
C
              IF(K.GT.0) CALL GFA(K,X,Y)
              K=0
C     NEXT CONTOUR
161   CONTINUE
C
C WHITE OUTLINE OF ELEMENT
C
              X(1)=PN(NEC(I,CNN(1)),4)
              X(2)=PN(NEC(I,CNN(2)),4)
              X(3)=PN(NEC(I,CNN(3)),4)
              X(4)=PN(NEC(I,CNN(4)),4)
              X(5)=X(1)
              Y(1)=PN(NEC(I,CNN(1)),5)
              Y(2)=PN(NEC(I,CNN(2)),5)
              Y(3)=PN(NEC(I,CNN(3)),5)
              Y(4)=PN(NEC(I,CNN(4)),5)
              Y(5)=Y(1)
              CALL GSPLCI(WHITE)
              CALL GPL(5,X,Y)
C
C NEXT SIDE
C
15    CONTINUE
C
C END OF BRICK
C
C CHECK IF 3D BEAM
              ELSE IF(NOETY(NEC(I,9)).EQ.6) THEN
C
C LOOP THRU TOP HALF THEN BOTTOM HALF
C
              DO 17 KK=1,2
              IF(KK.EQ.1) THEN
               CNN(1)=1
               CNN(2)=2
               CNN(3)=5
               CNN(4)=6
              ELSE
               CNN(1)=3
               CNN(2)=2
               CNN(3)=5
               CNN(4)=4
              END IF
C
C PLOTTING CONTOURS
C
              DO 163 J=1,CCN
              CALL GSFACI(CA(J))
              K=0
              SH=SMIN + SSCA*FLOAT(J)
              SL=SH-SSCA
C
C LOOP THRU SIDES OF ELEMENT
C
              DO 173 L=1,4
C SET NODE NUMBERS
               NA=NEC(I,CNN(L))
               PA=STS(I,CNN(L),NSTS)
               IF(L.EQ.4) THEN
                 NB=NEC(I,CNN(1))
                 PB=STS(I,CNN(1),NSTS)
               ELSE
                 NB=NEC(I,CNN(L+1))
                 PB=STS(I,CNN(L+1),NSTS)
               END IF
C
C IF STRUCTURE FOR FINDING VALUES
```

```
C
C IF BOTH OUT OF RANGE
              IF(PA.LT.SL.AND.PB.LT.SL) THEN

              ELSE IF(PB.GT.SH.AND.PA.GT.SH) THEN

C IF BOTH IN RANGE
              ELSE
IF(PA.LE.SH.AND.PA.GE.SL.AND.PB.LE.SH.AND.PB.GE.SL) THEN

               K=K+1
               X(K)=PN(NA,6)
               Y(K)=PN(NA,7)
C IF THRU RANGE 1
              ELSE IF(PA.LT.SL.AND.PB.GT.SH) THEN

               K=K+1
               CALL INPL(PA,SL,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
               K=K+1
               CALL INPL(PA,SH,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
C IF THRU RANGE 2
              ELSE IF(PB.LT.SL.AND.PA.GT.SH) THEN

               K=K+1
               CALL INPL(PA,SH,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
               K=K+1
               CALL INPL(PA,SL,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
C IF PA IN BUT PB OUT
              ELSE IF(PA.GE.SL.AND.PA.LE.SH) THEN

               K=K+1
               X(K)=PN(NA,6)
               Y(K)=PN(NA,7)
                ST=SL
                IF(PB.GE.SH) ST=SH
                K=K+1
               CALL INPL(PA,ST,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))

C IF PB IN BUT PA OUT
              ELSE IF(PB.GE.SL.AND.PB.LE.SH) THEN

               K=K+1
                ST=SL
                IF(PA.GE.SH) ST=SH
               CALL INPL(PA,ST,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
              END IF
C     NEXT EDGE OF ELEMENT
173   CONTINUE
C     INSURE CLOSED POLYGON??
               IF(K.GT.0.AND.K.LE.9) THEN
                K=K+1
                X(K)=X(1)
                Y(K)=Y(1)
                END IF
C
C PLOTTING A CONTOUR
C
              IF(K.GT.0) CALL GFA(K,X,Y)
              K=0
C     NEXT CONTOUR
163   CONTINUE
C
C NEXT HALF
17    CONTINUE
C
C WHITE OUTLINE OF ELEMENT
C
              X(1)=PN(NEC(I,1),6)
              X(2)=PN(NEC(I,3),6)
              X(3)=PN(NEC(I,4),6)
              X(4)=PN(NEC(I,6),6)
              X(5)=X(1)
              Y(1)=PN(NEC(I,1),7)
              Y(2)=PN(NEC(I,3),7)
              Y(3)=PN(NEC(I,4),7)
              Y(4)=PN(NEC(I,6),7)
              Y(5)=Y(1)
              CALL GSPLCI(WHITE)
              CALL GPL(5,X,Y)
C
C END OF 3D BEAM
C
C NOT A BRICK OR 3D BEAM
C
              ELSE

              DO 160 J=1,CCN
              CALL GSFACI(CA(J))
              K=0
              SH=SMIN + SSCA*FLOAT(J)
              SL=SH-SSCA
C
C LOOP THRU SIDES OF ELEMENT
C
              DO 170 L=1,NUMNOD(NOETY(NEC(I,9)))
C SET NODE NUMBERS
               NA=NEC(I,L)
               PA=STS(I,L,NSTS)
               IF(L.EQ.NUMNOD(NOETY(NEC(I,9)))) THEN
                NB=NEC(I,1)
                PB=STS(I,1,NSTS)
               ELSE
                NB=NEC(I,L+1)
                PB=STS(I,L+1,NSTS)
               END IF
C
C IF STRUCTURE FOR FINDING VALUES
C
C IF BOTH OUT OF RANGE
              IF(PA.LT.SL.AND.PB.LT.SL) THEN

              ELSE IF(PB.GT.SH.AND.PA.GT.SH) THEN

C IF BOTH IN RANGE
              ELSE
IF(PA.LE.SH.AND.PA.GE.SL.AND.PB.LE.SH.AND.PB.GE.SL) THEN

               K=K+1
               X(K)=PN(NA,6)
               Y(K)=PN(NA,7)
C IF THRU RANGE 1
              ELSE IF(PA.LT.SL.AND.PB.GT.SH) THEN

               K=K+1
               CALL INPL(PA,SL,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
               K=K+1
               CALL INPL(PA,SH,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
C IF THRU RANGE 2
              ELSE IF(PB.LT.SL.AND.PA.GT.SH) THEN

               K=K+1
               CALL INPL(PA,SH,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
               K=K+1
               CALL INPL(PA,SL,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
C IF PA IN BUT PB OUT
              ELSE IF(PA.GE.SL.AND.PA.LE.SH) THEN

               K=K+1
               X(K)=PN(NA,6)
               Y(K)=PN(NA,7)
                ST=SL
                IF(PB.GE.SH) ST=SH
                K=K+1
               CALL INPL(PA,ST,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))

C IF PB IN BUT PA OUT
              ELSE IF(PB.GE.SL.AND.PB.LE.SH) THEN

               K=K+1
                ST=SL
                IF(PA.GE.SH) ST=SH
               CALL INPL(PA,ST,PB,X,Y,K,PN(NA,6),PN(NA,7),
     &                   PN(NB,6),PN(NB,7))
              END IF
C     NEXT EDGE OF ELEMENT
170   CONTINUE
C     INSURE CLOSED POLYGON??
               IF(K.GT.0.AND.K.LE.9) THEN
                K=K+1
                X(K)=X(1)
                Y(K)=Y(1)
                END IF
C
C PLOTTING A CONTOUR
C
              IF(K.GT.0) CALL GFA(K,X,Y)
C     IF(K.GT.0) CALL GPL(K,X,Y)
              K=0
C     NEXT CONTOUR
```

```
160   CONTINUE
C  OUTLINE OF ELEMENT
              X(1)=PN(NEC(I,1),6)
              X(2)=PN(NEC(I,2),6)
              X(3)=PN(NEC(I,3),6)
              X(4)=PN(NEC(I,4),6)
              X(5)=X(1)
              Y(1)=PN(NEC(I,1),7)
              Y(2)=PN(NEC(I,2),7)
              Y(3)=PN(NEC(I,3),7)
              Y(4)=PN(NEC(I,4),7)
              Y(5)=Y(1)
              CALL GSPLCI(WHITE)
              CALL GPL(5,X,Y)

