TM-51-89-15CR

September 1989

NCEL

An Investigation Conducted by:
M. Sanayfi and S. Scampoli
Tufts University

Sponsored by: Office of Naval Technology
Arlington, VA 22217-5000

Program No.: RM33F60

Technical Memorandum

IDENTIFICATION OF STRUCTURAL ELEMENT STIFFNESSES OF REINFORCED CONCRETE DECKS FROM INCOMPLETE STATIC TEST DATA

ABSTRACT A parameter identification program **PARIS.SAS** (**PAR**ameter **I**dentification of **S**tructures) is developed for the systematic identification of plate bending stiffness parameters for the Navy's one-third scale pier deck model. The plate bending stiffnesses of the Naval pier deck are identified at the element level by using incomplete static test data on a subset of the degrees of freedom used to define the finite element model.

The finite element model of the reinforced concrete Navy pier deck is generated by using three-dimensional isoparametric elements to model the beams, and hourglass plate bending elements to model the orthotropic slab.

Several successful parameter identification examples are performed on the instrumented test span of the Navy pier deck using simulated data. The level of acceptable error is determined for the convergence of the algorithm.

It is believed that continued research of this approach of static parameter identification will lead to a practical procedure for damage assessment and load carrying capacity determination of full scale structures.

NAVAL CIVIL ENGINEERING LABORATORY PORT HUENEME CALIFORNIA 93043

Approved for public release; distribution is unlimited.

TABLE OF CONTENTS

Appendix Page

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

1.1 Damage Assessment of Naval Pier Decks

A large number of reinforced concrete Navy piers and bridges were built over four decades ago. In the course of the service life of the pier and bridge decks, these structures have sustained varying degrees of damage, including cracking, spalling, and chemical deterioration. Although some of the damage is apparent, it is difficult to evaluate the extent of the deterioration and the remaining load carrying capacity of these pier and bridge decks from visual inspections. The difficulty of visual inspection is that some of the damage may be internal (such as delamination and voids) and not detectable by field observations. Another disadvantage of visual inspection is that there is no relationship between the field observations and the remaining capacity of the structure. Identifying the actual plate (slab) bending stiffnesses and remaining capacity of the bridges and piers decks would classify structures in need of repair and rehabilitation. The systematic identification of damaged areas would allow civil engineers to plan and prepare maintenance and repairs that would significantly extend the structures' service life.

A finite element software package for PARarameter Identification of Structures (PARIS) has been developed to identify the stiffness properties of orthotropic plates using static force and displacement measurements. This program is

presented in Appendix B of this report, and is on the magnetic tape enclosed with this report. The plate bending stiffnesses of linear elastic plates are identified by using static force and displacement measurements on a subset of the global degrees of freedom of the finite element model. This method of static parameter identification is capable of determining major changes in the structural properties at the element level, including element failure.

The finite element model of the reinforced concrete pier deck is generated by employing an orthotropic plate bending element suitable for modelling reinforced concrete plates. The orthotropic element is capable of considering different amounts of reinforcement in orthogonal directions. The finite element is also capable of modelling cracked and uncracked sections of the pier deck by using equivalent moments of inertia.

Several successful examples of parameter identification (using simulated test data) were performed on the instrumented test span of the Navy pier deck. An error analysis was performed to determine the acceptable level of measurement error. Unfortunately, the actual test data was not analyzed as part of this report, as the test data was not provided to the incumbent by the Navy Civil Engineering Laboratory as stated in the Work Statement for Identification of Structural Element Stiffnesses of Reinforced Concrete Decks from Incomplete Static Test Data, contract N62583/89 P 1096.

If NCEL wishes to implement the parameter identification

program to a practical example the parameter identification program needs to be modified in the following way, the matrices of simulated force and displacement measurements must be replaced with the actual test data. For more details on the implementation of parameter identification to a practical example please refer to chapter seven of Ref. [17].

1.2 Literature Survey

The extent of literature on damage assessment of reinforced concrete pier decks is very limited. However, the properties and behavior of a reinforced concrete pier deck are very similar to the properties and behavior of a reinforced concrete bridge deck. Thus, the following literature survey will examine the work that has been accomplished in damage assessment of reinforced concrete bridges.

There are 574,045 bridges in the United States according to the Seventh Annual Report to Congress, "Highway Bridge Replacement and Rehabilitation Program" [1]. According to this report the number of bridges that are either "structurally deficient" or "functionally obsolete" is 243,917. Thus, approximately 43% of all the nation's bridges require some form of rehabilitation or reconstruction. In the state of California alone there are 22,269 bridges of which 5,737 are termed deficient.

The factors that cause a reinforced concrete bridge to become structurally deficient, include poor design details, environmental conditions, age, overstress, or lack of proper maintenance. The deterioration of a reinforced concrete structure can include many types of damage; cracks, scaling of the aggregate, spalling, voids, delamination, and corrosion of the reinforcement.

The purpose of any type of damage assessment is to record any type of damage before catastrophic failure of the structure. Damage assessment can be grouped into two categories; field inspection and parameter identification. These two categories can

be further subdivided. Field inspections can consist of visual inspection, non-destructive tests, and destructive tests. By contrast, parameter identification, which is based on the finite element method, can be subdivided into dynamic parameter identification, and static parameter identification.

Detailed methods of visual inspection are presented in the National Cooperative Highway Research Program Report 312, "Condition Surveys of Concrete Bridge Components" [2]. A visual inspection consists of inspecting the structural integrity of the bridge by visually identifying damaged areas. A visual inspection is effective only for certain types of damage; large visible cracks, rust stains (a sign of reinforcement corrosion), and spalling. One disadvantage of a visual inspection is that damage can only be detected when visible; slight damage such as small cracks, delamination, and voids can not be detected by visual inspections. Another drawback of visual inspection is that there is no relation between the field observations and the load carrying capacity of the damaged structure.

More advanced techniques are available for field inspections of bridges; nondestructive and destructive tests. There are many methods available of nondestructive testing. A detailed summary of the tests available, can be found in the NCHRP Synthesis of Highway Practice 118, "Detecting Defects and Deterioration in Highway Structures" [3]. A brief listing of some of the methods available, is: sonic methods, ultrasonic techniques, magnetic methods, nuclear methods, and radar. Quite simply, the methods

of nondestructive testing are based on measuring how different types of waves travel through concrete (sound, radar, etc.) to detect areas of internal (nonvisible) damage. The drawback of nondestructive tests is that the information derived from the tests is seldom complete. The methods are usually performed on a small area of the structure where damage is suspected. Similar to visual inspections, there is no correlation between the observed damage and the remaining capacity of the structure. The non-destructive test methods only provide limited information on a portion of the structure.

Destructive tests consist of taking samples of the structure for laboratory testing. Cores of reinforced concrete decks and reinforcing bar samples can be removed for laboratory tests. A more detailed explanation of destructive tests for reinforced concrete can be found in the American Society for Testing and Material Standards [4]. The disadvantages of destructive tests are similar to those of nondestructive tests. The tests only give limited information about certain areas of the structure, there is no relationship between the measured damage and the remaining load carrying capacity of the structure, and destructive tests can weaken the structure if not properly executed by a certified engineer.

Both dynamic and static parameter identification are based on the finite element method. The structure is discritized into finite elements, which can be represented by the following differential matrix equation

$$[M]\{\ddot{u}(t)\} + [C]\{\dot{u}(t)\} + [K]\{u(t)\} = \{p(t)\} \quad (1.1)$$

where,

[M] = Mass Matrix

[C] = Damping Matrix

[K] = Stiffness Matrix

$\{\ddot{u}(t)\}$ = Acceleration Vector

$\{\dot{u}(t)\}$ = Velocity Vector

$\{u(t)\}$ = Displacement Vector

$\{p(t)\}$ = Applied Force Vector

The intent of parameter identification is to adjust the parameters of the finite element model (the parameters may be element stiffness properties such as moment of inertia or modulus of elasticity, or the parameters may be entire matrices, [K], [M] or [C]) to match the analytical and measured responses, given the excitation. In all types of parameter identification some form of iterative solution is employed to achieve a match between the analytical and measured responses.

Dynamic parameter identification can be divided into two main branches according to whether the analysis takes place in the time domain or the frequency domain. In the time domain the intent is to match the measured and analytical time history responses by adjusting the stiffness, mass, and damping matrices. By contrast, in the frequency domain the intent is to evaluate the modal parameters in order to match the analytical and measured responses.

The dynamic parameter identification literature is quite extensive. Hart and Yao [5] have prepared a survey paper with an

extensive bibliography. Beck [6] summarized a large number of papers in the area of dynamic parameter identification from strong motion records. Hudson [7] has compiled a survey paper that discusses the subject of dynamic parameter identification of full scale structures. Liu and Yao [8], Berman [9], and Adelman and Haftka [10] have also compiled surveys of the work done in the area of dynamic parameter identification.

Several recent papers that discuss dynamic parameter identification of full scale structures, specifically reinforced concrete bridges include the following. Douglas and Reid [11] used a pullback and quick release method to recover mode shapes and natural frequencies of the Rose Creek Interchange, a reinforced concrete box girder bridge. Werner, Douglas, and Crouse [12] used strong motion records and a mean squared difference approach to recover the pseudo-static matrix, natural frequency, and damping ratio of the Meloland Road Overcrossing. Buckle, Richardson, and Sveinsson [13] determined mode shapes, natural frequencies, and damping ratios of the Dominion Road Bridge using the snap-back and quick-release method.

Although there have been successful examples of dynamic parameter identification of reinforced concrete structures there are several disadvantages to this method. A large amount of dynamic data is needed to derive an accurate response of the structure. In all cases, an estimate for the damping matrix must be used, which induces error into the system identification. The unknown parameters are generally very sensitive to measurement

errors. The identification process usually does not occur at the element level. Stiffness matrices are recovered from dynamic parameter identification, not the element properties.

There are only three papers available on static parameter identification. Sheena, Zalmanovitch, and Unger [14] presented a method, which tried to optimize the difference between the theoretical and corrected stiffness matrix subjected to equilibrium constraints. The main problem with this method is that it requires a complete set of force and displacement measurements for the structure. To create a complete set of test measurements artificially, Sheena et. al. [14] used spline theorems of different orders for different types of finite elements, which introduced a major source of error into the algorithm. Finally, they attempted to compute the entire stiffness matrix, not the element properties.

Sanayei and Nelson [15] proposed a method of static parameter identification from incomplete static test data. The element stiffness parameters (area, moment of inertia, etc.) are identified by utilizing force and displacement measurements on a subset of the global degrees of freedom. This method was successful for identifying element stiffness parameters for two dimensional truss and beam elements, including element failure. This method also considers the effects of errors in the force and displacement measurements. A Monte Carlo analysis is used to determine the effect of measurement errors on the identified stiffness parameters before the actual measurements are made. Sanayei and Schmolze [16] presented an expert system for static parameter identification

using the algorithm from reference [15].

The purpose of this report is to extend the parameter identification algorithm described in references [15] and [16] to recover the constitutive matrix of three dimensional orthotropic plate bending elements by using incomplete static test data. The identified constitutive matrix is used to locate the damaged areas of the reinforced concrete pier deck.

2.0 STATIC PARAMETER IDENTIFICATION

Static parameter identification (Ref. [15], [16], and [17]) of structural systems based on static force and displacement measurements, has become very important as researchers attempt to quantify changes in unknown parameters (i.e., area, moment of inertia, plate bending stiffness, . . .) of components of critical structural systems to determine if there has been any degradation or failure of these structural components.

The algorithm described is parameter identification of a linear elastic structure subjected to static loads. The objective is to perform parameter identification at the structural element level by measuring a set of applied static forces and induced displacements on a subset of the degrees of freedom used to define the finite element model of the structure.

In the finite element model, the topology of the structure, the behavior of the elements, and the boundary conditions are specified at the outset. The only unknown feature of the structural model is some or all of the parameters used to describe the structural components, for example cross sectional area, moment of inertia, or plate bending stiffness.

In the static parameter identification algorithm, an iterative procedure is used for automatically adjusting the structural element stiffness parameters, in order to improve the comparison between the measured and the theoretical response in an optimal way. Throughout the iterations, all the properties of the

stiffness matrix such as element connectivity, positive definiteness, symmetry, equilibrium, and bandedness are automatically preserved.

2.1 Static Parameter Identification Algorithm

Static parameter identification is based on the finite element method of static structural analysis, which relates forces to displacements in the usual way

$$\{f\} = [K]\{u\} \tag{2.1}$$

where $\{f\}$ is the vector of applied forces, $\{u\}$ is the vector of measured displacements, and $[K]$ is the global stiffness matrix, which relates forces and displacements.

For static parameter identification two independent stiffness matrices are formulated. $[k^a]$ is the analytical stiffness matrix derived from the finite element model. $[k^m]$ is the measured stiffness matrix derived from force and displacement measurements. Both are square matrices of the same size. Since only a subset of the degrees of freedom is measured, $[k^a]$ and $[k^m]$ are both smaller than the global stiffness matrix. To derive the analytical stiffness matrix from the global stiffness matrix, equation (2.1) is partitioned into measured and unmeasured portions

$$\left\{ \begin{array}{c} f^m \\ \hline f^u \end{array} \right\} = \left[\begin{array}{c|c} k_{mm} & k_{mu} \\ \hline k_{um} & k_{uu} \end{array} \right] \left\{ \begin{array}{c} u^m \\ \hline u^u \end{array} \right\} \tag{2.2}$$

It is assumed that there are no forces applied to the unmeasured degrees of freedom thus

$$\{f^u\}=\{0\} \tag{2.3}$$

Equation (2.2) can be simplified to

$$\{f^m\} = [k^s]\{u^m\} \tag{2.4}$$

where

$$[k^s] = [k_{mm}] - [k_{mu}][k_{uu}]^{-1}[k_{um}] \tag{2.5}$$

It should be noted that $[k^s]$ is a nonlinear function of the stiffness parameters because of the inversion of $[k_{uu}]$.

Measured static forces and displacements are obtained over a limited subset of the total number of the degrees of freedom used to define the finite element model. The force displacement relation obtained from the measurements may be written as

$$\{f^m\} = [k^m]\{u^m\} \tag{2.6}$$

Points on the structure where measurements are taken are chosen to match specific degrees of freedom in the finite element model. A unit force is applied to each degree of freedom selected and displacements are measured at all the chosen degrees of freedom. All the measured displacements can be arranged column-wise in a measured displacement matrix defined as $[u^m]$. The unit

forces can be arranged column-wise in a measured force matrix defined as $[f^m]$, which is obviously equivalent to the identity matrix, [I]. (If it is not possible to conduct the tests with unit loads, each load and corresponding displacements can be normalized for a linear elastic structure.) From these matrices the following relationship can be established

$$[f^m] = [k^m][u^m] \tag{2.7}$$

Since the measured force matrix is equivalent to the identity matrix,

$$\{f^m\} \equiv [I] \tag{2.8}$$

then

$$[I] = [k^m][u^m] \tag{2.9}$$

and

$$[k^m] = [u^m]^{-1} \tag{2.10}$$

According to the Maxwell-Betti reciprocity theorem for linear elastic structure, the matrix $[u^m]$ should be a symmetric matrix if the force and displacement measurements contain no error. If the force and displacement measurements are reasonably accurate, the

matrix $[k^m]$ may give an accurate representation of the measured substructure stiffness even though it may not be symmetric. If this is the case, the effect of measurement errors can be reduced by using $\frac{1}{2}([u^m]+[u^m]^T)$ instead of $[u^m]$.

For the same substructure from which $[k^m]$ is derived, the corresponding analytical stiffness matrix $[k^a(p)]$ is obtained by using the finite element method as described previously. The stiffness parameters $\{p\}$, in $[k^a(p)]$ are adjusted to match $[k^m]$ on an element by element basis.

To make this comparison all the independent elements of $[k^m]$ and $[k^a(p)]$ are assembled into vectors $\{k^m\}$ and $\{k^a(p)\}$, respectively. Since both $[k^m]$ and $[k^a(p)]$ are symmetric matrices, the size of vectors $\{k^m\}$ and $\{k^a(p)\}$ is equal to the number of elements in the upper triangle of either of these two matrices. The vectors $\{k^m\}$ and $\{k^a(p)\}$ can now be compared directly; the parameters $\{p\}$ can be adjusted on an element by element basis to equate the analytical and measured responses of the structural system.

In order to adjust the unknown parameters $\{p\}$ in $\{k^a(p)\}$, a first order Taylor series expansion is used to linearize the vector $\{k^a(p)\}$, which as shown previously is nonlinear function of the unknown parameters $\{p\}$.

$$\{k^a(p+\Delta p)\} \approx \{k^a(p)\} + [\frac{\partial}{\partial p}\{k^a(p)\}]\{\Delta p\} \qquad (2.11)$$

$[\frac{\partial}{\partial p}\{k^a(p)\}]$ is a matrix of partial derivatives, which is defined as

the sensitivity matrix [S(p)] of the system. Equation (2.11) can be written as

$$\{k^a(p+\Delta p)\} \approx \{k^a(p)\} + [S(p)]\{\Delta p\} \tag{2.12}$$

where

$$[S(p)] = [\frac{\partial}{\partial p} \{k^a(p)\}] \tag{2.13}$$

Differentiating Equation (2.5) with respect to each parameter will give us the coefficients for the sensitivity matrix.

$$[S(p)] = \frac{\partial}{\partial p_i}[k^a(p)] = \frac{\partial}{\partial p_i}\left[[k_{mm}] - [k_{mu}][k_{uu}]^{-1}[k_{um}]\right]$$

$$[S(p)] = \frac{\partial}{\partial p_i}[k_{mm}] - \left[\frac{\partial}{\partial p_i}[k_{mu}]\right][k_{uu}]^{-1}[k_{um}] +$$

$$[k_{mu}][k_{uu}]^{-1}\left[\frac{\partial}{\partial p_i}[k_{uu}]\right][k_{uu}]^{-1}[k_{um}] - [k_{mu}][k_{uu}]^{-1}\left[\frac{\partial}{\partial p_i}[k_{um}]\right] \tag{2.14}$$

The error induced by a first order Taylor series to represent $\{k^a(p+\Delta p)\}$ is the performance error.

$$\{E(\Delta p)\} = \{k^m\} - \{k^a(p+\Delta p)\}$$
$$= \{k^m\} - \{k^a(p)\} - [S(p)]\{\Delta p\} \tag{2.15}$$

The performance error vector can be represented by a scalar

quantity defined as

$$J(\Delta p) = \{E(\Delta p)\}^T\{E(\Delta p)\} \tag{2.16}$$

In order to minimize the scaler performance error $J(\Delta p)$, the partial derivative of the error function with respect to the parameter is set equal to zero, $\partial J(p)/\partial\{p\} = \{0\}$. Thus, the following relationship can be derived.

$$[S(p)]\{\Delta p\} = \{k^m\} - \{k^a(p)\} = \{\Delta k(p)\} \tag{2.17}$$

In general, the number of unknown parameters may be less than the number of independent measurements, in which case the sensitivity matrix in equation (2.17) will not be square. Then the method of least squares can be used to compute the unknown parameters for each iteration.

$$\{\Delta p\} = ([S(p)]^T[S(p)])^{-1}[S(p)]^T\{\Delta k(p)\} \tag{2.18}$$

Equation (2.18) can be used to set up an iterative procedure for static parameter identification.

$$\{p\}^{i+1} = \{p\}^i + ([S(p)]^T[S(p)])^{-1}[S(p)]^T\{\Delta k(p)\} \tag{2.19}$$

2.1 Additional Concepts in Static Parameter Identification

In order to successfully and efficiently use the static parameter identification algorithm, several guidelines that are useful in making a physical interpretation of the mathematical phenomena observed during the execution of the static parameter identification algorithm are briefly noted here. For more details see reference [17].

To deduce a unique set of parameters from a given set of measurements, the number of independent measurements must be greater than or equal to the number of unknown parameters. If the above condition is not satisfied, there will be an infinite number of parameters that satisfy the measurements.

The existence of a solution to equation (2.17) is determined by using elementary row operations to reduce the augmented matrix $[S(p), \Delta k(p)]$ to row echelon form. The number of nonzero rows in the row echelon form of a matrix is called the rank of a matrix. If the rank is less than the number of unknown parameters the system of equations is singular, and can not be solved uniquely. Also, in each nonzero row of the row echelon form of $[S(p)]$, the existence of more than one nonzero number represents a linear dependency between parameters corresponding to these columns. The number of linear dependencies in the sensitivity matrix reduces the number of linearly independent measurements, thus reducing the number of identifiable parameters.

The linear dependencies develop for several reasons. First, each finite element is capable of transferring only limited amount

of information. Second, the reduced analytical stiffness matrix may be banded because of the topology of the corresponding substructure. In this case, the analytical stiffness vector, $\{k^a(p)\}$, contains at least one inherent zero. This causes a row of zeros in the sensitivity matrix, which reduces its rank by one.

The sensitivity matrix may be ill-conditioned (nearly singular) thus, solving the system of equations with an ill-conditioned sensitivity matrix results in numerical problems, which leads to an erroneous $\{\Delta p\}$. This ill-conditioning of the sensitivity matrix may cause the algorithm to diverge in a case where there are no apparent linear dependencies. However, the ill-conditioning of the sensitivity matrix may be determined by computing the eigenvalues of the sensitivity matrix. If some of the eigenvalues are very close to zero, or if the condition number (the condition number is the ratio of the largest and smallest eigenvalues) of the sensitivity matrix is very high, the sensitivity matrix is ill-conditioned. In this case, either the number of unknown parameters should be reduced, or the number of measurements should be increased. Changing the subset of degrees of freedom measured for the given set of unknown parameters also may remove the ill-conditioning.

3.0 THREE DIMENSIONAL ISOPARAMETRIC ELEMENT

The basic procedure of isoparametric element formulation is to express the element coordinates and displacements in terms of interpolation functions (also called shape functions) using the natural coordinate system of the element. The three dimensional isoparmetric developed in this section uses tri-linear interpolation functions, and a 2 x 2 x 2 lattice of integration.

3.1 Isoparametric Element Formulation

Consider a general three dimensional isoparametric element (see Fig. 3.1) with local coordinates X, Y, and Z, and a parent element with natural coordinate system r, s, and t; the coordinate interpolation functions (Ref. [18]) are:

$$\begin{aligned} X(r,s,t) &= \Sigma\, h_i x_i \\ Y(r,s,t) &= \Sigma\, h_i y_i \\ Z(r,s,t) &= \Sigma\, h_i z_i \end{aligned} \qquad (3.1)$$

where X, Y, and Z are the coordinates at any point of the element, and x_i, y_i, and z_i, i = 1 to 8 are the coordinates of the eight element nodes. The interpolation functions h_i are tri-linear functions defined in the natural coordinate system (r, s, and t) of the parent element (Fig 3.1), where r, s, and t vary between -1 and +1. The tri-linear interpolation functions are

$$
\begin{aligned}
h_1 &= (1+r)(1+s)(1+t)/8 \\
h_2 &= (1-r)(1+s)(1+t)/8 \\
h_3 &= (1-r)(1-s)(1+t)/8 \\
h_4 &= (1+r)(1-s)(1+t)/8 \\
h_5 &= (1+r)(1+s)(1-t)/8 \\
h_6 &= (1-r)(1+s)(1-t)/8 \\
h_7 &= (1-r)(1-s)(1-t)/8 \\
h_8 &= (1+r)(1-s)(1-t)/8
\end{aligned}
\tag{3.2}
$$

In isoparametric formulation the element displacements are interpolated from the same tri-linear interpolation functions used for the geometry, thus

$$
\begin{aligned}
U(r,s,t) &= \Sigma\, h_i u_i \\
V(r,s,t) &= \Sigma\, h_i v_i \\
W(r,s,t) &= \Sigma\, h_i w_i
\end{aligned}
\tag{3.3}
$$

where U, V, and W are the local element displacements at any point of the element, and u_i, v_i, and w_i, $i = 1, \ldots 8$, are the corresponding element displacements at the nodes.

To evaluate the stiffness matrix of the element, the strain-displacement transformation matrix [B] must be calculated. The element strains are derived from the derivatives of the element displacements with respect to the local coordinate system. The element displacements are derived from (3.3), thus it is necessary

to relate the x, y, and z derivatives to the r, s, and t derivatives. This can be accomplished by finding the inverse of the Jacobian matrix [J]. The Jacobian operator relates the natural coordinate derivatives to the local coordinate derivatives in the following form.

$$[J] = \begin{bmatrix} \partial x/\partial r & \partial y/\partial r & \partial z/\partial r \\ \partial x/\partial s & \partial y/\partial s & \partial z/\partial s \\ \partial x/\partial t & \partial y/\partial t & \partial z/\partial t \end{bmatrix} \tag{3.4}$$

In matrix notation,

$$\begin{Bmatrix} \partial/\partial r \\ \partial/\partial s \\ \partial/\partial t \end{Bmatrix} = [J] \begin{Bmatrix} \partial/\partial x \\ \partial/\partial y \\ \partial/\partial z \end{Bmatrix} \tag{3.5}$$

Thus, to find the inverse relationship between the x, y, and z derivatives and the r, s, and t derivatives,

$$\begin{Bmatrix} \partial/\partial r \\ \partial/\partial s \\ \partial/\partial t \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \partial/\partial x \\ \partial/\partial y \\ \partial/\partial z \end{Bmatrix} \tag{3.6}$$

This relationship requires that the inverse of [J] exists. This inverse will exist provided there is a one to one correspondence between the local and the parent coordinates of the element. For a rectangular brick element, similar to the one used in the finite element model of the Naval Pier Deck, a one to one correspondence

exists. The inverse of the Jacobian will not exist only if the element is highly distorted or folds back on itself, then the one to one correspondence between the local and the natural coordinates does not hold.

The strain-displacement transformation matrix [B] can be constructed from equations (3.3) and (3.6) to relate the strains to the element displacements, in matrix form

$$\{\epsilon\} = [B]\ \{u\} \tag{3.7}$$

where {u} is the vector containing the nodal point displacement of the element, {ϵ} is the vector containing the element strains, and [B] is the strain-displacement transformation matrix.

The element stiffness matrix corresponding to the local element degrees of freedom is then

$$[K] = \int_V [B]^T[C][B]\ dV \tag{3.8}$$

Noting that [B] is a function of the natural coordinates r, s, and t, the integral over the volume can be expressed as

$$[K] = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} [B]^T[C][B]\ \det[J]\ dr\ ds\ dt \tag{3.9}$$

Where det[J] is the determinant of the Jacobian operator in (3.4) and [C] is the constitutive matrix for a three dimensional state of stress for an isotropic, as listed in Ref. [18].

In general, the above integral cannot be explicitly determined. Thus, numerical integration in three dimensions is employed to derive the element stiffness matrix for the three dimensional isoparametric element. A 2 x 2 x 2 lattice of integration utilizing Gaussian quadrature is used to evaluate the integral (3.9) for the Navy Pier Deck Model. Thus, in numerical format the stiffness matrix in (3.9) is evaluated as

$$[F] = [B]^T[C][B] \det[J] \tag{3.10}$$

$$[K] = \sum_{ijk} \alpha_{ijk} [F]_{ijk} \tag{3.11}$$

where $[F]_{ijk}$ is the matrix [F] evaluated at the sampling points r_i, s_j, and t_k, and the α_{ijk} (Ref. [18]) are the given weighting factors, which depend on the values of r_i, s_j, and t_k. The sampling points and the weighing factors are chosen to obtain the maximum accuracy for the integration. The abscissae and weight coefficient of the Gaussian quadrature formula can be found in Ref. [19].

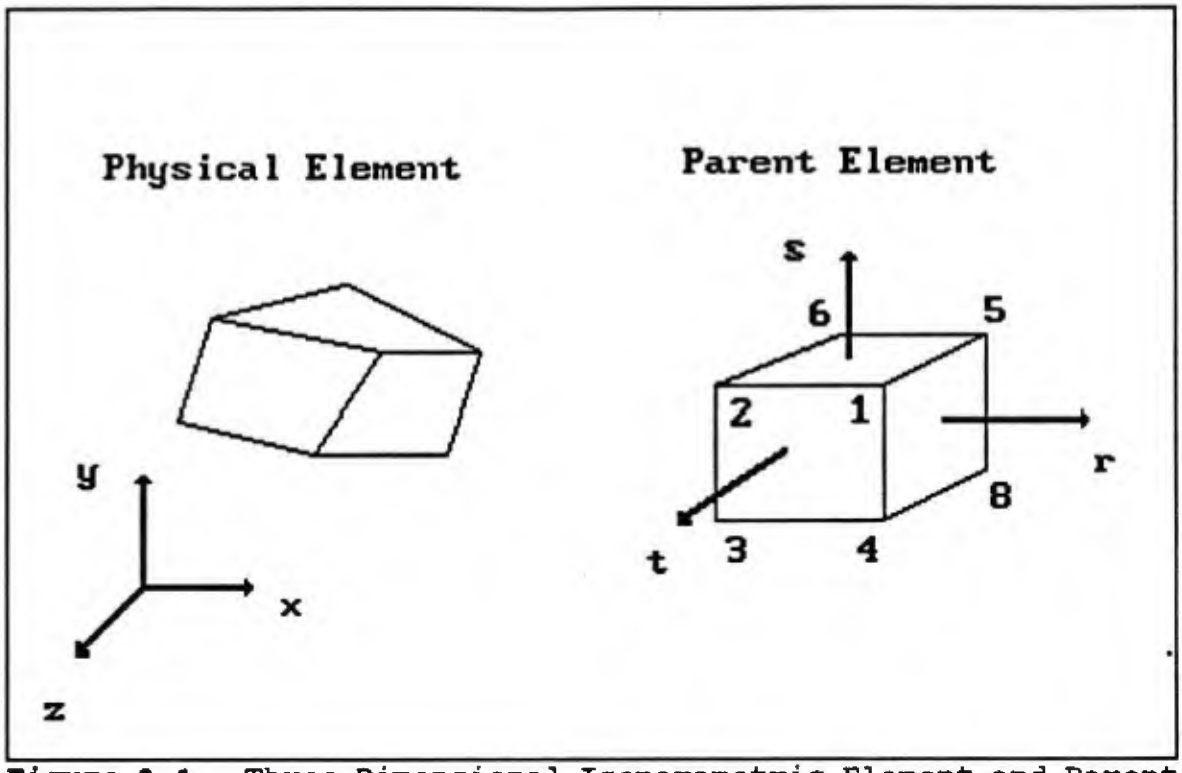

Figure 3.1 - Three Dimensional Isoparametric Element and Parent Element

3.2 Illustrative Numerical Examples

In this section, several numerical examples are presented to illustrate the behavior of the three dimensional isoparmetric element. A simple cantilever beam model is subjected to axial load, bending, and torsion; for comparison with the analytical solutions.

The three dimensional beam was generated with the same dimensions as the beams of the Navy Bridge Deck model (See Fig 3.2). The material properties were taken to be those of 4000 psi concrete, modulus of elasticity, E = 3605 ksi, and Poisson's ratio, ν = 0.19. The finite element mesh used to model this three dimensional beam is identical to the finite element used for parameter identification of the Navy Bridge model.[1]

The first example is a simple axial stretch. A twenty kip axial load was applied to the end of the cantilever by placing four 5 kip loads at the end nodes of the beam in the positive X - direction. The analytical displacement is 0.002774 in. The finite element solution was 0.00277. In this case, the finite element solution matches the analytical solution.

The second example is a beam bending mode. A twenty kip load was applied to the tip of the cantilever by placing four 5 kip loads at the tip nodes of the beam in the negative Z - direction. The analytical solution for a Timoshenko beam with a 5/6 shear correction factor is 0.4074 in. The finite element solution was

[1]See Chapter Five of this Report

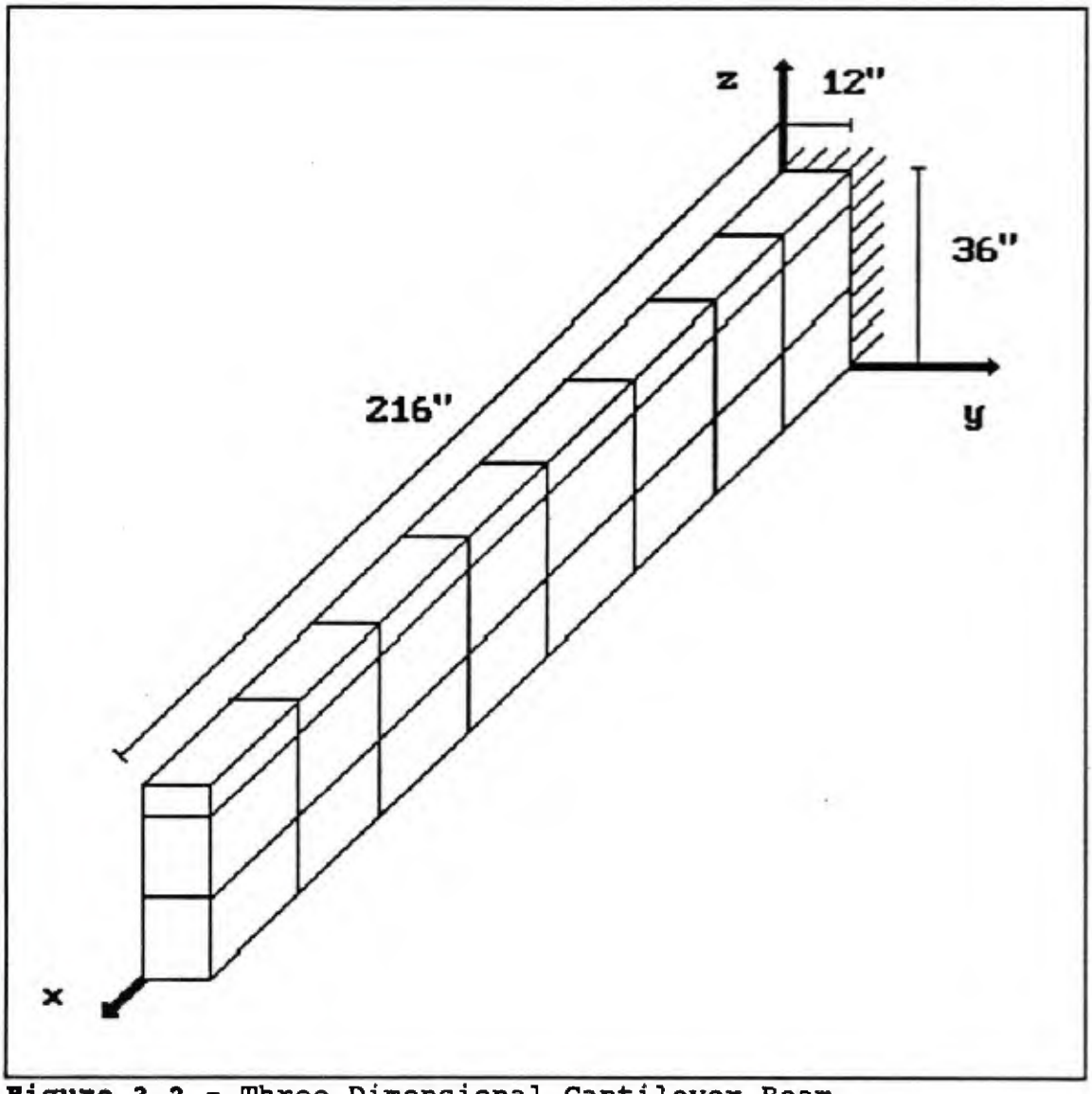

Figure 3.2 - Three Dimensional Cantilever Beam

0.3250 in. In this case, the linear interpolation functions induce error in the finite element solution. However, the finite element solution is still a reasonably accurate model of the cantilever beam.