               END IF
              END IF
             END IF
5     CONTINUE
C
C DRAW LEGEND
C
              KSOW=1
C SET X COORDS
              XW=XMN+0.8*XDIS
              X(1)=XMN+0.9*XDIS
              X(2)=XMN+0.95*XDIS
              X(3)=X(2)
              X(4)=X(1)
              X(5)=X(1)
C
              CALL GSCHH(YDIS/4.5/FLOAT(CCN))
              WRITE(BUF,199) SMIN
              CALL GTX( XW , (YMN+0.05*YDIS) ,BUF)
              DO 180 I=1,CCN
              Y(3)=YMN+0.05*YDIS+FLOAT(I)*YDIS/FLOAT(CCN)/
1.15
              Y(1)=Y(3)-YDIS/2.0/FLOAT(CCN)/1.15
              Y(2)=Y(1)
              Y(4)=Y(3)
              Y(5)=Y(1)
              WRITE(BUF,199) (SMIN+I*SSCA)
              CALL GTX( XW, (Y(3)+0.01*YDIS), BUF)
              CALL GSFACI(CA(I))
              CALL GFA(5,X,Y)
180   CALL GPL(5,X,Y)
              IF(NSTS.EQ.1) THEN
              BUF=' SX '
              ELSEIF(NSTS.EQ.2) THEN
              BUF=' SY '
              ELSEIF(NSTS.EQ.3) THEN
              BUF=' SZ'
              ELSEIF(NSTS.EQ.4) THEN
              BUF=' SXY'
              ELSEIF(NSTS.EQ.5) THEN
              BUF=' SYZ'
              ELSEIF(NSTS.EQ.6) THEN
              BUF=' SZX'
              ELSEIF(NSTS.EQ.7) THEN
              BUF=' SEQ'
              END IF
              CALL GTX( XW , (YMN+0.01*YDIS) ,BUF)

C
C BOUNDARY
              CALL BOX(xmn,xmx,ymn,ymx,DKBLUE,0)
C
C
              CALL GSLWSC(3.0)
              CALL GSPLCI(PURPLE)
C
C END OF ROUTINE
              CALL GSELNT (2)
              CALL GSCHH(1.50)
              RETURN
199   FORMAT(e12.4)
              END
C
C*************************************************************
*****
C
    SUBROUTINE INPL(PA,SV,PB,X,Y,K,XA,YA,XB,YB)
C
C THIS SUBROUTINE IS A LINEAR INTERPOLATER FINDER
C
C PA AND PB ARE VALUES
C SV IS THE SEARCH VALUE
C X & Y ARE THE RESULTING VALUES
C XA,YA ARE VALUES FOR PA
C XB,YB ARE VALUES FOR PB
C
              REAL X(10),Y(10),XA,XB,YA,YB,PA,SV,PB
              X(K)=(XA+(XB-XA)*(SV-PA)/(PB-PA))
              Y(K)=(YA+(YB-YA)*(SV-PA)/(PB-PA))
              RETURN
              END

C
C*************************************************************
**************
C
              SUBROUTINE META()
              CHARACTER*20 BUF,DUM
              INCLUDE 'COMMON.F'
C
C THIS SUBROUTINE GENERATES A COMPUTER GRAPHICS
METAFILE
C
C INPUT TITLE OF THIS PLOT
C
              IF(IPLAN.EQ.0) RETURN
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               CALL GSTXCI(GREEN)
               CALL GTX(2.0,88.0,' ENTER TITLE FOR ')
               CALL GTX(2.0,85.0,'THIS GRAPHIC PLOT')
               CALL GRQST(1,1,K,J,BUF)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
C
C COLOR OR GREY SCALE
C
               CALL GSTXCI(GREEN)
               CALL GTX(1.5,88.0,' COLOR (C) OR ')
               CALL GTX(1.5,85.0,'GREY SCALE (G) ?')
               CALL GRQST(1,1,K,J,DUM)
               CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
               CORG=0
               IF('C'.EQ.DUM.OR.'c'.EQ.DUM) CORG=1
C
C DEACTIVATING SCREEN WORKSTATION
C ACTIVATING METAFILE WORKSTATION
C
              CALL GACWK (2)
              CALL GDAWK (1)
              XMN=XMN-1.5*XDIS
              XMX=XMX+1.5*XDIS
              XDIS=XMX-XMN
              YMN=YMN-1.5*YDIS
              YMX=YMX+1.5*YDIS
              YDIS=YMX-YMN
              CALL GSWN (1,XMN,XMX,YMN,YMX)
C
C DRAW TO METAFILE
C

C
C IPLAN=1.....SOLID DISPLACEMENT
C IPLAN=2.....WIRE FRAME DISPLACEMENT
C IPLAN=3.....SUPERIMPOSED DISPLACEMENT
C IPLAN=4.....SX  \
C IPLAN=5.....SY  \\\
C IPLAN=6.....SZ  \\\\\
C IPLAN=7.....SXY  >>>>>> STRESS COUNTOURS
C IPLAN=8.....SYZ  /////
C IPLAN=9.....SZX  ///
C IPLAN=10....SEQ  /
C
C MAIN CALLER ROUTINE
C
             IF(IPLAN.EQ.1) THEN
              XG=2
              KSOW=1

              CALL DRAW(0,BUF,0)
             ELSEIF(IPLAN.EQ.2) THEN
              XG=2
              KSOW=0

              CALL DRAW(0,BUF,0)
             ELSEIF(IPLAN.EQ.3) THEN
              XG=0
              KSOW=1

              CALL DRAW(0,BUF,0)
              XG=2
              KSOW=0
              CALL DRAW(0,BUF,0)
             ELSEIF(IPLAN.GE.4.AND.IPLAN.LE.10) THEN

              CALL CONT(0,BUF)
```

```
                END IF
C
C TURN METAFILE OFF / TURN SCREEN ON
C
                CALL GDAWK (2)
                CALL GACWK (1)
C
                RETURN
                END