The third example is a concentrated torque on the end of the beam. A 720 in-kip concentrated torque was applied to the end of the beam. The analytical solution for the angle of twist is 0.3596 degrees (Ref. [20]). The finite element solution in 0.3167 degrees. Similar to the beam bending example, the finite element solution contains error. However, the finite element solution is still a reasonable model of the physical behavior of the beam.

The results of these three examples indicate the suitability of the three dimensional isoparametric element to model the behavior of a three dimensional beam. The finite element solutions are in all cases reasonable comparisons with the analytical solution. The three dimensional isoparametric element should prove suitable to model the beams of the Navy Pier Deck.

4.0 A THREE DIMENSIONAL FINITE ELEMENT FOR PLATE BENDING BASED ON HOURGLASS CONCEPTS[1]

The three dimensional finite element developed here is based on the decomposition of its behavior into uniform strain, hourglass, and rigid body modes using underintegration techniques. The element contains six uniform strain modes, three uniform curvature modes, six rigid body modes, and nine secondary "hourglass" modes.

This summary of the finite element derivation is separated into two sections. The first section reviews the decomposition of rigid body, uniform strain, and hourglass modes by utilizing generalized coordinates and a one point integration technique. The second section reviews the superposition of classical plate bending theory on the element stiffness matrix by using five hourglass modes to eliminate five of the twelve spurious zero energy modes induced by the one point integration.

4.1 One Point Integration of a 3D Finite Element

Consider a three dimensional isoparametric element, with spatial (local) coordinates X, Y, and Z; and natural (parent)

---

[1] The following discussion and derivation is based on the work of Professor Richard B. Nelson at the University of California, Los Angeles, Ref. [21]. This paper (Ref. [21]) is currently being revised by Nelson. However, it contains details of the finite element derivation which are not presented in this chapter. The finite element presented here is a simplification of the finite element of Ref. [21]. The simplified element is a brick element with all interior angles equal to ninety degrees.

coordinates ξ, η, and ζ. The spatial positions inside the element are related to the nodal coordinates by the usual relations.

$$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = \sum_{i=1}^{8} N_i(\xi, \eta, \zeta) \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix} \tag{4.1}$$

where the isoparametric interpolation functions $N_i(\xi, \eta, \zeta)$ are the tri-linear functions, as listed in Eq. (2) of Ref. [21]. Displacements U, V, and W are also related to the nodal displacements by the same interpolation functions $N_i(\xi, \eta, \zeta)$.

The derivation of the plate bending element is based on the separation of the uniform strain, rigid body, and hourglass modes. To separate the physics of the element, it is useful to consider the set of generalized displacements of the element, q_i for i = 1 to 24, or in vector form $\{q\}$. The definition of the generalized coordinates is presented on page 11 of Ref. [21].

The element displacements in the local coordinate system, u_i, v_i, w_i, for i = 1 to 8, or in vector form $\{\delta\}$, can be related to the generalized displacements through the mapping matrix [A], in matrix form

$$\{\delta\} = [A]\{q\} \tag{4.2}$$

and

$$\{q\} = [A]^{-1}\{\delta\} \tag{4.3}$$

where $\{\delta\}$ is the vector containing the twenty-four nodal displacements, $\{q\}$ is the vector containing the twenty-four

generalized displacements, and [A] is the matrix which relates the two. For a complete derivation of the [A] and $[A]^{-1}$ matrices refer to pages 12 and 14 of Ref. [21].

Once the vector of generalized displacements {q} is formed the different modes of physical behavior can be separated. The generalized displacements can be decomposed into the rigid body and uniform strain modes (q_1 through q_{12}), and the hourglass modes (q_{13} through q_{24}). Refer to page 11 of Ref. [21].

The stiffness matrix for the three dimensional continuum element requires the strain displacement transformation matrix, [B]. For the purpose of separating the uniform strain, rigid body and hourglass modes, it is convenient to represent the strain displacement transformation matrix in terms of the generalized displacements {q}. Thus, in matrix form

$$\{\epsilon\} = [B]\{q\} \tag{4.4}$$

where {ϵ} is a (6 x 1) vector containing the element strains, [B] is a (6 x 24) strain displacement transformation matrix, and {q} is a (24 x 1) vector of generalized displacements. For more details refer to Eq. (27a) of Ref. [21].

The strain displacement relationship can be partitioned to separate the rigid body, uniform strain, and hourglass modes. In matrix form

$$\{\epsilon\} = [B_o \mid B_{HG}]\{q\} \tag{4.5}$$

where $[B_o]$ contains the uniform strain and rigid body modes, and $[B_{HG}]$ contains the hourglass modes. For a complete derivation of the above equation refer to page 15 of Ref. [21].

To form the element stiffness matrix of the three dimensional element in terms of the generalized displacements {q} it is necessary to evaluate the volume integral

$$[k]_{qq} = \int_V [B]^T[C][B]\, dV \tag{4.6a}$$

$$[k]_{qq} = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} [B]^T[C][B]\, \det[J]\, d\xi\, d\eta\, d\zeta \tag{4.6b}$$

In equation (4.6), matrix [C] contains the elastic coefficients as defined in ref. [18], matrix [B] describes the strain displacement transformation as defined in ref. [21], and det[J] is the determinant of the Jacobian matrix. By partitioning the strain displacement matrix [B] as shown in equation (4.5), equation (4.6) may be written in the following partitioned form

$$[k]_{qq} = \left[\begin{array}{c|c} k_{11} & k_{12} \\ \hline k_{21} & k_{22} \end{array}\right]_{qq} \tag{4.7}$$

where the submatrices are

$$[k_{11}] = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} [B_o]^T[C][B_o]\, \det[J]\, d\xi\, d\eta\, d\zeta \tag{4.8a}$$

$$[k_{12}] = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} [B_o]^T[C][B_{HG}]\, \det[J]\, d\xi\, d\eta\, d\zeta \tag{4.8b}$$

$$[k_{21}] = [k_{21}]^T \tag{4.8c}$$

$$[k_{22}] = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} [B_{HG}]^T[C][B_{HG}]\ \det[J]\ d\xi\ d\eta\ d\zeta \qquad (4.8d)$$

The evaluation of equation (4.8) by the use of a one point Gaussian quadrature formula at the integration point of $\xi=\eta=\zeta=0$, gives the following results,

$$[k_{11}]_{1pt} = 8\ \det[J_o][B_o]^T[C][B_o] \qquad (4.9a)$$

$$[k_{12}]_{1pt} = [k_{21}]_{1pt} = [0] \qquad (4.9b)$$

$$[k_{22}]_{1pt} = [0] \qquad (4.9c)$$

Equations (4.9a) and (4.9b) are correct if the element is a parallelpiped, which is the case for the finite element used to model the Navy Pier Deck. (For more details see pages 16 and 17 of Ref. [21].) However, equation (4.9c) which sets $[k_{22}]$, is obviously a gross error as at least there should be diagonal stiffness terms. The failure of the one point integration to account for the hourglass stiffnesses, results in an element stiffness matrix with twelve zero energy hourglass modes.

To compensate for the zero energy modes small diagonal stiffnesses can be added to the diagonals of rows 13 to 24. These small diagonal stiffnesses can be termed hourglass control stiffnesses. Thus, the final element stiffness matrix in the generalized coordinates has the form

$$[k]_{qq} = \left[\begin{array}{c|c} k_{11} & 0 \\ \hline 0 & \bar{k} \end{array}\right]_{qq} \qquad (4.10)$$

where $\bar{k}$ is a small stiffness added to the diagonal terms of rows 13 through 24. The element stiffness matrix in the local coordinates is then

$$[k] = [A]^{-T}[k]_{qq}[A]^{-1} \tag{4.11}$$

In summary, the stiffness matrix developed in this section utilizes a one point integration technique to decouple the uniform strain, rigid body, and hourglass modes. In the following section, the development of hourglass stiffness correction terms to emulate thin plate bending will be summarized.

4.2 Hourglass Stiffness for Thin Plate Bending

In addition to serving as a basis for developing hourglass stiffness, uniform strain, and rigid body modes, the use of the generalized coordinates {q}, may be used to construct a plate bending stiffness matrix. For more details consult pages 20 to 22 of Ref. [21].

The development of the plate bending stiffness matrix presented here is restricted to the case of a finite element which is simply a brick element with all interior angles equal to ninety degrees. In this case there are several simplifications to the algorithm developed in Ref. [21], which will be noted in this section.

Consider an element in the form of a quadrilateral as shown in figure 4.1. Let the rectangular cartesian (X, Y, Z) coordinate system be a local system, where x = y = z = 0 denotes the center of the element. Thus, x = -a, +a and y = -b, +b denote the side surfaces, and z = -h, +h denotes the top and bottom surfaces. It is important to note that in this particular case the local (X, Y, Z) and the natural (ξ, η, ζ) coordinate systems coincide. Thus, the Jacobian matrix is constant

$$[J] = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & h \end{bmatrix} \tag{4.12}$$

and the determinant of the Jacobian becomes

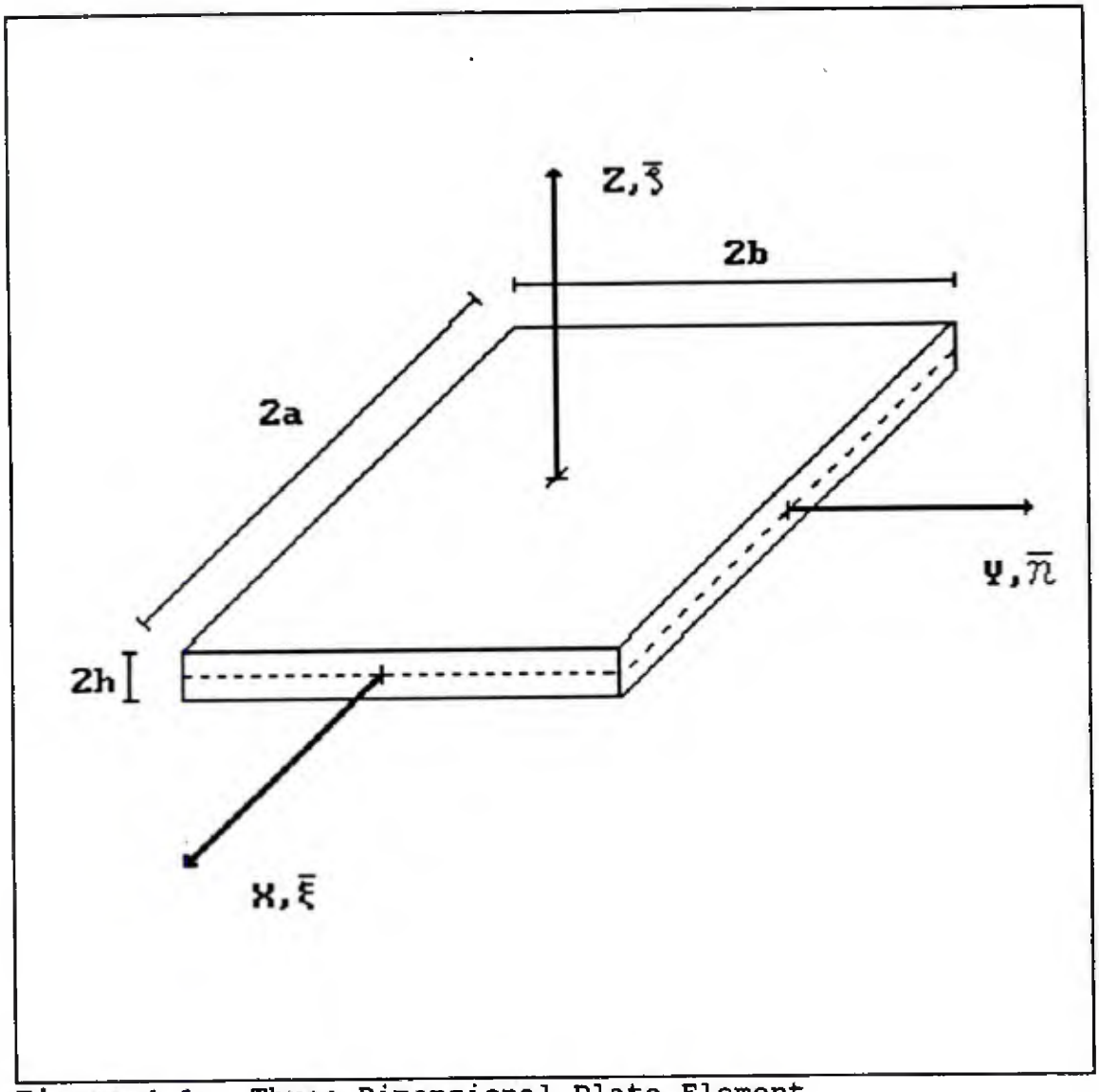

Figure 4.1 - Three Dimensional Plate Element

$$\det[J] = abh \tag{4.15}$$

Thus, the volume of this body is V = 8 det[J] = 8abh.

A relationship can be developed between the generalized coordinates q_{14}, q_{15}, q_{18}, q_{19}, and q_{21} and uniform bending curvature

in the natural coordinate system. For more details consult pages 20 to 22 of Ref. [21]. The resulting relationship between curvature and the generalized coordinates for the brick element is of the form

$$\begin{Bmatrix} \kappa_{\xi\xi o} \\ \kappa_{\eta\eta o} \\ 2\kappa_{\xi\eta o} \\ \bar{u}_{\xi} \\ \bar{u}_{\eta} \end{Bmatrix} = \begin{bmatrix} 0 & 1/ha & 0 & 0 & 0 \\ 0 & 0 & 1/hb & 0 & 0 \\ 0 & 0 & 0 & 0 & -2/ab \\ 1 & 0 & 0 & 0 & h/a \\ 0 & 0 & 0 & 1 & h/b \end{bmatrix} \begin{Bmatrix} q_{14} \\ q_{15} \\ q_{18} \\ q_{19} \\ q_{21} \end{Bmatrix} \qquad (4.16)$$

where $\kappa_{\xi\xi o}$, $\kappa_{\eta\eta o}$, and $\kappa_{\xi\eta o}$ are measures of uniform curvature in the parent coordinate system, and where $\bar{u}_{\xi}$ and $\bar{u}_{\eta}$ are two additional in surface hourglass motions necessary to complete the derivation.

Please note that equation (4.16) is a simplification of Eq. (55a) of Ref. [21]. In matrix form, the above relationship between curvature and the generalized coordinates can be expressed as

$$\{\kappa\} = [N]^{-1}\{\bar{q}\} \qquad (4.17)$$

The next step is to develop the augmented plate bending matrix of elastic coefficients, [D] in the parent coordinate system. Please note that for the special case of a brick element the parent coordinate system coincides with the local coordinate system. Thus, the mapping matrix [M] Eq. (63b) Ref. [21] is the identity matrix and is not necessary for this derivation. Thus, the [D] matrix is

$$[D] = \begin{bmatrix} D_{11} & D_{12} & 0 & 0 & 0 \\ D_{12} & D_{22} & 0 & 0 & 0 \\ 0 & 0 & D_{66} & 0 & 0 \\ 0 & 0 & 0 & k_{14} & 0 \\ 0 & 0 & 0 & 0 & k_{19} \end{bmatrix} \qquad (4.18)$$

where D_{11}, D_{12}, D_{22}, and D_{66}, are elastic coefficients and k_{14}, and k_{19} are small hourglass control stiffnesses. The small stiffnesses are simply used to control the hourglass motion, thus they are set to a small value, of the order of 1/100 of the plate bending stiffness.

For the purpose of modelling the reinforced concrete Navy Pier Deck, the following relations for the elastic coefficients are used. Once the values of the equivalent moments of inertia[2] for a reinforced concrete cross section are found, the plate bending stiffnesses for an orthotropic reinforced concrete plate can be computed from the following equations, as listed in Ref. [22].

$$\begin{aligned} D_{11} &= \{E_c/(1 - \nu_c^2)\} \times [I_{eq} \text{ X-direction}] \\ D_{22} &= \{E_c/(1 - \nu_c^2)\} \times [I_{eq} \text{ Y-direction}] \\ D_{12} &= \nu_c\sqrt{D_{11}D_{22}} \\ D_{66} &= \{(1 - \nu_c)/2\}\sqrt{D_{11}D_{22}} \end{aligned} \qquad (4.19)$$

where E_c = 3605 ksi and ν_c = 0.19 for 4000 psi concrete.

Once the [D] matrix is evaluated, the plate bending stiffness matrix $[k]_q$, may be computed.

---

[2] Please see Chapter 5 of this report.

$$[k]_{\bar{q}} = 4ab[N]^{-T}[D][N]^{-1} \qquad (4.20)$$

where $[k]_{\bar{q}}$ is the is the plate bending stiffness matrix, $[N]^{-1}$ is the mapping matrix between curvature and the five hourglass terms (q_{14}, q_{15}, q_{18}, q_{19}, and q_{21}), and [D] is the matrix of elastic coefficients of orthotropic plate bending.

The final plate bending stiffness matrix is obtained by combining the one point integration stiffness in Eq. (4.10) and the plate bending stiffness matrix in Eq. (4.20). The element stiffness matrix is formed by eliminating the small stiffness correction terms $\bar{k}$ in Eq. (4.10) in diagonal locations 14, 15, 18, 19, and 21; and then inserting the elements of the matrix $[k]_{\bar{q}}$ into locations with row (or column) numbers 14, 15, 18, 19, and 21 corresponding to element row (or column) numbers 1, 2, 3, 4, 5, respectively. Thus, the plate bending stiffness matrix, in {q} coordinates, has a form similar to equation (4.10), except that the 12 x 12 submatrix of small diagonal stiffnesses $[\bar{k}]$ is replaced with a 12 x 12 matrix $[k_b]$ which accounts for plate bending. Thus, the final 24 x 24 element stiffness matrix in the {q} coordinates is

$$[k]_q = \left[\begin{array}{c|c} k_{11} & 0 \\ \hline 0 & k_b \end{array}\right]_q \qquad (4.21)$$

To transform the element stiffness matrix $[k]_q$ into the local (X,

Y, Z) coordinates,

$$[k] = [A]^{-T}[k]_q[A]^{-1} \tag{4.22}$$

Please note that for the Navy Pier Deck problem, the local and the global coordinate systems coincide, thus, the matrix of direction cosines presented in Eq. (70a) of Ref. [21] is not necessary. Thus, the final form of the element stiffness matrix is presented in Eq. (4.22). The final element stiffness matrix will have uniform strain, rigid body displacement, and uniform curvature capabilities required for thin plate bending analysis.

The nine small stiffnesses used to control the hourglass motion in this element have no direct counterpart to the classical theory of plate bending, and thus, are taken to be quite small, of the order of 1/100 of the plate bending stiffnesses.

4.3 Illustrative Numerical Examples

To demonstrate the capabilities of the simplified plate bending element, several numerical examples will be presented; a cantilever beam, a one element plate with two adjacent edges simply supported and two adjacent edges free, a simply supported plate, and a fixed plate. For all the examples presented, except the fixed and simply supported plates, the material was assumed to be isotropic with a modulus of elasticity, E = 30,000 ksi and Poisson's ratio, ν = 0.3. For all examples presented in this section, the hourglass stiffness terms k_{14} and k_{19} were set equal to 0.05 Max[D_{11}, D_{22}]. The remaining hourglass stiffness terms $\bar{k}$ were set equal to 0.001 Max[D_{11}, D_{22}].

A short cantilever beam was generated using two elements to investigate the behavior of the element for three types of loading; axial load, pure bending due to a concentrated load, and pure bending due to a concentrated moment. Figure 4.2 shows the finite element model of the beam.

To compare the analytical solution for axial loads with the finite element solution, a four kip load was applied to the end of the cantilever by placing a one kip load in the positive Y-direction to each of the four end nodes. The analytical solution for axial deformation is δ = 0.00053 in, the finite element solution was δ = 0.00053 in. In this case, there is no difference between the finite element solution and the analytical solution.

To investigate bending behavior, a four kip load was

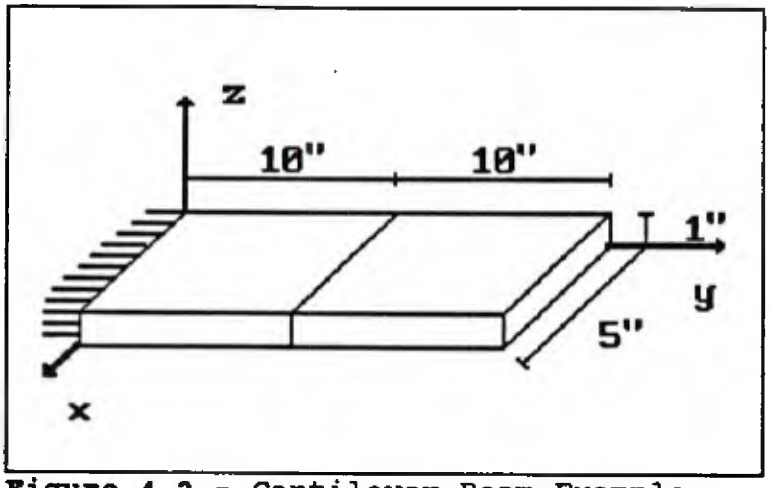

Figure 4.2 - Cantilever Beam Example

applied to the tip of the cantilever by placing a one kip load in the negative Z direction to each of the four tip nodes. The analytical solution for a Timoshenko beam with a 5/6 shear correction factor is is δ_L = -0.8550 in, the finite element solution was δ_L = -0.8014 in. In this case, the error induced by the finite element solution is expected as the finite element mesh is not fine enough to capture the behavior of the beam. However, it is important to note that even with a coarse mesh this element adequately models the beam.

This cantilever beam example was also used to compare the finite element solution with analytical solution for a concentrated moment at the tip of the beam. This type of loading will induce a constant curvature in the beam, thus, it is expected that the finite element solution will agree with the analytical solution. A 2 in - kip moment was applied to the end of the cantilever by placing one kip loads in the negative Y-direction to the top end nodes of the beam, and one kip loads in the positive Y-direction

to the bottom end nodes of the beam. The analytical solution for a 2 in - kip concentrated moment is δ = 0.032 in, the finite element solution was δ = 0.031999 in. In this case there is no error induced by the finite element solution.

A one element plate with two adjacent edges simply supported and two adjacent edges free (See Fig. 4.3) was generated to investigate plate twisting behavior.

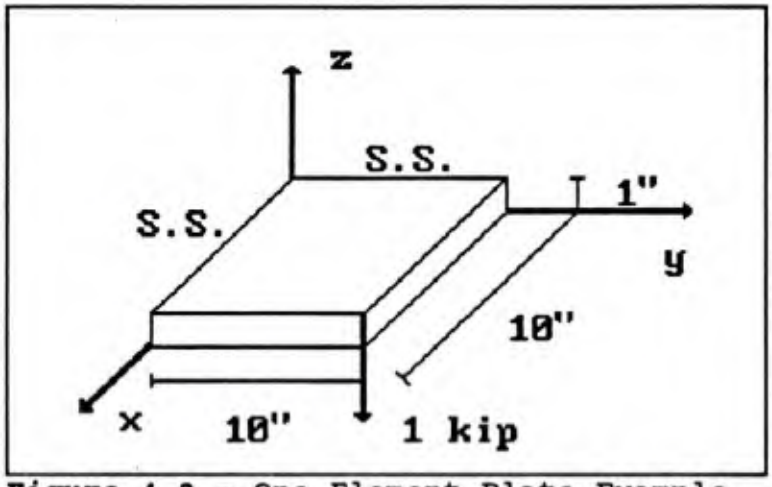

Figure 4.3 - One Element Plate Example

A one kip load was applied to the free corner by placing one half kip loads in the negative Z-direction to the top and bottom corner nodes. The analytical solution was derived by using energy methods assuming a bi-linear displacement field δ, where δ_o is the tip displacement.

$$\delta = \delta_o(x/10)(y/10) \qquad (4.23)$$

Thus, the XY partial derivative of the displacement field is

$$\delta_{,xy} = \delta_o/100 \tag{4.24}$$

The strain energy has the form (Ref. [22])

$$U = 1/2 \iint_A 2D\ (1-\nu)\ (\delta_{,xy})^2\ dA \tag{4.25}$$

where,

$$D = Eh^3/12(1-\nu^2) \tag{4.26}$$

Thus, integrating over the area

$$U = Eh^3\delta_o^{\,2}/1200(1+\nu) \tag{4.27}$$

The partial derivative of the strain energy with respect to the tip displacement is equal to the applied load, thus

$$\partial U/\partial \delta_o = 1 \text{ kip} \tag{4.28a}$$

$$\partial U/\partial \delta_o = Eh^3\delta_o/600(1+\nu) = 1 \text{ kip} \tag{4.28b}$$

Thus, solving equation (4.28b) for δ_o, the analytical solution is $\delta_o = -0.026$ inches at the corner of the plate. The finite element solution was $\delta = -0.026$ inches at the corner of the plate. This example illustrates that the element contains the bi-linear physical behavior, which is expected.

To investigate the behavior of a plate similar to the size of

the instrumented test span of the Navy Pier Deck model, a plate was generated with the dimensions as shown in figure 4.4. The thickness of the plate was taken to be 5 3/8 in. (the same as the instrumented slab of the Navy Pier Deck). The material properties were assumed to be isotropic with the modulus of elasticity of 4000 psi concrete, E = 3605 ksi, and Poisson's ratio, ν = 0.19. In all cases, the transverse shear terms were set to 1000 times their correct value to facilitate a comparison with classical plate bending theory.

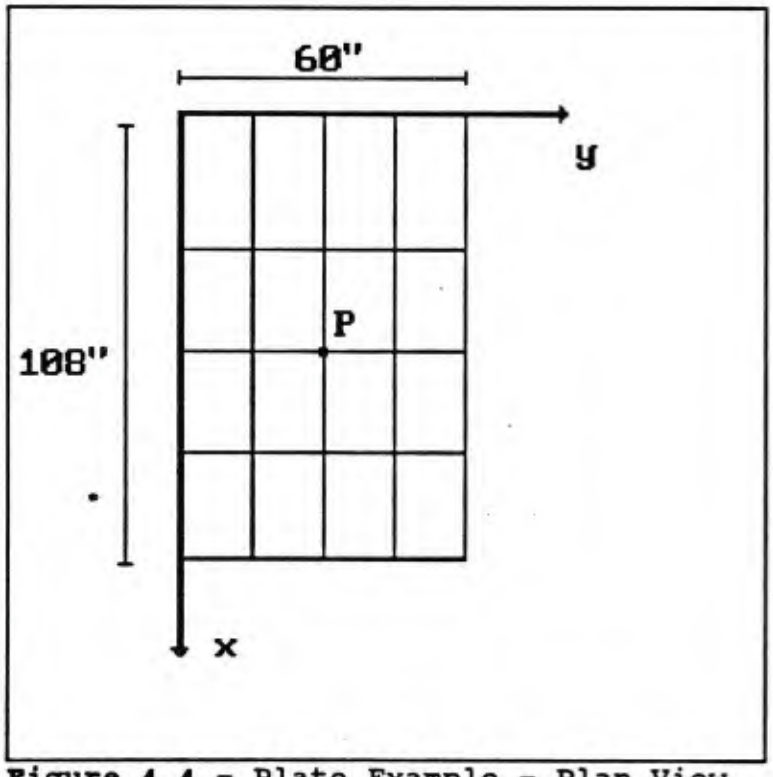

Figure 4.4 - Plate Example - Plan View

Two different boundary conditions were investigated; all four edges simply supported and fixed. A five kip load was applied to the center of the plate by applying a 2.5 kip load in the negative

Z-direction the top and bottom center nodes. In both cases, the analytical solution was computed from solutions presented in Ref. [22]. For the simply supported case, the analytical solution is 0.006025 in, the finite element solution was 0.006116 in. For this case, the error induced by the finite element solution is low. The finite element solution slightly over estimates the analytical solution, this can be corrected by adjusting the hourglass stiffness terms.

For the fixed case, the analytical solution is 0.002677 in, the finite element solution was 0.002119 in. The error induced by the finite element solution can be attributed to the size of the finite element mesh. However, it is important to note that even with a coarse mesh this element adequately models the fixed plate.

The examples indicate the suitability of this finite element (Ref. [21]) to model thin plate bending behavior. The performance of this finite element compares favorably with the analytical models of the examples presented in this section. Thus, this element should prove to be suitable to model the thin slab of the Navy Pier Deck.

5.0 FINITE ELEMENT MODEL

The purpose of this chapter is to describe the finite element model that is selected to model the Navy Pier Deck. Through numerical and analytical considerations a finite element model is selected to represent the physical model accurately. The two objectives of the analysis are to create a finite element model which accurately represents the behavior the physical model, and to limit the degrees of freedom of the finite element mesh to a number which can be manipulated within the core memory of the VAX 8550 at Tufts University.

This chapter is separated into four parts. The first section describes the decision to model only three spans of the bridge deck by employing the proper boundary conditions. The second section describes the selection of the size of the finite element mesh. The third section discusses the equivalent moments of inertia used in the finite element model to represent the cross sectional properties of reinforced concrete. The fourth section discusses the formulation of a super-element to condense out the degrees of freedom away from the instrumented test span.

5.1 Three Span Model

As illustrated by finite element analysis of the Navy Bridge Deck model a large concentrated load applied to the fourth span would have only a small effect on the displacements and stress

level of the third and fifth spans (Fig. 5.2). This behavior is due to the large stiffness of the beams supporting a relatively flexible slab. The carry over of stress from one span to the next is low; thus, it was concluded that to reduce modelling costs a three span model would be investigated.

The five span Navy Bridge Deck with a concentrated load at the center of the fourth span is a symmetric structure with a XY plane of symmetry. (See Fig. 5.1) Thus, the five span model can be modeled by one half of the structure by imposing the proper boundary conditions at the plane of symmetry. It is important to note that symmetry is not used in parameter identification, thus reducing the size of the finite element model is an important consideration.

A full five span bridge deck model (using XY-symmetry) was generated using an eight node three-dimensional isoparametric brick element (ANSYS element Stiff 45, quadratic displacement shapes and a 2x2x2 lattice of integration, Ref. [23]). A fifty kip (25 kips on the symmetric structure) concentrated load was applied to the fourth span of the model. As expected the third and fifth spans experienced a low stress level due to the high concentrated load on the forth span. See figures 5.1, 5.2 and 5.3 for the geometry plot, deflected shape, and von Mises stress contours of the ANSYS finite element model.

Several three span models were generated (also using XY-symmetry) to investigate the accuracy of portraying the behavior of the continuous five span model with a three span model by

applying the proper boundary conditions at the continuous edge. Three finite element models were generated using the same three-dimensional isoparametric element (ANSYS element Stiff 45).

The first model (Fig. 5.4) includes spans 3, 4 and 5, and the last three beams with the continuous edge fixed. The second finite element mesh (Fig. 5.7) includes spans 3, 4 and 5, and the last four beams with the base of beam three fixed. The third model (Fig. 5.10) includes spans 3, 4 and 5, the last three beams and one extra row elements (the elements which would be over beam three in the five span model) with the continuous edge fixed. The geometry plots, von Mises stress contours, and deflected shapes are presented for comparison with the five span finite element model in figures 5.4 to 5.12. A table which lists deflections along two midspan lines is presented for comparison of the four models.

Table 5.1

Comparison of Five Span Bridge Deck Model with Three Span Models

3-Sp1 - Three Spans with Edge Fixed (Fig. 5.4)
3-Sp2 - Three Spans with Beam Three (Fig. 5.7)
3-Sp3 - Three Spans with Extra Row of Elements (Fig. 5.10)

XY Cross-Section of Test Span, Z* = 108"

X-Coord	5-Sp	3-Sp1	% Error	3-Sp2	% Error	3-Sp2	% Error
84	-0.00190	-0.00190	-0.12	-0.00190	-0.01	-0.00190	0.07
94	-0.01358	-0.01354	-0.26	-0.01357	-0.04	-0.01360	0.18
104	-0.02906	-0.02899	-0.24	-0.02905	-0.04	-0.02911	0.17
114	-0.03838	-0.03829	-0.25	-0.03837	-0.04	-0.03845	0.18
124	-0.02891	-0.02881	-0.35	-0.02890	-0.06	-0.02899	0.26
134	-0.01340	-0.01333	-0.53	-0.01339	-0.09	-0.01345	0.38
144	-0.00194	-0.00195	0.26	-0.00194	0.03	-0.00194	-0.16

YZ Cross-Section of Test Span, X* = 114"

Z-Coord	5-Sp	3-Sp1	% Error	3-Sp2	% Error	3-Sp2	% Error
0	-0.00289	-0.00281	-3.02	-0.00288	-0.40	-0.00296	2.23
12	-0.00288	-0.00279	-2.94	-0.00287	-0.39	-0.00294	2.17
24	-0.00302	-0.00293	-2.82	-0.00300	-0.38	-0.00308	2.07
36	-0.00340	-0.00331	-2.55	-0.00339	-0.36	-0.00346	1.86
48	-0.00433	-0.00424	-2.06	-0.00431	-0.30	-0.00439	1.49
60	-0.00644	-0.00634	-1.42	-0.00642	-0.22	-0.00650	1.02
72	-0.01080	-0.01071	-0.86	-0.01079	-0.14	-0.01087	0.62
84	-0.01854	-0.01844	-0.51	-0.01852	-0.08	-0.01861	0.36
96	-0.02980	-0.02970	-0.32	-0.02978	-0.05	-0.02987	0.23
108	-0.03838	-0.03829	-0.25	-0.03837	-0.04	-0.03845	0.18

* X = 114" and Z = 108" is the point load position in Fig. 5.1.

As illustrated in table 5.1 and in the comparison of the deflected shapes and the von Mises stress contours, the three span model which includes the last four beams with the base of beam three fixed is a reasonably accurate model of the five span Navy bridge deck. It is also apparent that a three span model is a less computationally expensive than the full five span model.