C
C**********************************************************
**************
C
                SUBROUTINE DRAW(IKJ,BUF,ISORT)
C
C     THIS ROUTINE DRAWS THE GRAPHICS
C
C     IKJ.... = 0 FOR META DRAW, 1 FOR SCREEN
C     BUF.... IS TITLE FOR META DRAW
C     ISORT.. 1=SORT, 0=NO SORT
C     CF..... CORRECTION FACTOR FOR # PLACEMENT
(META/SCREEN)
C     FMAX....MAXIMUM FORCE
C     MMAX....MAXIMUM MOMENT
C     KSOW....FLAG SOLID OR WIREFRAME 0=WIRE,1=SOLID
C     XG......EXAGGERATED=2, NOT=0
C     CSC.....CLEAR=1,NOT=0
C
                CHARACTER*20 BUF
                INTEGER IKJ,ISORT
                REAL CP,CS,SS,SP,FMAX,MMAX,CF
                INCLUDE 'COMMON.F
C
C GOING TO TRANSFORMATION #1
C
                CALL GSELNT (1)
                CP=COS(AP)
                SP=SIN(AP)
                CS=COS(AS)
                SS=SIN(AS)
                CALL GSLWSC(0.5)
C
C CODE FOR METAFILE OUTPUT OR SCREEN DRAW
C
                IF(IKJ.EQ.0) THEN
                 CF=0.0
                 CALL GSTXCI(BLACK)
                 CALL GTX(XMN+0.37*XDIS,YMN+0.37*YDIS,BUF)
                 CALL GSPMCI(BLACK)
                ELSE
                 CF=0.12
                 CALL GSTXCI(WHITE)
                 CALL GSPMCI(YELLOW)
                END IF
                 CALL GSCHH(YDIS/55.0)
                 IF(CSC.EQ.0) CALL GSLN(3)
C
C REMAKING SCREEN AS SEGMENT
C
                IF(CSC.EQ.1) THEN
                CALL box(xmn,xmx,ymn,ymx,0,1)
                END IF
C
C CALLING SORT
C
                IF(ISORT.EQ.1) CALL SORT()
C
C SET SOLID OR WIREFRAME
                CALL GSFAIS(KSOW)
C IF ELEMENTS ACTIVE...
   IF(KES.EQ.1) THEN
C LOOP THRU ELEMENTS
                DO 5 JJ=1,NOE
                J=NOOSE(JJ)
C CHECK IF ELEMENT IS ACTIVE
                IF(IEF(J).NE.0) THEN
C CHECK IF ELEMENTS GROUP IS ACTIVE
                IF(NOGAC(NEC(J,9)).EQ.1) THEN
C GOTO APPROPRIATE PLOTTING CODE
                 GOTO
(10,10,10,10,10,20,30,40,10,30,30,30),NOETY(NEC(J,9))
C          1 2 3 4 5 6 7 8 9 10 11 12
C 3D SPRING, 2D TRUSS.BEAM.& GRID BEAM, 3D TRUSS.BEAM
10    X(1)=PN(NEC(J,1),4+XG)
                X(2)=PN(NEC(J,2),4+XG)
                Y(1)=PN(NEC(J,1),5+XG)
                Y(2)=PN(NEC(J,2),5+XG)
                IF(CSC.EQ.0) THEN
C SUPERIMPOSED
                 CALL GSPLCI(WHITE)
                 CALL GPL(2,X,Y)
                ELSE
                 CALL GSPLCI( NUMCOL(NOETY(NEC(J,9))) )
                 IF(CSC.EQ.0) CALL GSPLCI(WHITE)
                 CALL GPL(2,X,Y)
                END IF
C NUMBER
                IF(KEN.EQ.1) THEN
                  X(1)=(X(1)+X(2))/2.0
                  Y(1)=(Y(1)+Y(2))/2.0
                  X(1)=X(1)-CF*XDIS
                  WRITE(BUF,*) J
                  CALL GTX(X(1),Y(1),BUF)
                  END IF
                 GOTO 5
C
C #2 3D BEAM
20    X(1)=PN(NEC(J,1),4+XG)
                X(2)=PN(NEC(J,3),4+XG)
                X(3)=PN(NEC(J,4),4+XG)
                X(4)=PN(NEC(J,6),4+XG)
                Y(1)=PN(NEC(J,1),5+XG)
                Y(2)=PN(NEC(J,3),5+XG)
                Y(3)=PN(NEC(J,4),5+XG)
                Y(4)=PN(NEC(J,6),5+XG)
                Y(5)=Y(1)
                X(5)=X(1)
                IF(CSC.EQ.0) THEN
C SUPERIMPOSED
                 CALL GSPLCI(WHITE)
                 CALL GPL(5,X,Y)
                ELSE
                 IF(KSOW.EQ.0) THEN
C WIRE FRAME
                 CALL GSPLCI(8)
                 CALL GPL(5,X,Y)
                 ELSE
C SOLID
                 CALL GSFACI(8)
                 CALL GFA(4,X,Y)
                 CALL GSPLCI(WHITE)
                 CALL GPL(5,X,Y)
                 END IF
                END IF
C NUMBER
    IF(KEN.EQ.1) THEN
                 X(1)=(X(1)+X(2)+X(3)+X(4))/4.0
                 Y(1)=(Y(1)+Y(2)+Y(3)+Y(4))/4.0
                 X(1)=X(1)-CF*XDIS
                 WRITE(BUF,*) J
                 CALL GTX(X(1),Y(1),BUF)
                 END IF
                GOTO 5
C
C #3 3D PLATE - 2D PLANE STRAIN/STRESS - GRID PLATE
30    X(1)=PN(NEC(J,1),4+XG)
                X(2)=PN(NEC(J,2),4+XG)
                X(3)=PN(NEC(J,3),4+XG)
                X(4)=PN(NEC(J,4),4+XG)
                Y(1)=PN(NEC(J,1),5+XG)
                Y(2)=PN(NEC(J,2),5+XG)
                Y(3)=PN(NEC(J,3),5+XG)
                Y(4)=PN(NEC(J,4),5+XG)
                X(5)=X(1)
                Y(5)=Y(1)
                IF(CSC.EQ.0) THEN
C SUPERIMPOSED
                 CALL GSPLCI(WHITE)
                 CALL GPL(5,X,Y)
                ELSE
                 IF(KSOW.EQ.0) THEN
C WIRE FRAME
                 CALL GSPLCI( NUMCOL(NOETY(NEC(J,9))) )
                 CALL GPL(5,X,Y)
                 ELSE
C SOLID
                 CALL GSFACI( NUMCOL(NOETY(NEC(J,9))) )
                 CALL GFA(4,X,Y)
                 CALL GSPLCI(WHITE)
                 CALL GPL(5,X,Y)
                 END IF
                END IF
C NUMBER
                IF(KEN.EQ.1) THEN
                 X(1)=(X(1)+X(2)+X(3)+X(4))/4.0
                 X(1)=X(1)-CF*XDIS
                 Y(1)=(Y(1)+Y(2)+Y(3)+Y(4))/4.0
                 WRITE(BUF,*) J
                 CALL GTX(X(1),Y(1),BUF)
```

```
                END IF
                GOTO 5
C
C 3D BRICK
C
C VIEW POINT
40    XV=100000.0*COS(AS)*COS(AP)
                YV=100000.0*SIN(AS)
                ZV=100000.0*COS(AS)*SIN(AP)
C TOP BOTTOM PICK

SX=PN(NEC(J,1),1)+PN(NEC(J,2),1)+PN(NEC(J,3),1)+PN(NEC(J,4),1)/4
.

SY=PN(NEC(J,1),2)+PN(NEC(J,2),2)+PN(NEC(J,3),2)+PN(NEC(J,4),2)/4
.

SZ=PN(NEC(J,1),3)+PN(NEC(J,2),3)+PN(NEC(J,3),3)+PN(NEC(J,4),3)/4.
                EDTVPA=( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(J,5),1)+PN(NEC(J,6),1)+PN(NEC(J,7),1)+PN(NEC(J,8),1)/4
.

SY=PN(NEC(J,5),2)+PN(NEC(J,6),2)+PN(NEC(J,7),2)+PN(NEC(J,8),2)/4
.

SZ=PN(NEC(J,5),3)+PN(NEC(J,6),3)+PN(NEC(J,7),3)+PN(NEC(J,8),3)/4.
                EDTVPB=( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
    IF(EDTVPA.LT.EDTVPB) THEN
                X(1)=PN(NEC(J,1),4+XG)
                Y(1)=PN(NEC(J,1),5+XG)
                X(2)=PN(NEC(J,2),4+XG)
                Y(2)=PN(NEC(J,2),5+XG)
                X(3)=PN(NEC(J,3),4+XG)
                Y(3)=PN(NEC(J,3),5+XG)
                X(4)=PN(NEC(J,4),4+XG)
                Y(4)=PN(NEC(J,4),5+XG)
               ELSE
                X(1)=PN(NEC(J,5),4+XG)
                Y(1)=PN(NEC(J,5),5+XG)
                X(2)=PN(NEC(J,6),4+XG)
                Y(2)=PN(NEC(J,6),5+XG)
                X(3)=PN(NEC(J,7),4+XG)
                Y(3)=PN(NEC(J,7),5+XG)
                X(4)=PN(NEC(J,8),4+XG)
                Y(4)=PN(NEC(J,8),5+XG)
                END IF
               X(5)=X(1)
               Y(5)=Y(1)
               IF(CSC.EQ.0) THEN
C SUPERIMPOSED
                CALL GSPLCI(WHITE)
                CALL GPL(5,X,Y)
               ELSE
                IF(KSOW.EQ.0) THEN
C WIRE FRAME
                CALL GSPLCI(GREY)
                CALL GPL(5,X,Y)
               ELSE
C SOLID
                CALL GSFACI(GREY)
                CALL GFA(4,X,Y)
                CALL GSPLCI(WHITE)
                CALL GPL(5,X,Y)
                END IF
               END IF
C LEFT RIGHT PICK

SX=PN(NEC(J,1),1)+PN(NEC(J,2),1)+PN(NEC(J,6),1)+PN(NEC(J,5),1)/4
.

SY=PN(NEC(J,1),2)+PN(NEC(J,2),2)+PN(NEC(J,6),2)+PN(NEC(J,5),2)/4
.

SZ=PN(NEC(J,1),3)+PN(NEC(J,2),3)+PN(NEC(J,6),3)+PN(NEC(J,5),3)/4.
               EDTVPA=( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(J,4),1)+PN(NEC(J,3),1)+PN(NEC(J,8),1)+PN(NEC(J,7),1)/4
.

SY=PN(NEC(J,4),2)+PN(NEC(J,3),2)+PN(NEC(J,8),2)+PN(NEC(J,7),2)/4
.

SZ=PN(NEC(J,4),3)+PN(NEC(J,3),3)+PN(NEC(J,8),3)+PN(NEC(J,7),3)/4.
               EDTVPB=( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
    IF(EDTVPA.LT.EDTVPB) THEN
                X(1)=PN(NEC(J,1),4+XG)
                Y(1)=PN(NEC(J,1),5+XG)
                X(2)=PN(NEC(J,2),4+XG)
                Y(2)=PN(NEC(J,2),5+XG)
                X(3)=PN(NEC(J,6),4+XG)
                Y(3)=PN(NEC(J,6),5+XG)
                X(4)=PN(NEC(J,5),4+XG)
                Y(4)=PN(NEC(J,5),5+XG)
               ELSE
                X(1)=PN(NEC(J,3),4+XG)
                Y(1)=PN(NEC(J,3),5+XG)
                X(2)=PN(NEC(J,4),4+XG)
                Y(2)=PN(NEC(J,4),5+XG)
                X(3)=PN(NEC(J,8),4+XG)
                Y(3)=PN(NEC(J,8),5+XG)
                X(4)=PN(NEC(J,7),4+XG)
                Y(4)=PN(NEC(J,7),5+XG)
                END IF
               X(5)=X(1)
               Y(5)=Y(1)
               IF(CSC.EQ.0) THEN
C SUPERIMPOSED
                CALL GSPLCI(WHITE)
                CALL GPL(5,X,Y)
               ELSE
                IF(KSOW.EQ.0) THEN
C WIRE FRAME
                CALL GSPLCI(GREY)
                CALL GPL(5,X,Y)
               ELSE
C SOLID
                CALL GSFACI(GREY)
                CALL GFA(4,X,Y)
                CALL GSPLCI(WHITE)
                CALL GPL(5,X,Y)
                END IF
               END IF
C FRONT BACK PICK

SX=PN(NEC(J,1),1)+PN(NEC(J,4),1)+PN(NEC(J,8),1)+PN(NEC(J,5),1)/4
.