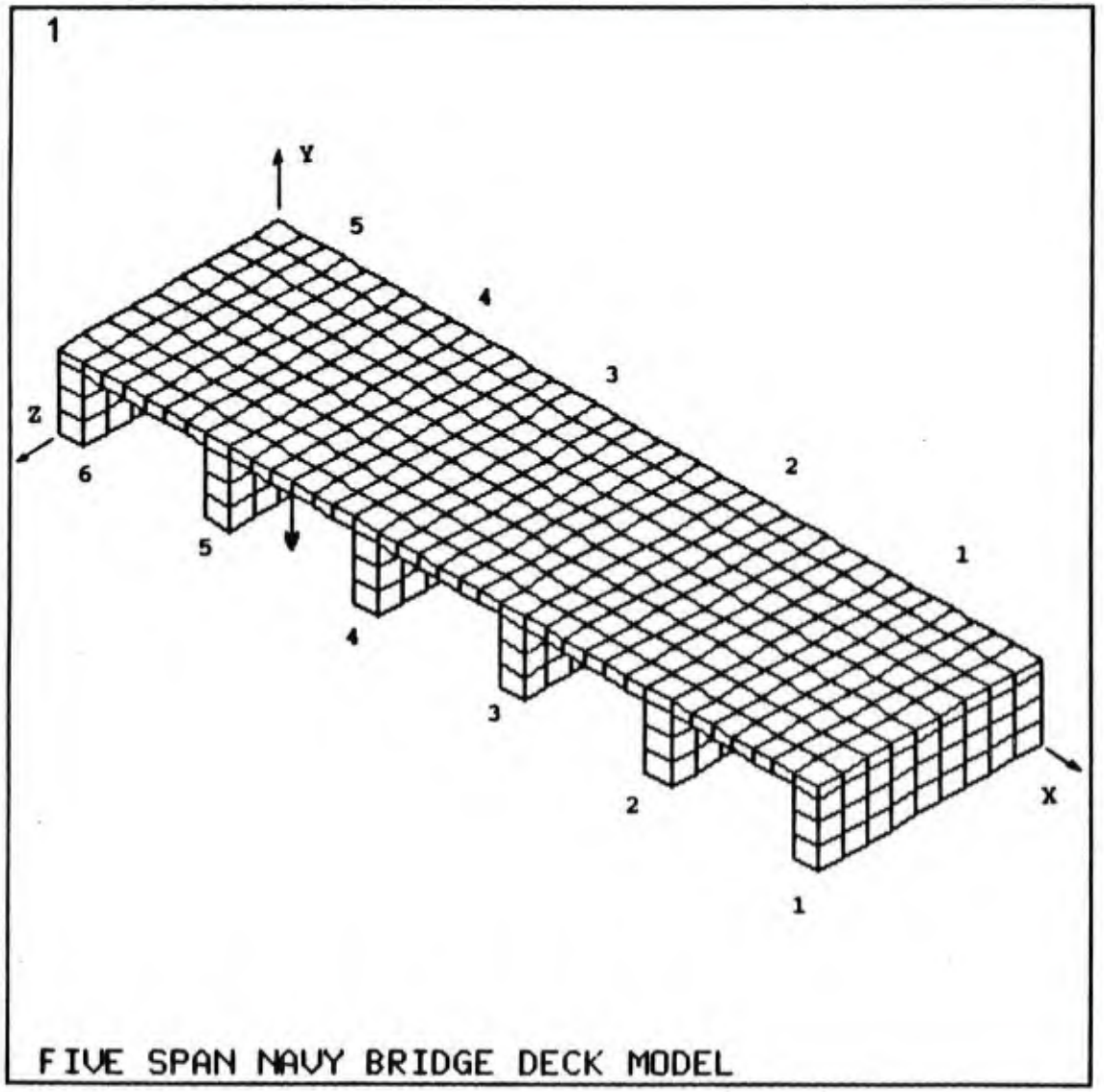

Figure 5.1 - Five Span Finite Element Model, Geometry Plot

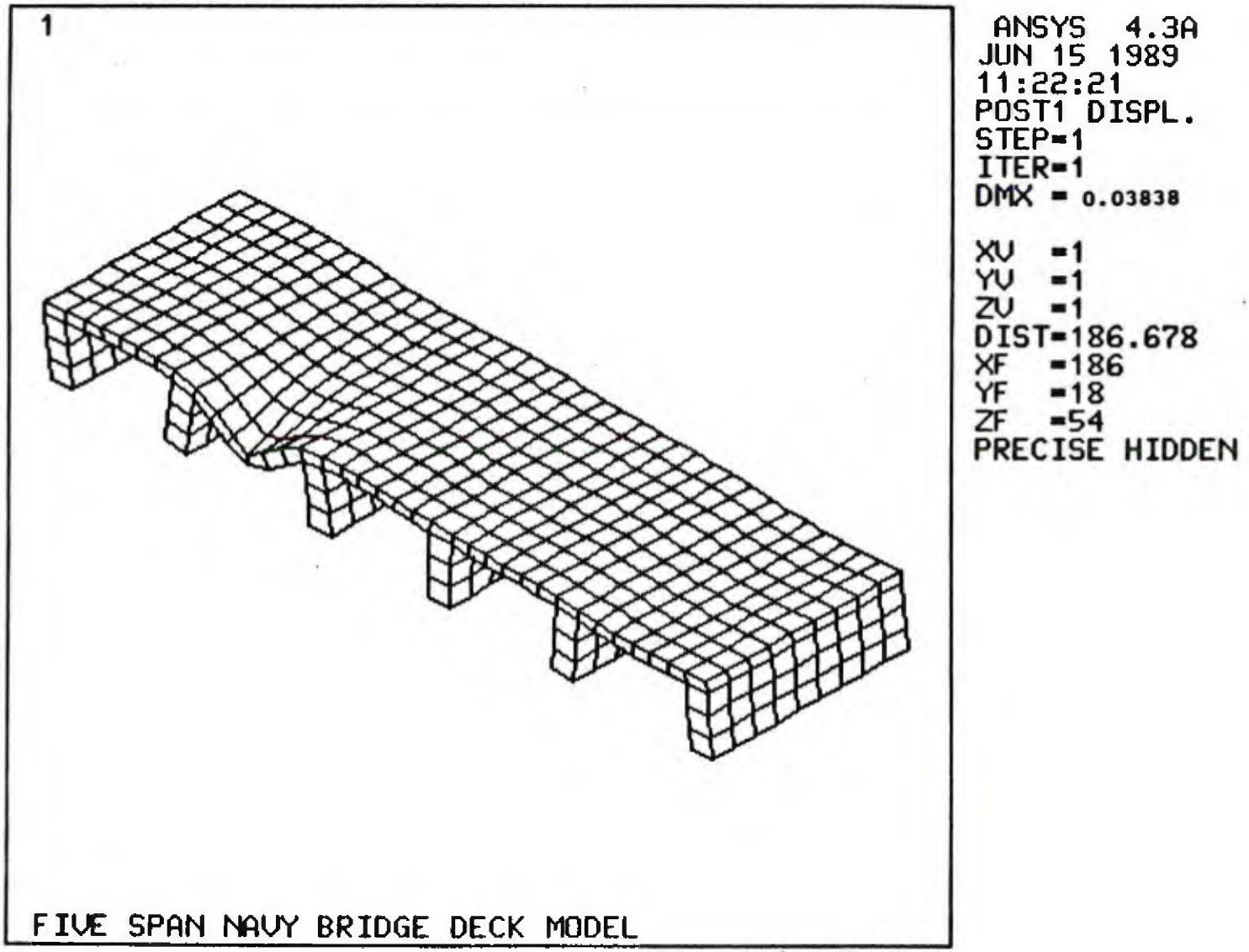

Figure 5.2 - Five Span Finite Element Model, Deflected Shape

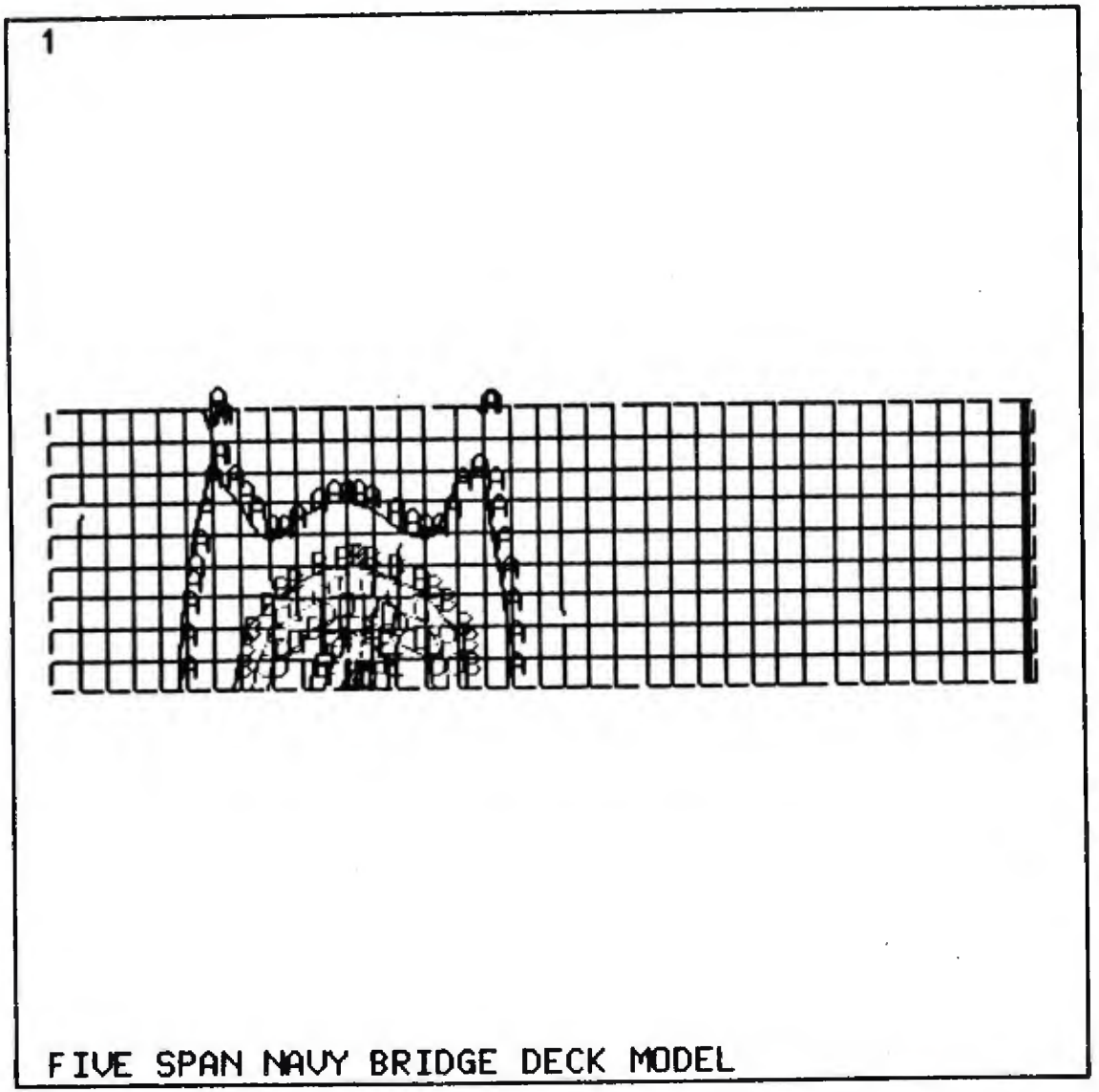

Figure 5.3 - **Five Span Finite Element Model, von Mises Stress Contours on the Top of the Slab**

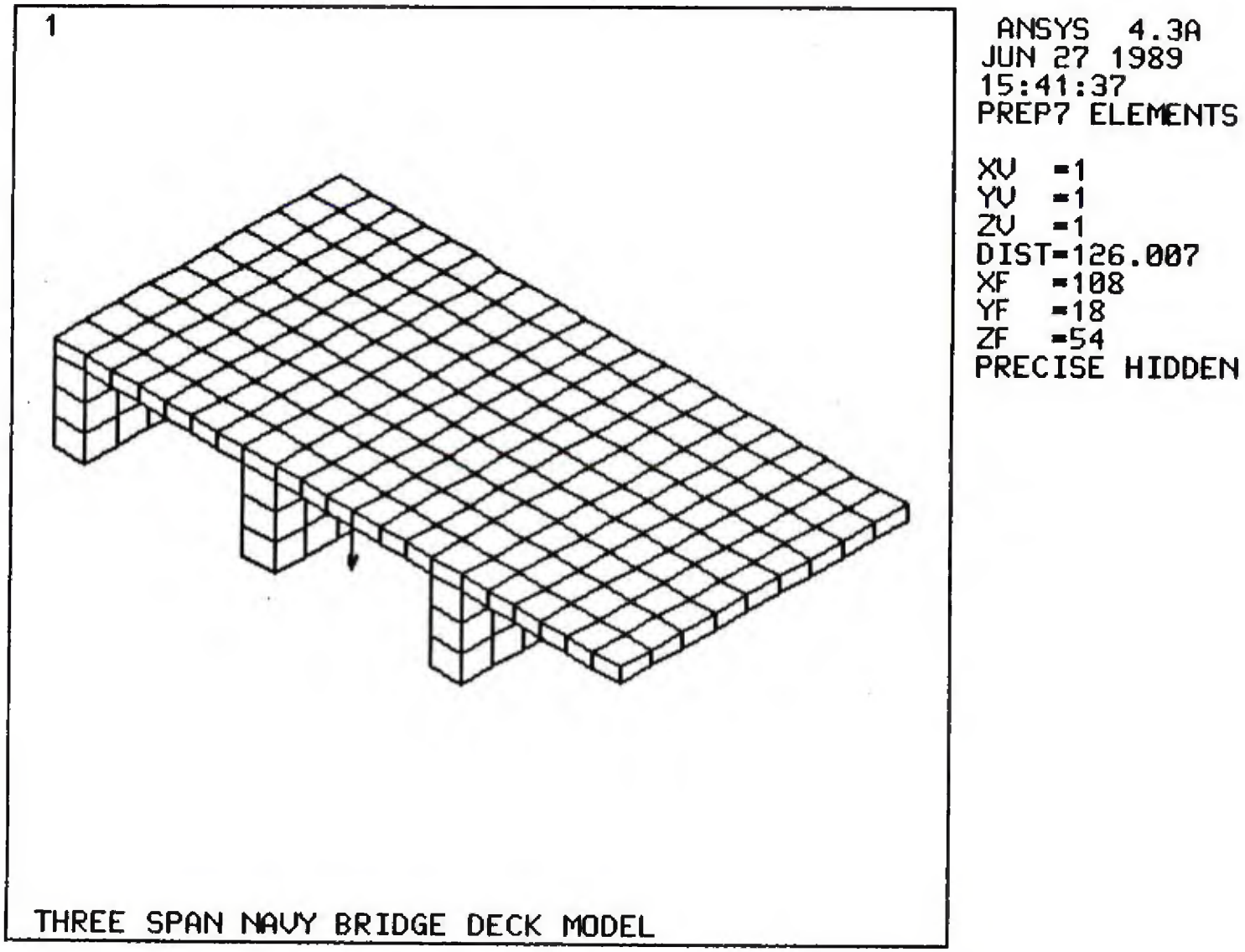

Figure 5.4 - Three Span Finite Element Model with the Continuous Edge Fixed, Geometry Plot

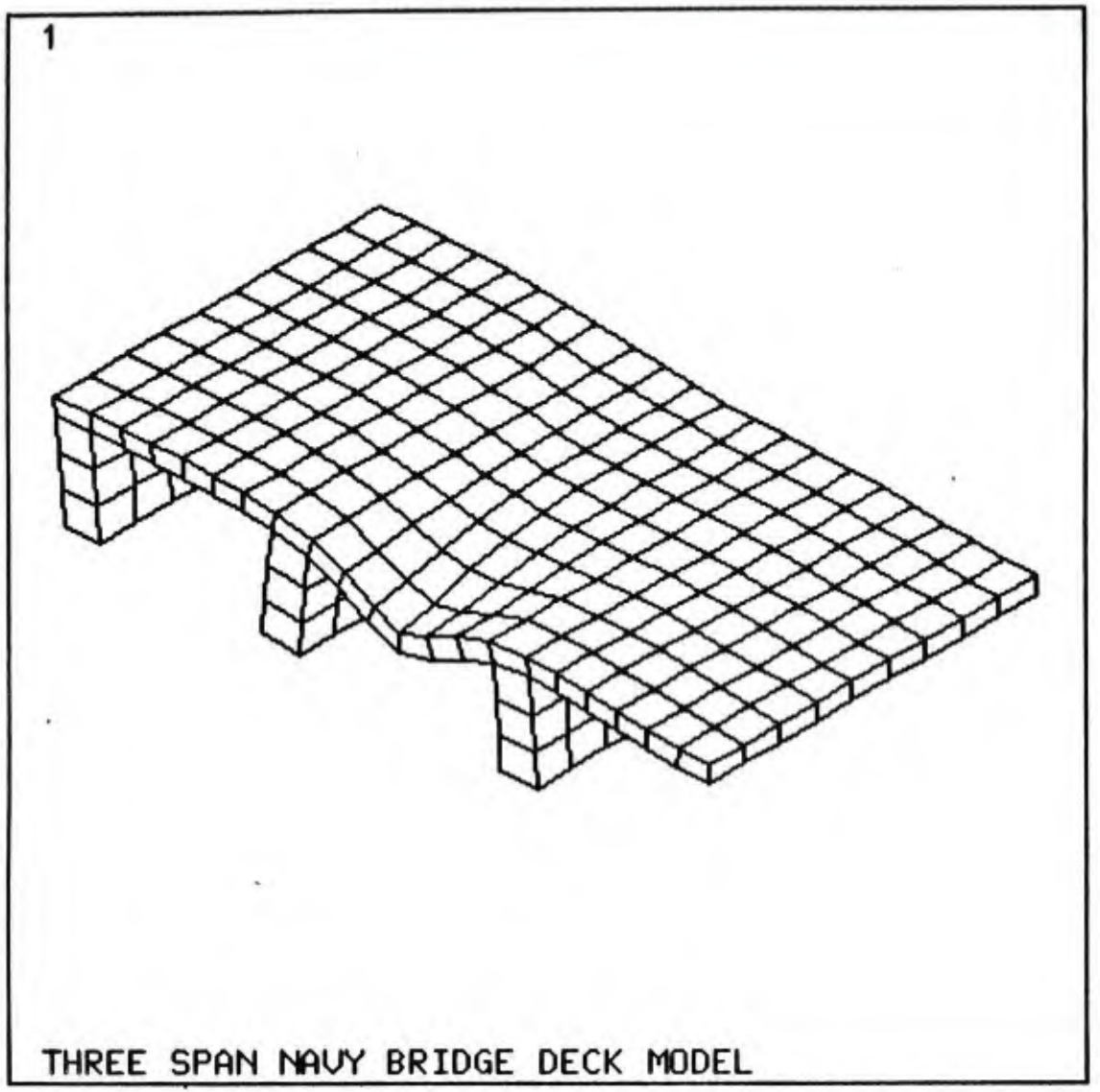

Figure 5.5 - Three Span Finite Element Model with the Continuous Edge Fixed, Deflected Shape

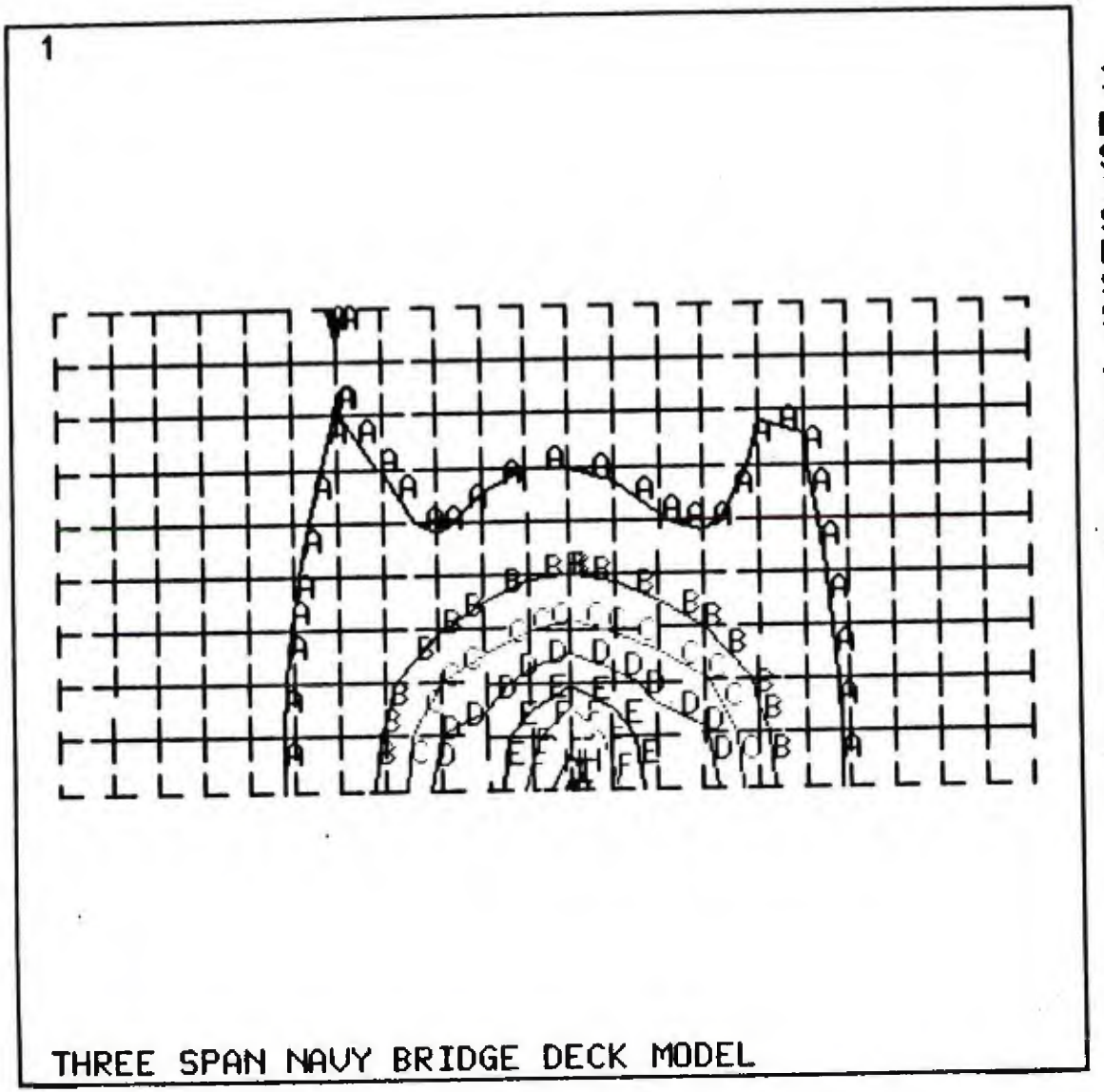

Figure 5.6 - Three Span Finite Element Model with the Continuous Edge Fixed, von Mises Stress Contours on the Top of the Slab

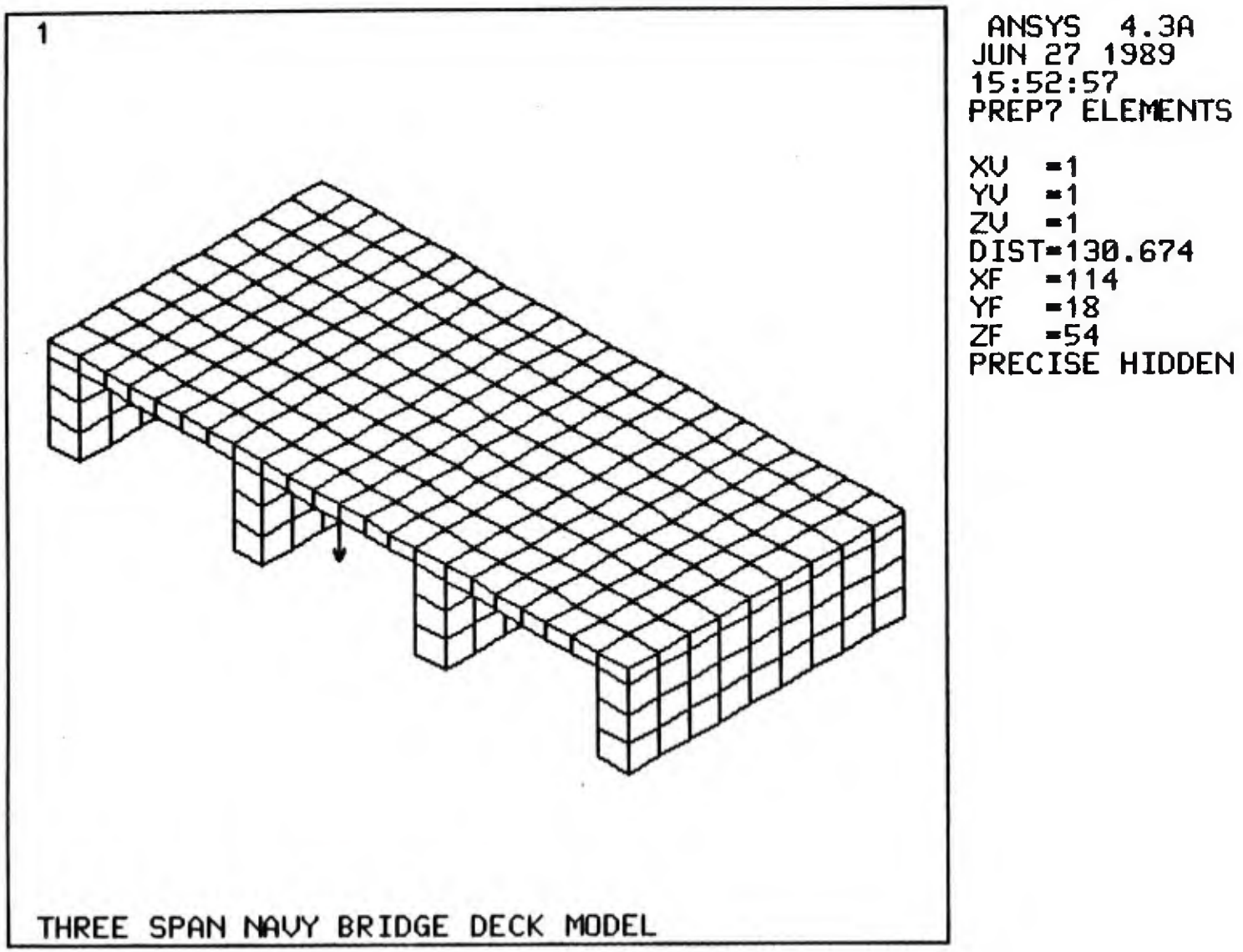

Figure 5.7 - Three Span Finite Element Model with Beam Three, Geometry Plot

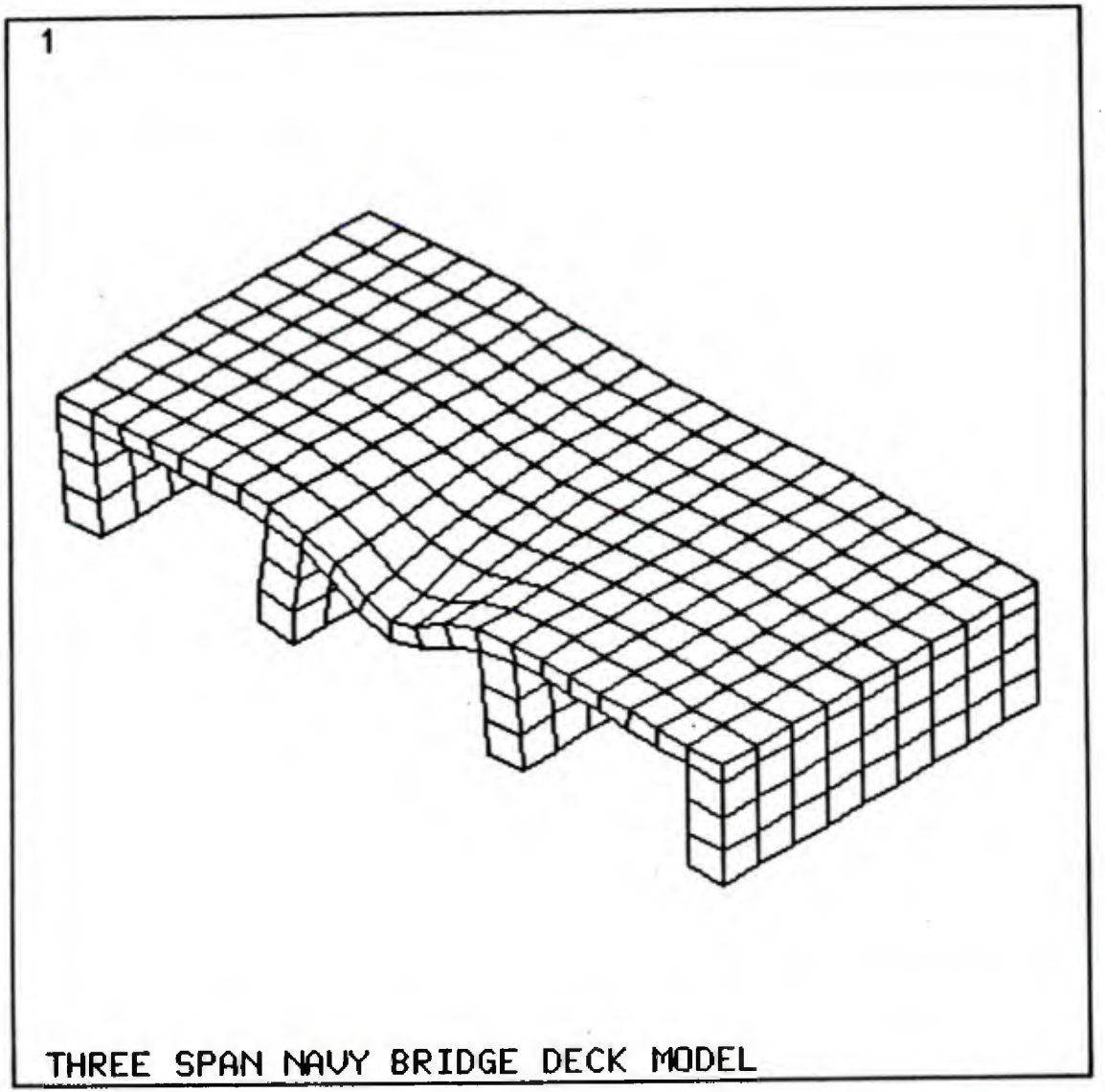

Figure 5.8 - Three Span Finite Element Model with Beam Three, Deflected Shape

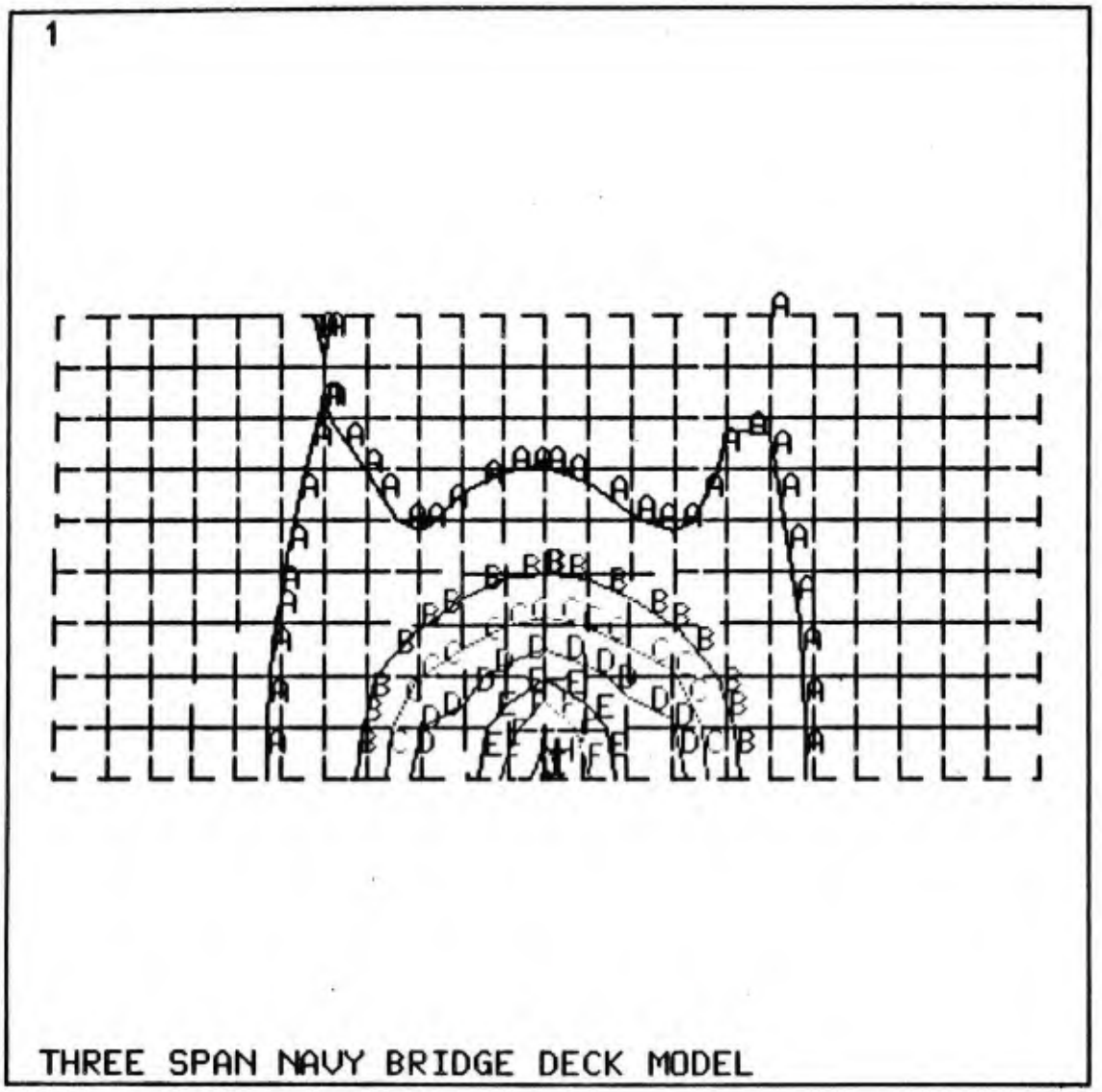

Figure 5.9 - Three Span Finite Element Model with Beam Three, von Mises Stress Contours on the Top of the Slab