SY=PN(NEC(J,1),2)+PN(NEC(J,4),2)+PN(NEC(J,8),2)+PN(NEC(J,5),2)/4
.

SZ=PN(NEC(J,1),3)+PN(NEC(J,4),3)+PN(NEC(J,8),3)+PN(NEC(J,5),3)/4.
               EDTVPA=( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5

SX=PN(NEC(J,2),1)+PN(NEC(J,3),1)+PN(NEC(J,7),1)+PN(NEC(J,6),1)/4
.

SY=PN(NEC(J,2),2)+PN(NEC(J,3),2)+PN(NEC(J,7),2)+PN(NEC(J,6),2)/4
.

SZ=PN(NEC(J,2),3)+PN(NEC(J,3),3)+PN(NEC(J,7),3)+PN(NEC(J,6),3)/4.
               EDTVPB=( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2
)**0.5
    IF(EDTVPA.LT.EDTVPB) THEN
                X(1)=PN(NEC(J,1),4+XG)
                Y(1)=PN(NEC(J,1),5+XG)
                X(2)=PN(NEC(J,4),4+XG)
                Y(2)=PN(NEC(J,4),5+XG)
                X(3)=PN(NEC(J,8),4+XG)
                Y(3)=PN(NEC(J,8),5+XG)
                X(4)=PN(NEC(J,5),4+XG)
                Y(4)=PN(NEC(J,5),5+XG)
               ELSE
                X(1)=PN(NEC(J,2),4+XG)
                Y(1)=PN(NEC(J,2),5+XG)
                X(2)=PN(NEC(J,3),4+XG)
                Y(2)=PN(NEC(J,3),5+XG)
                X(3)=PN(NEC(J,7),4+XG)
                Y(3)=PN(NEC(J,7),5+XG)
                X(4)=PN(NEC(J,6),4+XG)
                Y(4)=PN(NEC(J,6),5+XG)
                END IF
               X(5)=X(1)
               Y(5)=Y(1)
               IF(CSC.EQ.0) THEN
C SUPERIMPOSED
                CALL GSPLCI(WHITE)
                CALL GPL(5,X,Y)
               ELSE
                IF(KSOW.EQ.0) THEN
C WIRE FRAME
                CALL GSPLCI(GREY)
                CALL GPL(5,X,Y)
               ELSE
```

```
C SOLID
                CALL GSFACI(GREY)
                CALL GFA(4,X,Y)
                CALL GSPLCI(WHITE)
                CALL GPL(5,X,Y)
                END IF
               END IF

C BRICK NUMBER
              IF(KEN.EQ.1) THEN
               X(1)=PN(NEC(J,1),4+XG)+PN(NEC(J,2),4+XG)/4.0
               Y(1)=PN(NEC(J,1),5+XG)+PN(NEC(J,2),5+XG)/4.0
               X(1)=X(1)-CF*XDIS
               WRITE(BUF,*) J
               CALL GTX(X(1),Y(1),BUF)
               END IF
C END OF BRICK
              GOTO 5
              END IF
             END IF
5    CONTINUE
   END IF
C
C AXIS X,R=DKRED,Y,Z=DKBLUE,Z=DKGREEN
             CALL GSLN(1)
             X(1)=0.0-AX(1)-AY(1)-AZ(1)
             Y(1)=0.0-AX(2)-AY(2)-AZ(2)
             X(2)=X(1)+AX(1)
             Y(2)=Y(1)+AX(2)
             CALL GSTXCI(DKRED)
             CALL GSPLCI(DKRED)
             CALL GPL(2,X,Y)
             IF(NODIM.LT.4) THEN
              CALL GTX(X(2),Y(2),'X')
              ELSE
              CALL GTX(X(2),Y(2),'R')
             END IF
             X(2)=X(1)+AY(1)
             Y(2)=Y(1)+AY(2)
             CALL GSTXCI(DKBLUE)
             CALL GSPLCI(DKBLUE)
             CALL GPL(2,X,Y)
             IF(NODIM.LT.4) THEN
              CALL GTX(X(2),Y(2),'Y')
              ELSE
              CALL GTX(X(2),Y(2),'Z')
             END IF
             IF(NODIM.EQ.3) THEN
             X(2)=X(1)+AZ(1)
             Y(2)=Y(1)+AZ(2)
             CALL GSTXCI(DKGREEN)
             CALL GSPLCI(DKGREEN)
             CALL GPL(2,X,Y)
             CALL GTX(X(2),Y(2),'Z')
             CALL GSTXCI(WHITE)
             END IF
C
C NODAL LOOP
C
           IF(KNS.EQ.1.AND.IKJ.EQ.1) THEN
            CALL GSTXCI(YELLOW)
            CF=CF*1.1
            DO 70 I=1,NON
            IF(INF(I).EQ.0) GOTO 70
            X(1)=PN(I,4+XG)
            Y(1)=PN(I,5+XG)
            CALL GPM(1,X,Y)
            X(1)=X(1)-CF*XDIS
            Y(1)=Y(1)+0.01*YDIS
            IF(KNN.EQ.1) THEN
             WRITE(BUF,*) I
             CALL GTX(X(1),Y(1),BUF)
             END IF
70    CONTINUE
           END IF
           IF(CSC.EQ.0) GOTO 1000
C
C BOUNDARY CONDITION LOOP
C
           IF(KBC.EQ.1) THEN
           CALL GSTXCI(PURPLE)
           DO 80 I=1,NBC
            X(1)=PN(IDL(I),4+XG)
            Y(1)=PN(IDL(I),5+XG)
            IF(ID(I,1).EQ.1) CALL GTX(X(1)+0.0*XDIS ,Y(1),'DX')
            IF(ID(I,2).EQ.1) CALL GTX(X(1)+0.03*XDIS,Y(1),'DY')
            IF(ID(I,3).EQ.1) CALL GTX(X(1)+0.06*XDIS ,Y(1),'DZ')
            IF(ID(I,4).EQ.1) CALL GTX(X(1)+0.0*XDIS ,Y(1)-
0.03*YDIS,'RX')
            IF(ID(I,5).EQ.1) CALL GTX(X(1)+0.03*XDIS,Y(1)-
0.03*YDIS,'RY')
80     IF(ID(I,6).EQ.1) CALL GTX(X(1)+0.06*XDIS ,Y(1)-
0.03*YDIS,'RZ')
           END IF
C
C FORCES LOOP
C FORCES IN YELLOW , MOMENTS IN PURPLE
    IF(KFOR.EQ.1) THEN
           FMAX=0.0
           MMAX=0.0
           DO 89 I=1,NON
           DO 89 J=1,3
           IF(F(I,J).GT.FMAX) FMAX=F(I,J)
89     IF(F(I,J+3).GT.MMAX) MMAX=F(I,J+3)
           FMAX=FMAX*4.0/YDIS
           MMAX=MMAX*4.0/YDIS
           DO 90 I=1,NON
           IF(MMAX.EQ.0.0) GOTO 87
C MOMENTS
           CALL GSLWSC(2.0)
           CALL GSLN(3)
           CALL GSPLCI(PURPLE)
           X(1)=PN(I,4+XG)
           Y(1)=PN(I,5+XG)
           X(2)=X(1)+F(I,4)*SP/MMAX
           Y(2)=Y(1)-F(I,4)*SS*CP/MMAX
           CALL GPL(2,X,Y)
           X(2)=X(1)
           Y(2)=Y(1)+F(I,5)*CS/MMAX
           CALL GPL(2,X,Y)
           X(2)=X(1)-F(I,6)*CP/MMAX
           Y(2)=Y(1)-F(I,6)*SS*SP/MMAX
           CALL GPL(2,X,Y)
           CALL GSLN(1)
           CALL GSLWSC(0.5)

C FORCES
87     IF(FMAX.EQ.0.0) GOTO 90
           CALL GSPLCI(YELLOW)
           CALL GSLWSC(2.0)
           X(1)=PN(I,4+XG)
           Y(1)=PN(I,5+XG)
           X(2)=X(1)+F(I,1)*SP/FMAX
           Y(2)=Y(1)-F(I,1)*SS*CP/FMAX
           CALL GPL(2,X,Y)
           X(2)=X(1)
           Y(2)=Y(1)+F(I,2)*CS/FMAX
           CALL GPL(2,X,Y)
           X(2)=X(1)-F(I,3)*CP/FMAX
           Y(2)=Y(1)-F(I,3)*SS*SP/FMAX
           CALL GPL(2,X,Y)
90     CONTINUE
           END IF
1000   CONTINUE
C
C
C BOUNDARY
           CALL BOX(xmn,xmx,ymn,ymx,DKBLUE,0)
C
C END OF SEGMENT
C
           CALL GSLWSC(3.0)
           CALL GSPLCI(PURPLE)
C
C END OF ROUTINE
           CALL GSELNT (2)
           CALL GSCHH(1.50)
           RETURN
           END
C
C*************************************************************
**************
C
           SUBROUTINE SEARCH()
           INCLUDE 'COMMON.F'
C
C THIS SUBROUTINE SEARCHES FOR MAX DIMENSIONS AND
SETS SCREEN ACCORD.
C
           XMN=PN(1,4)
           XMX=PN(1,4)
           YMN=PN(1,5)
           YMX=PN(I,5)
C
C FIND MAX
C
           DO 10 I=2,NON
           IF(PN(I,4).GT.XMX) XMX=PN(I,4)
           IF(PN(I,4).LT.XMN) XMN=PN(I,4)
```

```
            IF(PN(I,5).GT.YMX) YMX=PN(I,5)
            IF(PN(I,5).LT.YMN) YMN=PN(I,5)
10    CONTINUE
C
C RATIO CHECK
C
            XDIS=XMX-XMN
            YDIS=YMX-YMN
            IF(YDIS.GT.XDIS) THEN
             XDIS=YDIS
             XMX=XMN+XDIS
            ELSE
             YDIS=XDIS
             YMX=YMN+YDIS
            END IF
            YMN=YMN-0.05*YDIS
            YMX=YMX+0.05*YDIS
            XMN=XMN-0.05*XDIS
            XMX=XMX+0.05*XDIS
            XDIS=XMX-XMN
            YDIS=YMX-YMN

C
C SCALE WINDOW
C
            CALL GSELNT (1)

            CALL GSWN (1,XMN,XMX,YMN,YMX)
            CALL GSELNT (2)
C
            RETURN
            END
C
C*************************************************************
**************
C
            SUBROUTINE RESET()
            INCLUDE 'COMMON.F'
C
C THIS SUBROUTINE RESETS THE ANGLE AND 2D DRAWING
C
            DO 10 I=1,NON
             PN(I,4)=PN(I,1)
10     PN(I,5)=PN(I,2)
            AX(1)=2.0
            AX(2)=0.0
            AY(1)=0.0
            AY(2)=2.0
            AZ(1)=0.0
            AZ(2)=0.0
            AS=0.0
            AP=1.57079

            CALL DRAW(1,'  ',0)
            RETURN
            END
C
C*************************************************************
**************
C
            SUBROUTINE ROTATE(FP,FS)
            REAL FP,FS,CP,SP,CS,SS
            INCLUDE 'COMMON.F'
C
C THIS SUBROUTINE ROTATES THE 3D COORDS AND MAPS
C THEM ONTO THE 2D VIEWING SURFACE AND CALCULATES
C THE DISPLACED POSITION
C
            AP=AP+FP*0.523599
            AS=AS+FS*0.523599
            CP=COS(AP)
            SP=SIN(AP)
            CS=COS(AS)
            SS=SIN(AS)
            DO 10 I=1,NON
             PN(I,4)=PN(I,1)*SP-PN(I,3)*CP
10     PN(I,5)=PN(I,2)*CS-PN(I,1)*SS*CP-PN(I,3)*SS*SP
            DO 20 I=1,NON
             PN(I,6)=(PN(I,1)+EXAG*RDISP(I,1))*SP
   &       -(PN(I,3)+EXAG*RDISP(I,3))*CP
             PN(I,7)=(PN(I,2)+EXAG*RDISP(I,2))*CS
   &       -(PN(I,1)+EXAG*RDISP(I,1))*SS*CP
   &       -(PN(I,3)+EXAG*RDISP(I,3))*SS*SP
20   CONTINUE
            AX(1)=2.0*SP
            AX(2)=-2.0*SS*CP
            AY(1)=0.0
            AY(2)=2.0*CS
            AZ(1)=-2.0*CP
            AZ(2)=-2.0*SS*SP
            RETURN
            END
C
C
*************************************************************
*************
C
            SUBROUTINE ZOOMIN()
            INCLUDE 'COMMON.F
C
C    THIS SUBROUTINE ZOOMS IN ON THE DRAWING
C
            CALL GSTXCI(WHITE)
            CALL GTX(2.0,86.0,'PICK ZOOM BOX')
            CALL GRQLC(1,2,1,NT,X(1),Y(1))
            X(1)=XMN+X(1)*XDIS/0.73
            Y(1)=YMN+Y(1)*YDIS/0.74874
            CALL GRQLC(1,2,1,NT,X(2),Y(2))
            X(2)=XMN+X(2)*XDIS/0.73
            Y(2)=YMN+Y(2)*YDIS/0.74874
            YMN=MIN0(Y(1),Y(2))
            XMN=MIN0(X(1),X(2))
            YMX=MAX0(Y(1),Y(2))
            XMX=MAX0(X(1),X(2))
            XDIS=XMX-XMN
            YDIS=YMX-YMN
            IF(YDIS.GT.XDIS) THEN
             XDIS=YDIS
             XMX=XMN+XDIS
            ELSE
             YDIS=XDIS
             YMX=YMN+YDIS
            END IF
            CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)

            CALL GSWN (1,XMN,XMX,YMN,YMX)
            CALL DRAW (1,'  ',0)

            RETURN
            END
C
C
*************************************************************
*************
C
            SUBROUTINE ZOOMOUT()
            INCLUDE 'COMMON.F
C
C    THIS SUBROUTINE ZOOMS OUT TO A FACTOR OF 2
C
            XMN=XMN-0.5*XDIS
            XMX=XMX+0.5*XDIS
            YMN=YMN-0.5*YDIS
            YMX=YMX+0.5*YDIS
            YDIS=YMX-YMN
            XDIS=XMX-XMN

            CALL GSWN (1,XMN,XMX,YMN,YMX)
            CALL DRAW (1,'  ',0)
            RETURN
            END
C
C*************************************************************
**************
C
            SUBROUTINE CENTER()
            INCLUDE 'COMMON.F
C
            CALL GSTXCI(WHITE)
            CALL GTX(2.0,85.0,'PICK CENTER')
            CALL GRQLC(1,2,1,NT,X(1),Y(1))
            X(1)=XMN+X(1)*XDIS/0.73
            Y(1)=YMN+Y(1)*YDIS/0.74874
            YMN=Y(1)-(YDIS/2.)
            YMX=Y(1)+(YDIS/2.)
            XMN=X(1)-(XDIS/2.)
            XMX=X(1)+(XDIS/2.)

            CALL GSWN (1,XMN,XMX,YMN,YMX)
            CALL DRAW(1,'  ',0)
            CALL BOX(-1.0,31.0,80.4,90.8,GREY,1)
            RETURN
            END
C
C*************************************************************
**************
C
            SUBROUTINE LIST()
            CHARACTER*90 BUF,NAME
            INCLUDE 'COMMON.F
```

```
C
C THIS SUBROUTINE LISTS THE
ELEM,NODE,BC,FORCES,GROUPS,MATERIALS
C
              CALL GSTXCI(WHITE)
C 1
              CALL GTX( 2.0, 34.0, 'NODES')
C 2
              CALL GTX(11.0, 34.0, 'ELEMENTS')
C 3
              CALL GTX(21.0, 34.0, ' B.C.')
C 4
              CALL GTX(2.0, 24.0, 'FORCES')
C 5
              CALL GTX(11.0, 24.0, 'GROUPS')
C 8
              CALL GTX(11.0,14.0, ' DISP.')
C 7
              CALL GSTXCI(14)
              CALL GTX( 2.0, 14.0, 'DONE')
              CALL GSELNT (1)
              CALL GSCHH(YDIS/50.0)
5     CALL GRQPK(1,2,II,SEG,JJ)
              CALL GSTXCI(WHITE)
C
C DONE
              IF(SEG.EQ.SEVEN) THEN
              CALL GSELNT (2)
              CALL GSCHH(1.5)
              CALL GSTXCI(DKBLUE)
              CALL GTX( 2.0, 34.0, 'NODES')
              CALL GTX(11.0, 34.0, 'ELEMENTS')
              CALL GTX(21.0, 34.0, ' B.C.')
              CALL GTX(2.0, 24.0, 'FORCES')
              CALL GTX(11.0, 24.0, 'GROUPS')
              CALL GTX( 2.0, 14.0, 'DONE')
              CALL GTX(11.0,14.0, ' DISP.')
              RETURN
C
C NODES
              ELSEIF(SEG.EQ.ONE) THEN
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='NODE#     X      Y      Z'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)