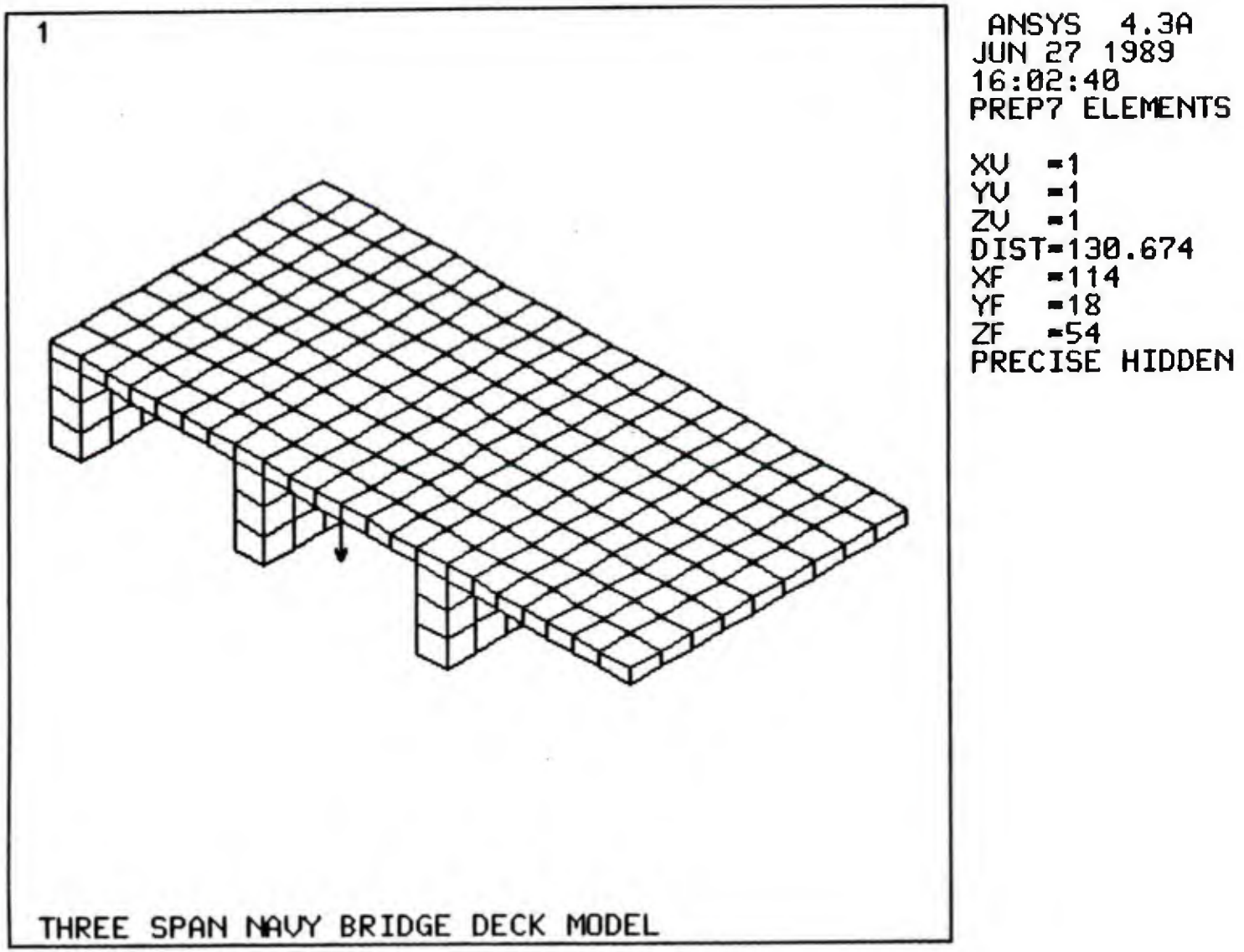

Figure 5.10 - Three Span Finite Element Model with Extra Row of Elements, Geometry Plot

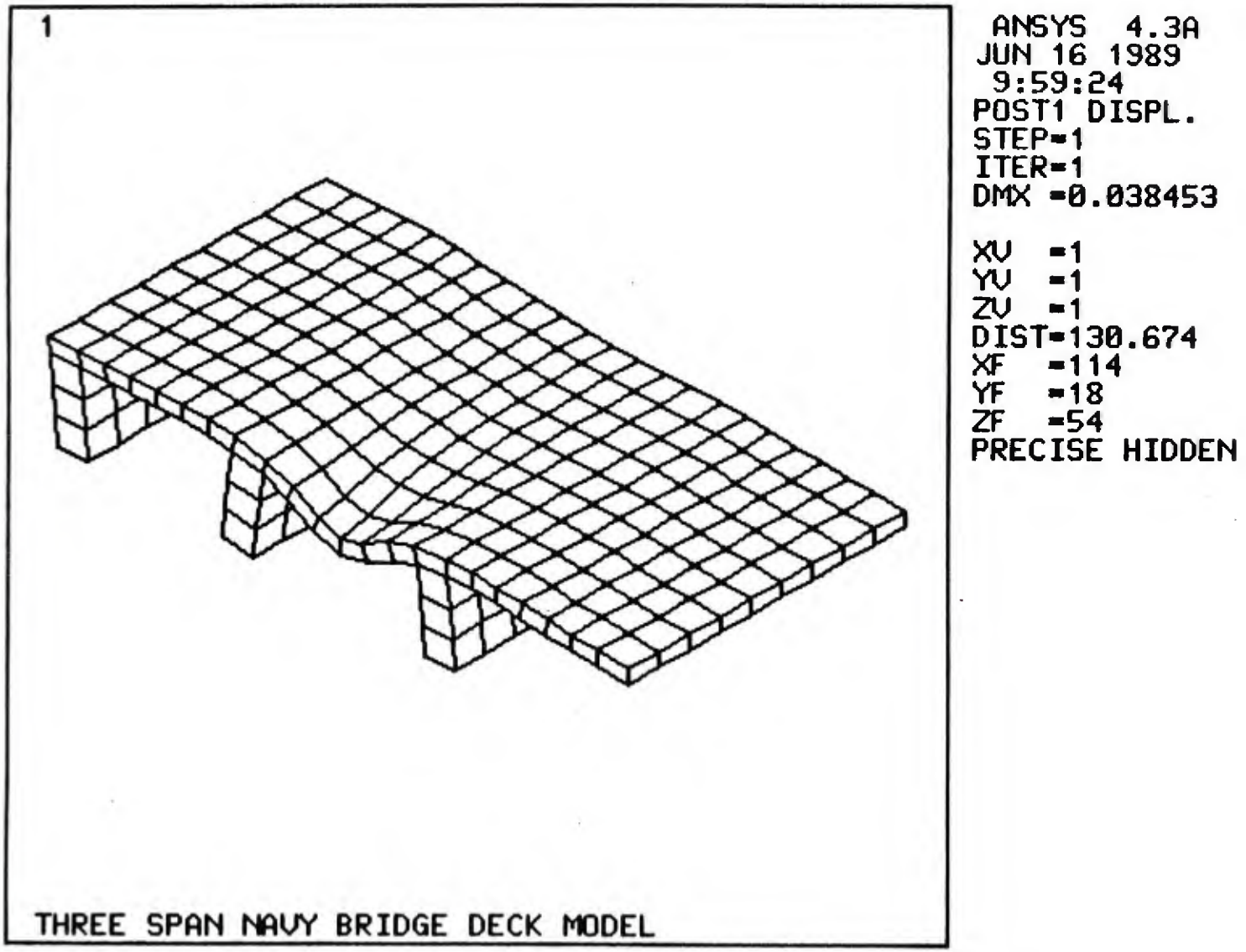

Figure 5.11 - Three Span Finite Element Model with Extra Row of Elements, Deflected Shape

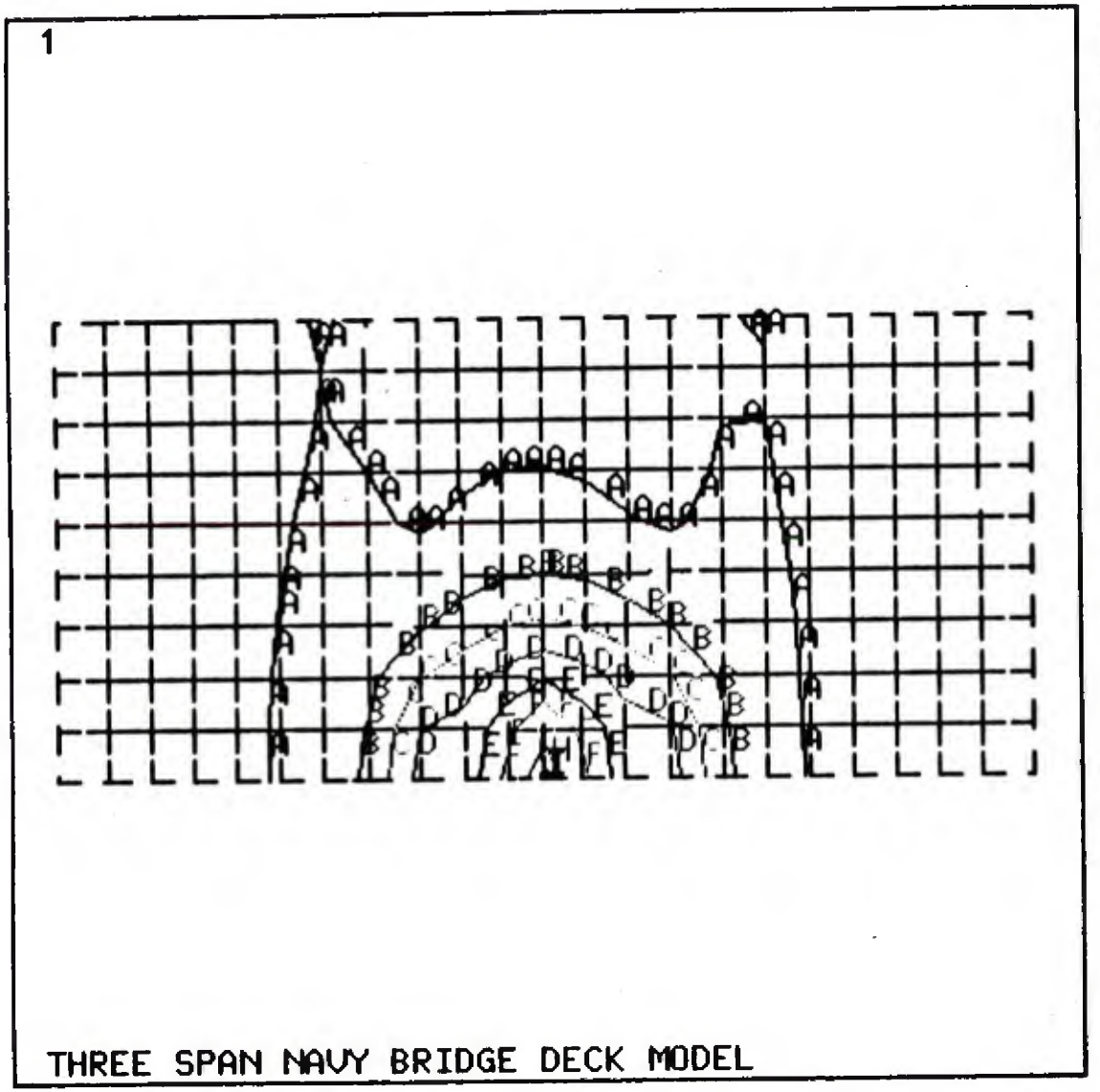

Figure 5.12 - Three Span Finite Element Model with Extra Row of Elements, von Mises Stress Contours on the Top of the Slab

5.2 Finite Element Mesh Selection

The first consideration of the finite element mesh was the boundary conditions of the test span. Clearly two edges are free, but how close to fixed or how close to simply supported are the other two edges? To investigate this edge boundary condition for the test span, two ANSYS finite element models of the test span were created for comparison with the five span model. The first model (Fig. 5.13) consists of a plate of the same size, element type, material, and mesh size used in the five span model with the boundary conditions of two opposite sides free and two opposite sides simply supported. The second model (Fig. 5.14) consists of a plate of the same size, element type, material, and mesh size used in the five span model with the boundary conditions of two opposite sides fixed and two opposite sides free. The results of this analysis are presented in table 5.2.

Table 5.2

Comparison of Boundary Conditions

δ - Deflection under the load

Case	δ (in.)	Figure
Five Span Model	-0.03838	5.1
Simply Supported	-0.066703	5.13
Fixed	-0.027793	5.14

The results of this test seem to indicate that the boundary condition of the test span is somewhere in between the fixed case and the simply supported case. Thus, the finite element mesh selected must accurately simulate the behavior of both a fixed plate and a simply supported plate.

To investigate the size of the mesh required several small examples using the hourglass plate bending elements with different mesh sizes for both the simply supported case and the fixed case were analyzed (Fig. 5.15). A model was generated based on the dimensions of the one half of the fourth span of the Navy Bridge Deck, 108 in x 60 in x 5 3/8 in. Four increments of 27 inches were used in the X-direction and this was held constant for the series of examples. This increment was selected based on the spacing of the instrumentation points on the physical model. In the Y-direction increments of 15 in, 12 in, 10 in, and 7.5 in were used which resulted in 16 element, 20 element, 24 element, and 32 element meshes. These four finite element models were then analyzed with two sets of boundary conditions, one with four sides fixed and another with four sides simply supported. Please note, that for all analyses the transverse shear terms were set to artificially high values to facilitate a direct comparison with classical plate bending theory. Sketches of the finite element meshes are presented in figure 5.15. A five kip load was applied to the center of the plate in all test cases. The finite element results were compared with the analytical results given in Ref. [22]. The results of the series of tests with the hourglass plate