              DO 10 I=1,NON,15
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='NODE#     X      Y      Z'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)
              DO 12 J=1,15
              Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
              WRITE(BUF,95) I+J-1,(PN(I+J-1,JJ),JJ=1,3)
              IF(INF(I+J-1).EQ.0) GOTO 12
              CALL GTX(X(1),Y(1),BUF)
12    CONTINUE
10    CALL GRQST(1,1,K,J,BUF)
C
C ELEMENTS
              ELSEIF(SEG.EQ.TWO) THEN
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='  ELEM#  GROUP#      NODES'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)

              DO 20 I=1,NOE,15
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='  ELEM#  GROUP#      NODES'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)
              DO 22 J=1,15
              Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
              IF(IEF(I+J-1).EQ.0) GOTO 22
              WRITE(BUF,96) I+J-1,NEC(I+J-1,9),(NEC(I+J-1,K),K=1,
     @                NUMNOD(NOETY(NEC(I+J-1,9))))
              CALL GTX(X(1),Y(1),BUF)
22    CONTINUE
20    CALL GRQST(1,1,K,J,BUF)
C
C B.C.
              ELSEIF(SEG.EQ.THREE) THEN
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='  NODE# Dx    Dy    Dz    Rx    Ry    Rz 1=
FIXED/0=
   @FREE'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)

              DO 30 I=1,NBC,15
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='  NODE# Dx    Dy    Dz    Rx    Ry    Rz 1=
FIXED/0=
   @FREE'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)
              DO 32 J=1,15
              IF((I+J-1).GT.NBC) GOTO 32
              Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
              WRITE(BUF,97)IDL(I+J-1),(ID(I+J-1,K),K=1,6)
              CALL GTX(X(1),Y(1),BUF)
32    CONTINUE
30    CALL GRQST(1,1,K,J,BUF)
C
C FORCES
              ELSEIF(SEG.EQ.FOUR) THEN
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='  NODE#  Fx     Fy     Fz     Mx     My     Mz'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)

              DO 40 I=1,NON,15
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='  NODE#  Fx     Fy     Fz     Mx     My     Mz'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)
              DO 42 J=1,15
              IF((I+J-1).GT.NON) GOTO 42
              Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
              WRITE(BUF,92) I+J-1,(F(I+J-1,K),K=1,6)
              CALL GTX(X(1),Y(1),BUF)
42    CONTINUE
40    CALL GRQST(1,1,K,J,BUF)
C
C DISPLACEMENTS
              ELSEIF(SEG.EQ.EIGHT) THEN
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='  NODE#  Dx     Dy     Dz     Rx     Ry     Rz'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)

              DO 47 I=1,NON,15
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='  NODE#  Dx     Dy     Dz     Rx     Ry     Rz'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)
              DO 48 J=1,15
              IF((I+J-1).GT.NON) GOTO 48
              Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
              WRITE(BUF,92) (I+J-1),(RDISP(I+J-1,K),K=1,6)
              CALL GTX(X(1),Y(1),BUF)
48    CONTINUE
47    CALL GRQST(1,1,K,J,BUF)

C
C GROUPS
              ELSEIF(SEG.EQ.FIVE) THEN
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='    GROUP#  ELEM TYPE  #OF ELEMS
MATERIAL#'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)

              DO 50 I=1,NOG,15
              CALL BOX(xmn,xmx,ymn,ymx,black,1)
              BUF='    GROUP#  ELEM TYPE  #OF ELEMS
MATERIAL#'
              X(1)=XMN+0.02*XDIS
              Y(1)=YMN+0.95*YDIS
              CALL GTX(X(1),Y(1),BUF)
              DO 52 J=1,15
              IF((I+J-1).GT.NOG) GOTO 52
                IF(NOETY(I+J-1).EQ.1) THEN
                 NAME='3D SPRING '
                ELSEIF(NOETY(I+J-1).EQ.2) THEN
                 NAME='2D TRUSS  '
                ELSEIF(NOETY(I+J-1).EQ.3) THEN
                 NAME='2D BEAM   '
                ELSEIF(NOETY(I+J-1).EQ.4) THEN
```

```
                NAME='GRID BEAM '
               ELSEIF(NOETY(I+J-1).EQ.5) THEN
                NAME='3D TRUSS  '
               ELSEIF(NOETY(I+J-1).EQ.6) THEN
                NAME='3D BEAM   '
               ELSEIF(NOETY(I+J-1).EQ.7) THEN
                NAME='3D PLATE  '
               ELSEIF(NOETY(I+J-1).EQ.8) THEN
                NAME='3D BRICK  '
               ELSEIF(NOETY(I+J-1).EQ.9) THEN
                NAME='3D FRAME  '
               ELSEIF(NOETY(I+J-1).EQ.10) THEN
                NAME='2D PLANE STRAIN '
               ELSEIF(NOETY(I+J-1).EQ.11) THEN
                NAME='2D PLANE STRESS '
               ELSEIF(NOETY(I+J-1).EQ.12) THEN
                NAME='GRID PLATE '
               END IF
              Y(1)=YMN+0.95*YDIS-J*0.05*YDIS
              WRITE(BUF,98) I+J-1,NOETY(I+J-1),NOEIG(I+J-1),I+J-1
              CALL GTX(X(1),Y(1),BUF)
              CALL GTX(XMN+0.7*XDIS,Y(1),NAME)
52    CONTINUE
50    CALL GRQST(1,1,K,J,BUF)

              END IF
              GOTO 5
92    FORMAT(I7,8F10.2)
95    FORMAT(I7,3F9.3)
96    FORMAT(10I7)
97    FORMAT(7I7)
98    FORMAT(4I11)
              END
C
C
C*************************************************************
**************
C
              SUBROUTINE SORT()
              REAL EDTVP(800)
              INCLUDE 'COMMON.F'
C     OPEN(UNIT=5,FILE='SORT.DAT')
C
C THIS SUBROUTINE SORTS THE ELEMENT FROM THE CLOSEST
TO
C THE FURTHEST AND DRAWS THEM FAR TO CLOSE IN DRAW
C
C NOOSE...NUMBER OF ORDER SORTED ELEMENTS
C EDTVP...ELEMENT DISTANCE TO VIEW POINT
C
C CALCULATE VIEW POINT
C
              XV=100000.0*COS(AS)*COS(AP)
              YV=100000.0*SIN(AS)
              ZV=100000.0*COS(AS)*SIN(AP)
C     WRITE(5,*)' VIEW POINT ',XV,YV,ZV
C
C CALCULATE EDTVP's
C
C     WRITE(5,*) 'NOE=',NOE
              DO 20 I=1,NOE
              NOOSE(I)=I
C     WRITE(5,*)' ELEMENT #',I
C AVERAGE OF ELEMENT
              SX=0.0
              SY=0.0
              SZ=0.0
              DO 30 J=1,NUMNOD(NOETY(NEC(I,9)))
              SX=SX+PN(NEC(I,J),1)
              SY=SY+PN(NEC(I,J),2)
30    SZ=SZ+PN(NEC(I,J),3)
              SX=SX/NUMNOD(NOETY(NEC(I,9)))
              SY=SY/NUMNOD(NOETY(NEC(I,9)))
              SZ=SZ/NUMNOD(NOETY(NEC(I,9)))
C     WRITE(5,*)'AVERAGE POINT',SX,SY,SZ
C FIND DISTANCE
20    EDTVP(I)= ( (XV-SX)**2 + (YV-SY)**2 + (ZV-SZ)**2 )**0.5
C20    WRITE(5,*)'DISTANCE=',EDTVP(I)
C
C SORTING ROUTINE (BUBBLE DOWN)
C
              DO 40 I=NOE-1,2,-1
              DO 40 J=1,I
C     WRITE(5,*) I,J
               IF(EDTVP(NOOSE(J)).LT.EDTVP(NOOSE(J+1)))
THEN
                IJK=NOOSE(J)
                NOOSE(J)=NOOSE(J+1)
                NOOSE(J+1)=IJK
                END IF
40    CONTINUE
C     WRITE(5,*)' SORTED ORDER '
C     DO 50 I=1,NOE
C50    WRITE(5,*) NOOSE(I)
C     WRITE(5,*)' DISTANCE ORDER '
C     DO 60 I=1,NOE
C60    WRITE(5,*)EDTVP(NOOSE(I))
C
C     CLOSE(UNIT=5)
              RETURN
              END
```

Appendix C

Potential Flow Element Matrix and Load Vector

The potential flow element described in the body has the following form:

$$[S] = \begin{bmatrix} 3S_{11} & -S_{12} & -S_{13} & 4S_{12} & 0 & 4S_{13} \\ & 3S_{22} & -S_{23} & 4S_{12} & 4S_{23} & 0 \\ & & 3S_{33} & 0 & 4S_{23} & 4S_{13} \\ & & & 8(S_{33}-S_{12}) & 8S_{13} & 8S_{23} \\ & SYMM. & & & 8(S_{11}-S_{23}) & 8S_{12} \\ & & & & & 8(S_{22}-S_{31}) \end{bmatrix}$$

where $S_{ij} = (a_i a_j + b_i b_j) \rho / (12 A)$ with A the area of the element triangle

The corresponding load vector is:

$$\{SL\} = \frac{\rho}{6} \begin{Bmatrix} L_2 + L_3 \\ L_3 + L_1 \\ L_1 + L_2 \\ 4L_3 \\ 4L_1 \\ 4L_2 \end{Bmatrix}$$

where $L_i = V_n L_i$ with V_n the normal velocity across side length L_i

Velocity in is -V and velocity exiting is +V.

Appendix D

Typical Data Files

The data files generated by the INPUT program and read in by the SOLVER program are described in appendix A in the MHYFECS Manual. The data file generated by the SOLVER program and used by the OUTPUT program will be demonstrated here. The following is a small data file generated by SOLVER and read in by OUTPUT, notice that all of these files have names ending in .STS as this is the extension given the stresses and displacements results files.

```
 15   15    8    8    3    1   15    1    1
  1  0.0000  0.0000  0.0000
  2  0.0000  4.0000  0.0000
  3  0.0000  8.0000  0.0000
  4  4.0000  0.0000  0.0000
  5  4.0000  4.0000  0.0000
  6  4.0000  8.0000  0.0000
  7  8.0000  0.0000  0.0000
  8  8.0000  4.0000  0.0000
  9  8.0000  8.0000  0.0000
 10 12.0000  0.0000  0.0000
 11 12.0000  4.0000  0.0000
 12 12.0000  8.0000  0.0000
 13 16.0000  0.0000  0.0000
 14 16.0000  4.0000  0.0000
 15 16.0000  8.0000  0.0000
  7    8    1
  1    1    4    5    2
  2    2    5    6    3
  3    4    7    8    5
  4    5    8    9    6
  5    7   10   11    8
  6    8   11   12    9
  7   10   13   14   11
  8   11   14   15   12
  1    1    1    1    0    0    1
  2    1    1    1    0    0    1
  3    1    1    1    0    0    1
  4    0    0    0    0    0    1
  5    0    0    0    0    0    1
  6    0    0    0    0    0    1
  7    0    0    0    0    0    1
  8    0    0    0    0    0    1
  9    0    0    0    0    0    1
 10    0    0    0    0    0    1
 11    0    0    0    0    0    1
 12    0    0    0    0    0    1
 13    0    0    0    0    0    1
 14    0    0    0    0    0    1
 15    0    0    0    0    0    1
```

```
         15       0
 1 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 2 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 3 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 4 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
```

```
 5 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 6 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 7 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 8 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 9 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
10 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
11 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
12 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
13 0.000000E+00 6.250000E+01 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
14 0.000000E+00 1.250000E+02 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
15 0.000000E+00 6.250000E+01 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
 1  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 2  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 3  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 4  0.37898E-04  0.32897E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 5 -0.58372E-20  0.26667E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 6 -0.37898E-04  0.32897E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 7  0.65987E-04  0.94130E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 8 -0.20663E-19  0.90986E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 9 -0.65987E-04  0.94130E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
10  0.82660E-04  0.17852E-03  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
11 -0.33886E-19  0.17691E-03  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
12 -0.82660E-04  0.17852E-03  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
13  0.88340E-04  0.27433E-03  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
14 -0.29869E-19  0.27377E-03  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
15 -0.88340E-04  0.27433E-03  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
     1       4
 0.31235E+03  0.93704E+02  0.12182E+03  0.94896E+02  0.00000E+00  0.00000E+00  0.26356E+03
 0.29694E+03  0.42358E+02  0.10179E+03 -0.14425E+02  0.00000E+00  0.00000E+00  0.23203E+03
-0.15404E+02 -0.51346E+02 -0.20025E+02 -0.32396E+02  0.00000E+00  0.00000E+00  0.65541E+02
 0.00000E+00  0.00000E+00  0.00000E+00  0.76925E+02  0.00000E+00  0.00000E+00  0.13324E+03
     2       4
 0.00000E+00  0.00000E+00  0.00000E+00  0.76925E+02  0.00000E+00  0.00000E+00  0.13324E+03
 0.15404E+02  0.51346E+02  0.20025E+02 -0.32396E+02  0.00000E+00  0.00000E+00  0.65541E+02
-0.29694E+03 -0.42358E+02 -0.10179E+03 -0.14425E+02  0.00000E+00  0.00000E+00  0.23203E+03
-0.31235E+03 -0.93704E+02 -0.12182E+03  0.94896E+02  0.00000E+00  0.00000E+00  0.26356E+03
     3       4
 0.21610E+03  0.18104E+02  0.70261E+02  0.67312E+02  0.00000E+00  0.00000E+00  0.21257E+03
 0.22373E+03  0.43540E+02  0.80180E+02 -0.13714E+02  0.00000E+00  0.00000E+00  0.16665E+03
-0.77732E+01 -0.25911E+02 -0.10105E+02 -0.48115E+01  0.00000E+00  0.00000E+00  0.19015E+02
-0.15404E+02 -0.51346E+02 -0.20025E+02  0.76214E+02  0.00000E+00  0.00000E+00  0.13628E+03
     4       4
 0.15404E+02  0.51346E+02  0.20025E+02  0.76214E+02  0.00000E+00  0.00000E+00  0.13628E+03
 0.77732E+01  0.25911E+02  0.10105E+02 -0.48115E+01  0.00000E+00  0.00000E+00  0.19015E+02
-0.22373E+03 -0.43540E+02 -0.80180E+02 -0.13714E+02  0.00000E+00  0.00000E+00  0.16665E+03
-0.21610E+03 -0.18104E+02 -0.70261E+02  0.67312E+02  0.00000E+00  0.00000E+00  0.21257E+03
     5       4
 0.12964E+03  0.15314E+02  0.43487E+02  0.53084E+02  0.00000E+00  0.00000E+00  0.13819E+03
 0.13344E+03  0.27964E+02  0.48420E+02  0.49887E+01  0.00000E+00  0.00000E+00  0.97263E+02
-0.39783E+01 -0.13261E+02 -0.51717E+01  0.94161E+01  0.00000E+00  0.00000E+00  0.18507E+02
-0.77732E+01 -0.25911E+02 -0.10105E+02  0.57511E+02  0.00000E+00  0.00000E+00  0.10107E+03
     6       4
 0.77732E+01  0.25911E+02  0.10105E+02  0.57511E+02  0.00000E+00  0.00000E+00  0.10107E+03
 0.39783E+01  0.13261E+02  0.51717E+01  0.94161E+01  0.00000E+00  0.00000E+00  0.18507E+02
-0.13344E+03 -0.27964E+02 -0.48420E+02  0.49887E+01  0.00000E+00  0.00000E+00  0.97263E+02
-0.12964E+03 -0.15314E+02 -0.43487E+02  0.53084E+02  0.00000E+00  0.00000E+00  0.13819E+03
     7       4
 0.42832E+02  0.78221E+00  0.13084E+02  0.37936E+02  0.00000E+00  0.00000E+00  0.75628E+02
 0.45414E+02  0.93882E+01  0.16441E+02  0.21552E+02  0.00000E+00  0.00000E+00  0.49870E+02
-0.13965E+01 -0.46549E+01 -0.18154E+01  0.24564E+02  0.00000E+00  0.00000E+00  0.42657E+02
-0.39783E+01 -0.13261E+02 -0.51717E+01  0.40948E+02  0.00000E+00  0.00000E+00  0.71461E+02
     8       4
 0.39783E+01  0.13261E+02  0.51717E+01  0.40948E+02  0.00000E+00  0.00000E+00  0.71461E+02
 0.13965E+01  0.46549E+01  0.18154E+01  0.24564E+02  0.00000E+00  0.00000E+00  0.42657E+02
-0.45414E+02 -0.93882E+01 -0.16441E+02  0.21552E+02  0.00000E+00  0.00000E+00  0.49870E+02
```

```
-0.42832E+02 -0.78221E+00 -0.13084E+02  0.37936E+02  0.00000E+00  0.00000E+00  0.75628E+02
```

The first part of the data file is the same as the input files, the first line is the control data, then the nodal position list, the elements connectivity list, and the boundary conditions. The file then lists the forces after the force control data line and then the displacement at each node list. The last part of the data file is the stresses for each element. The control line for each element is the element number and how many nodes are in that element. There is then 7 stresses for each node in the element, they are Sx, Sy, Sz, Sxy, Syz, Szx, and Seq.

The other data file not yet discussed is the fluid flow input file. The file listed below is the input data file for the fluid flow model of the sluice gate open 5 meters.

```
   76      185       6
  1    1    15    16    14     3     2
  2   16    80    81    79    18    17
  3   81    82    83    84    85    89
  4   81    89    85    88    91    90
  5   18    79    81    90    91    78
  6    3    14    16    17    18    13
                    :
                    :
                    :
 70  132   130   134   182   100   172
 71   97   109    98   175   122   149
 72   98   119    99   177   124   174
 73   99   131   100   183   135   176
 74  112   111   122   146    34   143
 75  122   121   124   148    36   145
 76  124   123   135   184    37   147
    1 0.000000000000000      15.0119649836519
    2 2.50000000000000       15.0199871704997
    3 5.00000000000000       15.0280093573475
    4 8.75000000000000       15.2158866074041
    5 12.5000000000000       15.4037747765614
    6 16.2500000000000       15.6864755740981
                    :
                    :
                    :
  179 20.6250000000000       2.13000000000000
  180 25.4065000000000       1.55000000000000
  181 28.1105000000000       0.860000000000000
  182 30.7475000000000       0.220000000000000
  183 33.8830000000000       0.000000000000000
  184 37.5170000000000       0.000000000000000
  185 25.1362000000000       4.80354627428975
1   0    1    1    BC IN
2   0    2    1    BC IN
3   0    3    1    BC IN
4   1   33    2    BC OUT
5   1   34    2    BC OUT
```

```
6    1    35   2      BC OUT
1     83              HEAD IN
32    37              HEAD OUT
2                    WIDTH OF FLOW IN SURFACE
5    7               FLOW IN VECTOR
7     4              LENGTH,DEPTH OF FLEX-NETWORK
94   95   96   101
107  117  129  133
31   42   40   38
110  120  132  134
97   98   99   100
112  122  124  135
32   34   36   37
0.0001   250                  CONCRI & ITLIM
9.81    998.0                 GRAV & DENSITY
158  156  154  152  150  144   MID NODES
```

This data file starts with the line # of nodes, # of elements, and # of boundary conditions. The list of elements connectivities and nodal positions has been shortened to save space. The data file then lists the inflow and outflow boundary conditions. They are the 3 elements on the inflow edge, elements 1,2,3, and on the outflow edge, 33,34,35. The 0 preceding the inflow elements is the flag for inflow and 1 preceding the outflow elements the flag for outflow. The numbers proceeding the element numbers is which side of the element the boundary condition is applied to. Next is the pair of nodes the represent the head of the incoming flow and then the pair representing the exit flow. The next two lines are the control for the free surface before the gate. The width is how many corner nodes are in the free surface upstream of the gate and the following list is those nodes. Next is the control numbers for the flexible network used to model the exit flow stream and then the list of the numbers in the flexible network. Following that is the convergence criteria and iteration limit, then the gravity and density constants. The last line is the list of mid side nodes on the exit stream free surface that are used to insure the flow is parallel to the free surface as should be. The data generated by the fluid solder is graphically displayed in the solution phase and only stored in a file for debugging purposes so will not be discussed here.

Appendix E

Three Dimensional Rotation Matrices

For a true three dimensional rotation matrix, the rotation angles about all three axis must be used so these angles will be defined first. The angle of rotation about the X axis will be denoted by the variable θ_x and is shown in *Figure D1*.

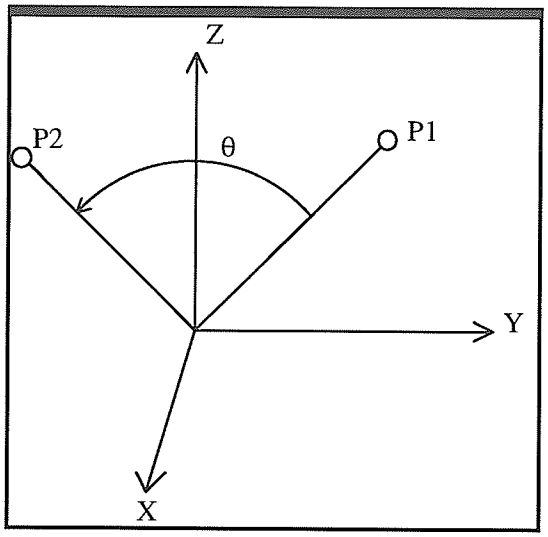


Figure D1 Rotation Angle about X axis

In *Figure D1*, θ_X is 90 degrees so if P1 had the XYZ coordinates of (0,1,1), then P2 would become (0,-1,1) after the rotation through θ_X. This rotation can be better represented with a rotation matrix in the equation:

$$[R_x]\{P_1\} = \{P_2\} \text{ where } P_1 = \begin{Bmatrix} X_1 \\ Y_2 \\ Z_3 \end{Bmatrix} \text{ and } P_2 = \begin{Bmatrix} X_2 \\ Y_2 \\ Z_3 \end{Bmatrix}$$

$$\text{with } X_1 = X_2$$

$$\text{and } R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}$$

Similarly, the matrices $\mathbf{R_y}$ and $\mathbf{R_Z}$ can be shown to be:

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \text{ and } \mathbf{R}_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The need for these three rotation matrices arises from the fact that the element stiffness matrices used in the MHYFECS programs are all derived as if they extended down the X axis or lie on the XY plane. In a real model though, the element may be at any orientation so the stiffness matrix must be accordingly rotated to reflect the change in position. To do this, it is first necessary to define, relative to an element, what the rotation angles are. The example presented here is the 6 noded beam element developed just for the MHYFECS programs. The four beam positions shown in Figure D2 are one, the beam in its unrotated position, two, the beam after its X rotation, three, the beam after its Z rotation, and four, the beam in its final position after the Y rotation.

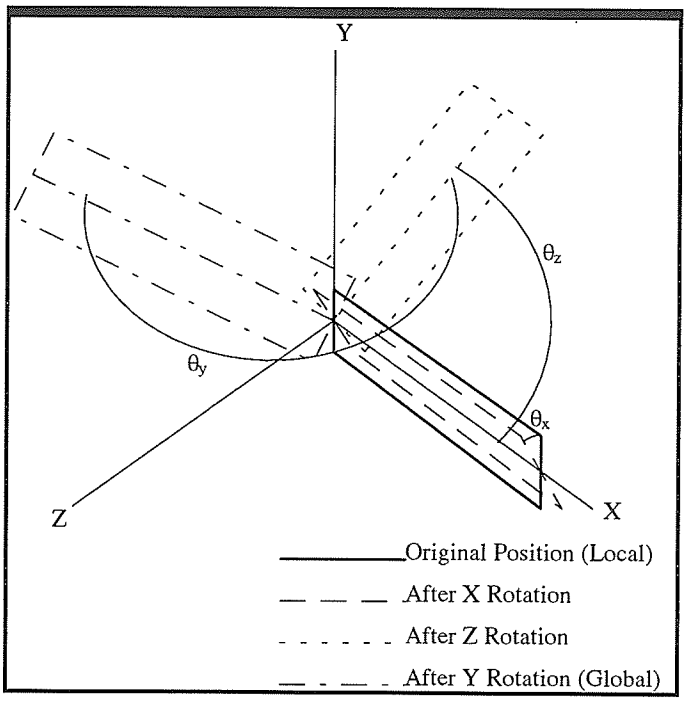


Figure D2 Rotation Angles

From Figure D2 the local to global rotation matrix can be seen to be:

$$[R_{L\to G}] = [R_X]\,[R_Z]\,[R_Y]$$

which can be expanded to be:

$$[R_{L\to G}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}$$

$$\therefore [R_{L\to G}] = \begin{bmatrix} C_yC_z & -S_z & -S_yC_z \\ C_xC_yS_z + S_xS_y & C_xC_z & S_xC_y - C_xS_yS_z \\ C_xS_y - S_xC_yS_z & -S_xC_z & S_xS_yS_z + C_xC_y \end{bmatrix}$$

where $C_x = \cos\theta_x$, $S_x = \sin\theta_x$, etc.

$[\mathbf{R_{L\to G}}]$ is used to convert Local matrices into Global coordinates through the use of the following equation:

$$[K_G] = [Q]^T\,[K_L]\,[Q]$$

where $[Q]$ is the global rotation matrix

The $[\mathbf{K_G}]$ can now be assembled into the global stiffness matrix from which the solution is found. The resulting solutions are in the form of displacements from which strains can be found. The resulting displacements are in global coordinates and thus must be rotated back to local coordinates to be used to find strains. This is done by simply reversing the rotation angles and the sequence they are used in. The global to local rotation matrix can be found through the multiplication of $[\mathbf{R_y}]^{-1}[\mathbf{R_z}]^{-1}[\mathbf{R_x}]^{-1}$ which is actually just the transpose of $[\mathbf{R_{L\to G}}]$. The displacements, once rotated to the local coordinate system, are used to find the strains in the element and then the stresses through the stress-strain relationship $\sigma=E\varepsilon$. The stresses found are in local coordinates so they must be rotated back into global coordinates so that the stress plots of the structure are in global stress

coordinates. The conversion of local stresses to global stresses, of any other local vector to global vector, is done through the use of the following equation:

$$[R_{L\to G}]\{\sigma_L\} = \{\sigma_G\}$$

The implementation of the rotation matrices can be seen in the SOLVER program listing.