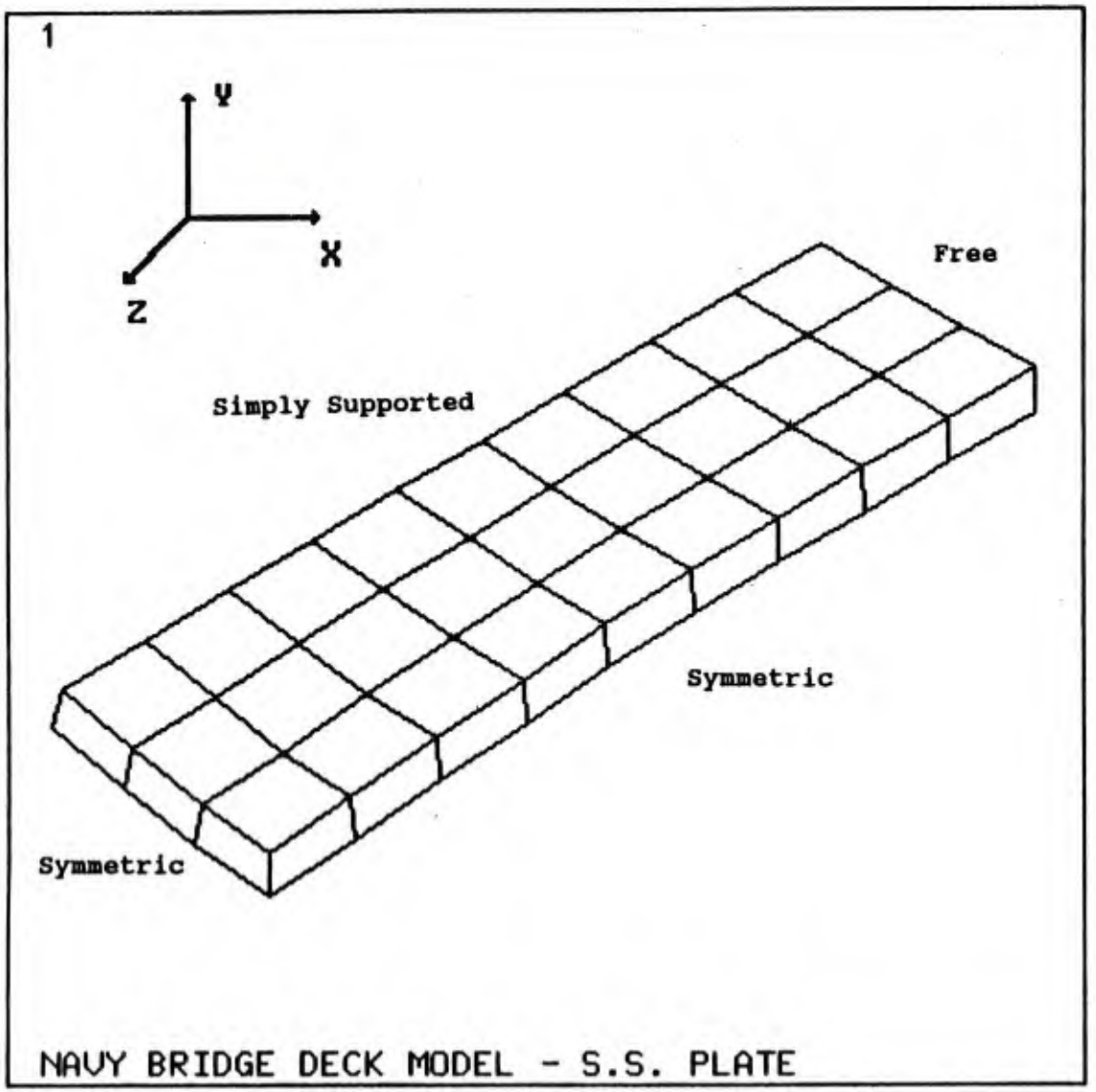

Figure 5.13 - Bi-symmetric Simply Supported Plate

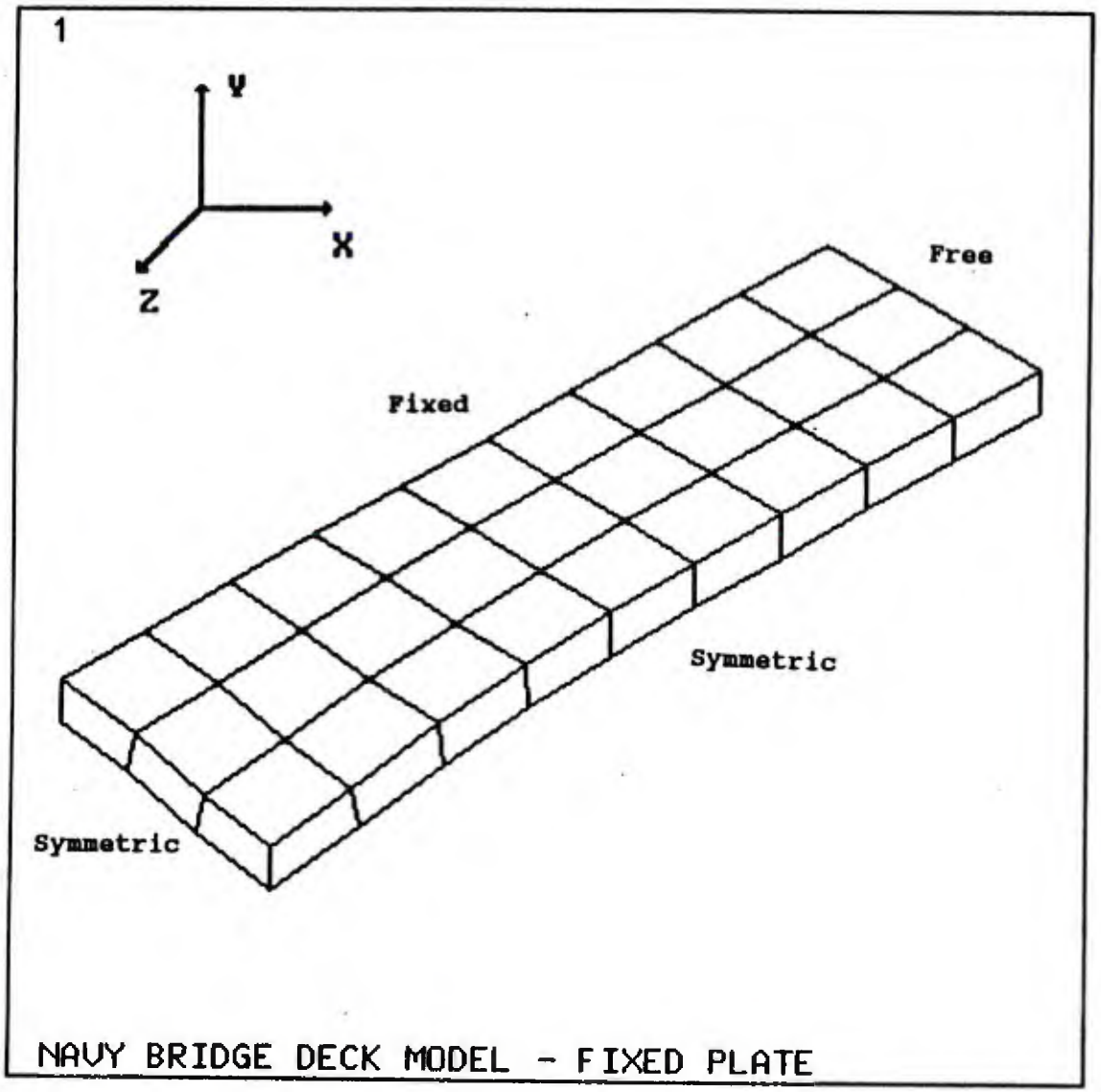

Figure 5.14 - Bi-symmetric Fixed Plate

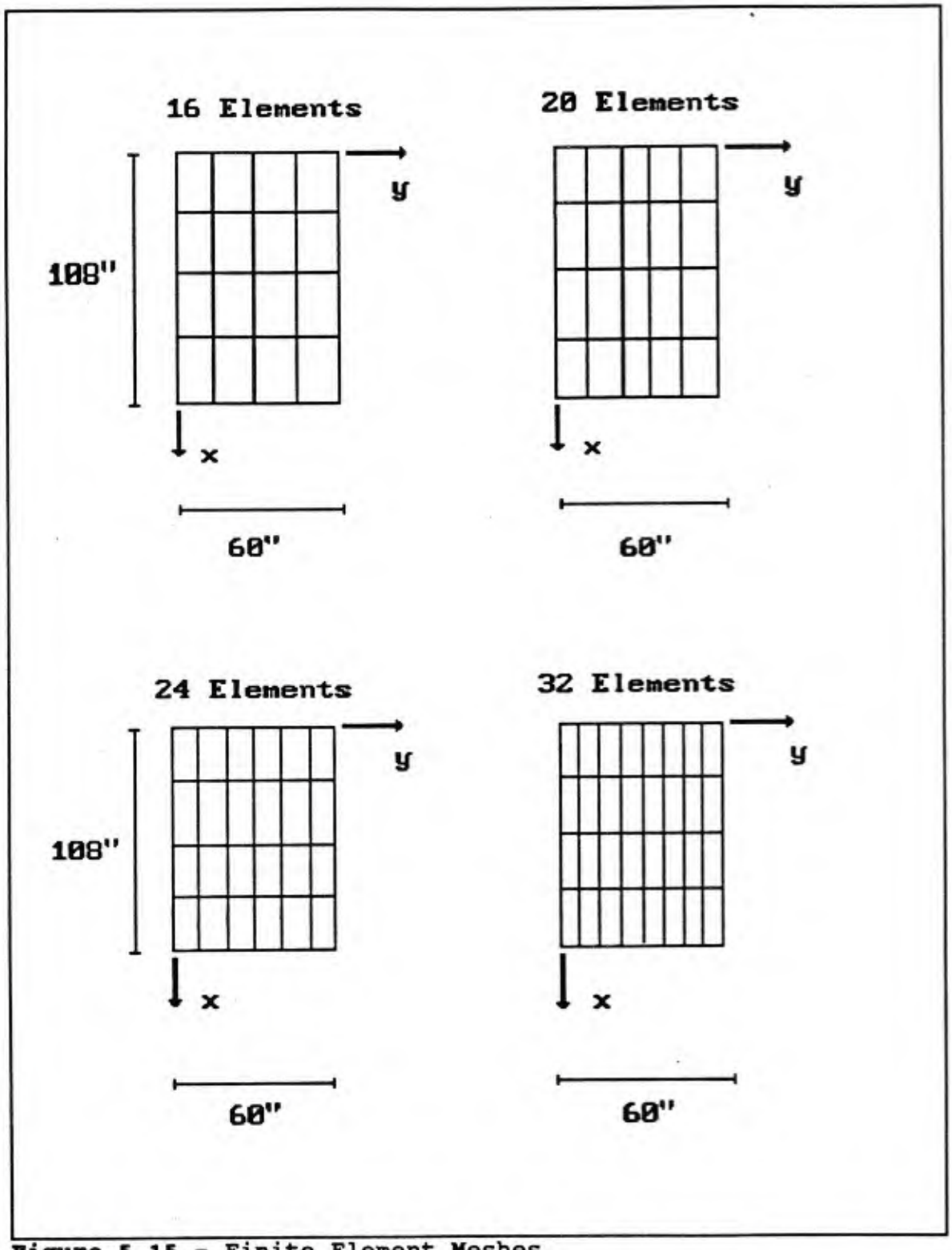

Figure 5.15 - Finite Element Meshes

bending element are presented in table 5.3.

Table 5.3

Comparison of Test Span Finite Element Meshes

δ - Deflection under the load

Error - $100 \times (\delta_{FE} - \delta_{analytical}) / \delta_{analytical}$

Aspect Ratio - Ratio of X - dimension to Y - dimension

Simply Supported Plate:

Case	δ (in.)	Error	Aspect Ratio
Analytical	0.006025	--	--
16 Elements	0.00612	+1.5%	1.8
20 Elements*	0.00574	-4.7%	2.25
24 Elements	0.00617	+2.4%	2.7
32 Elements	0.00619	+2.7%	3.6

Fixed Plate:

Case	δ (in.)	Error	Aspect Ratio
Analytical	0.002677	--	--
16 Elements	0.00212	-21%	1.8
20 Elements*	0.00208	-22%	2.25
24 Elements	0.00238	-11%	2.7
32 Elements	0.00246	-8.2%	3.6

*In the 20 element mesh a center node for the application of the five kip load does not exist. Thus, one half of the load was applied to each of the two nodes next to the center of the plate. The center deflection was found through cubic spline interpolation.

As illustrated in the preceding tables the hourglass plate bending element performs well in all cases even with a large aspect ratio. For the purpose of modelling the Navy Bridge Deck the

aspect ratio will be held to a limit of about two. Which limits the number of elements in the Y-direction to 4 or 5, which as shown in table 5.3 are both reasonable models. Another limitation of the mesh size is that by increasing the number of elements in the test span the number of unknown stiffness parameters also increases. (There is a maximum of 6 unknown stiffness parameters per element.) As well as the fact that by changing the mesh in the test span the instrumented points may no longer be located at nodal points. Since, the instrumentation points are already located and limited to ten stations it is not acceptable to increase the number of elements in the test span.

Thus, the conclusion which was derived was to continue to model the test span with four elements in the Y-direction as previously decided based upon the selection of instrumentation points. It was also decided to model the adjacent spans with four elements in the Y-direction to insure the continuity of the model. Sketches of the selected mesh are included in figures 5.16, 5.17 and 5.18.

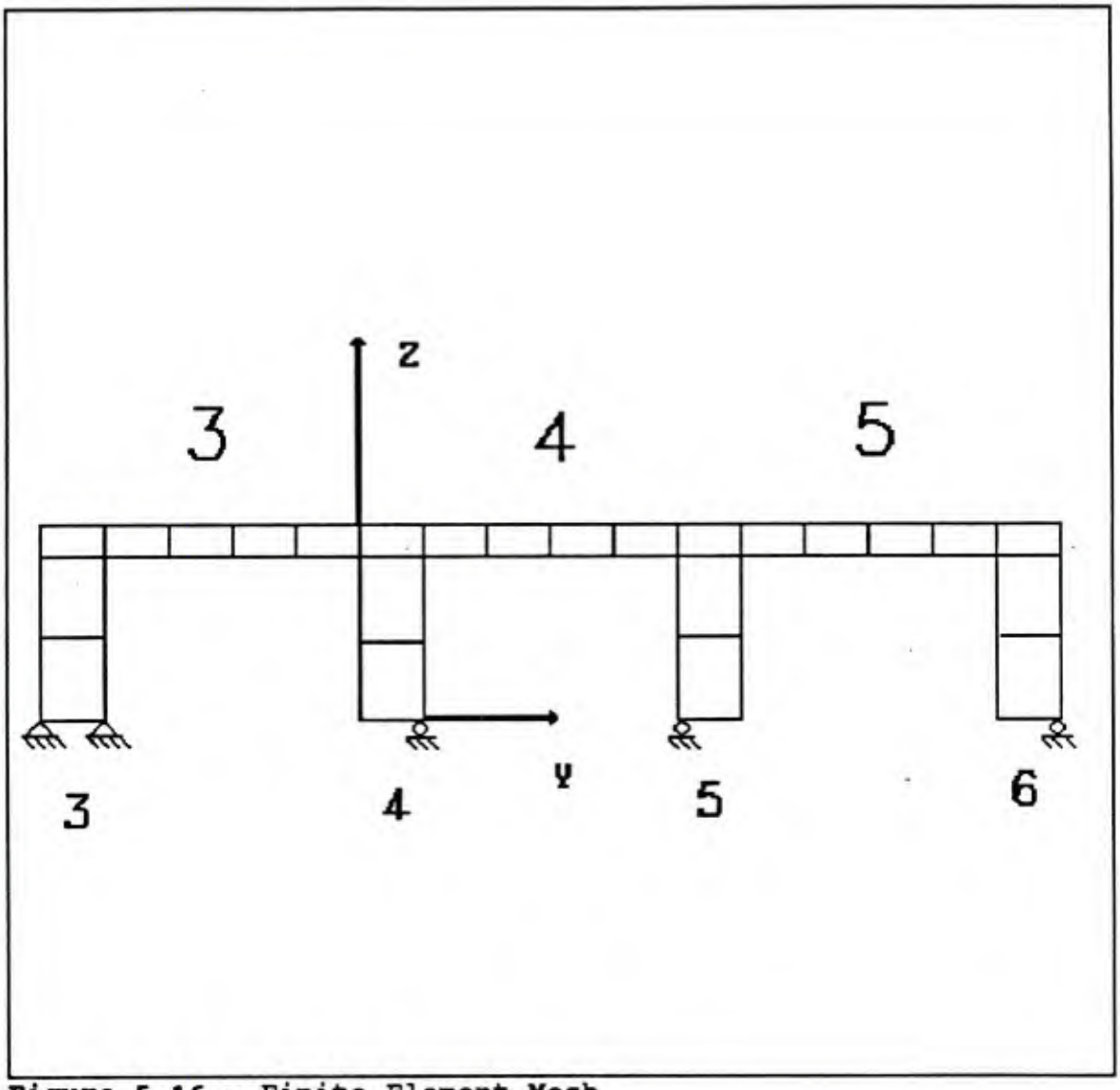

Figure 5.16 - Finite Element Mesh

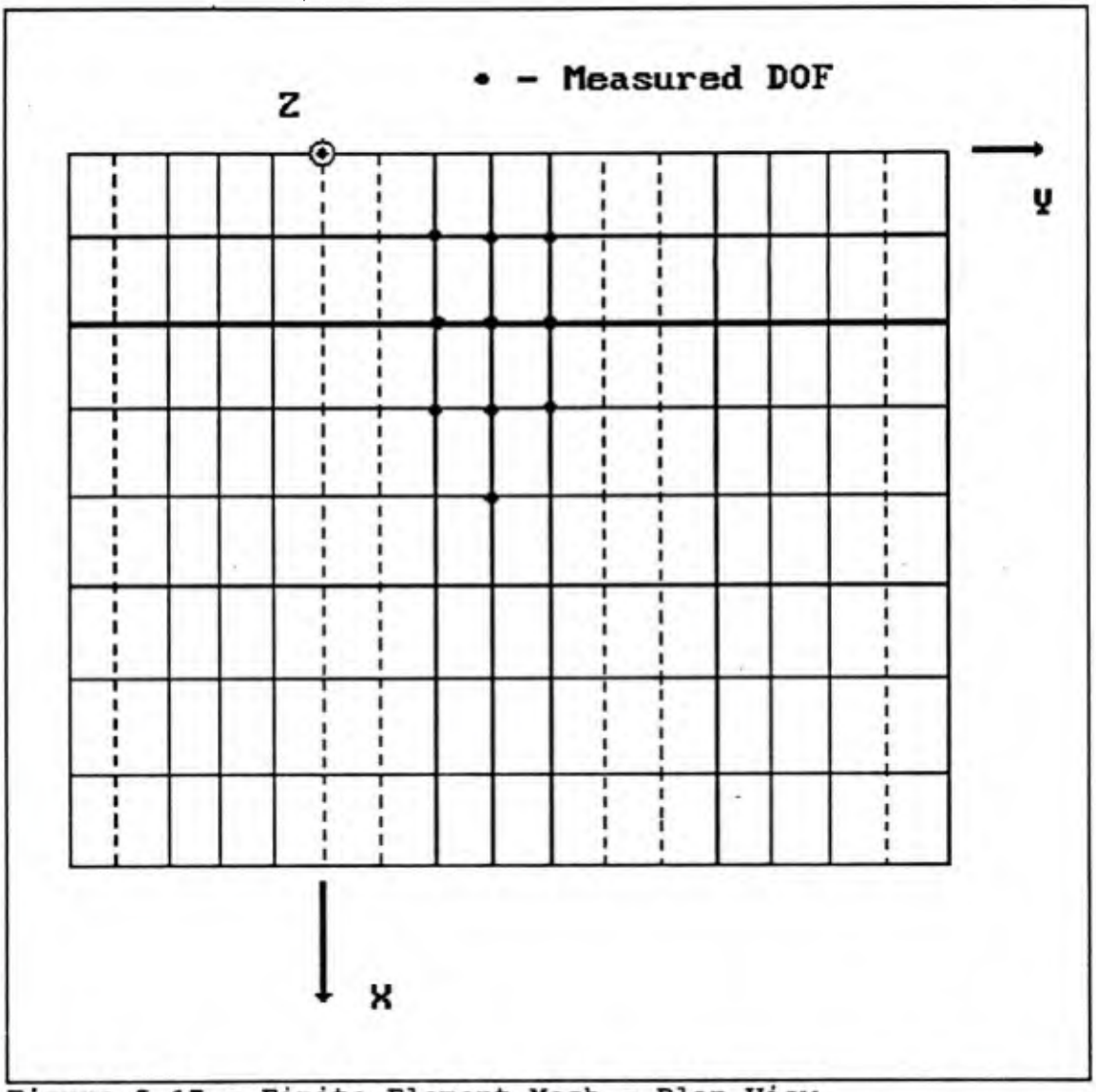

Figure 5.17 - Finite Element Mesh - Plan View

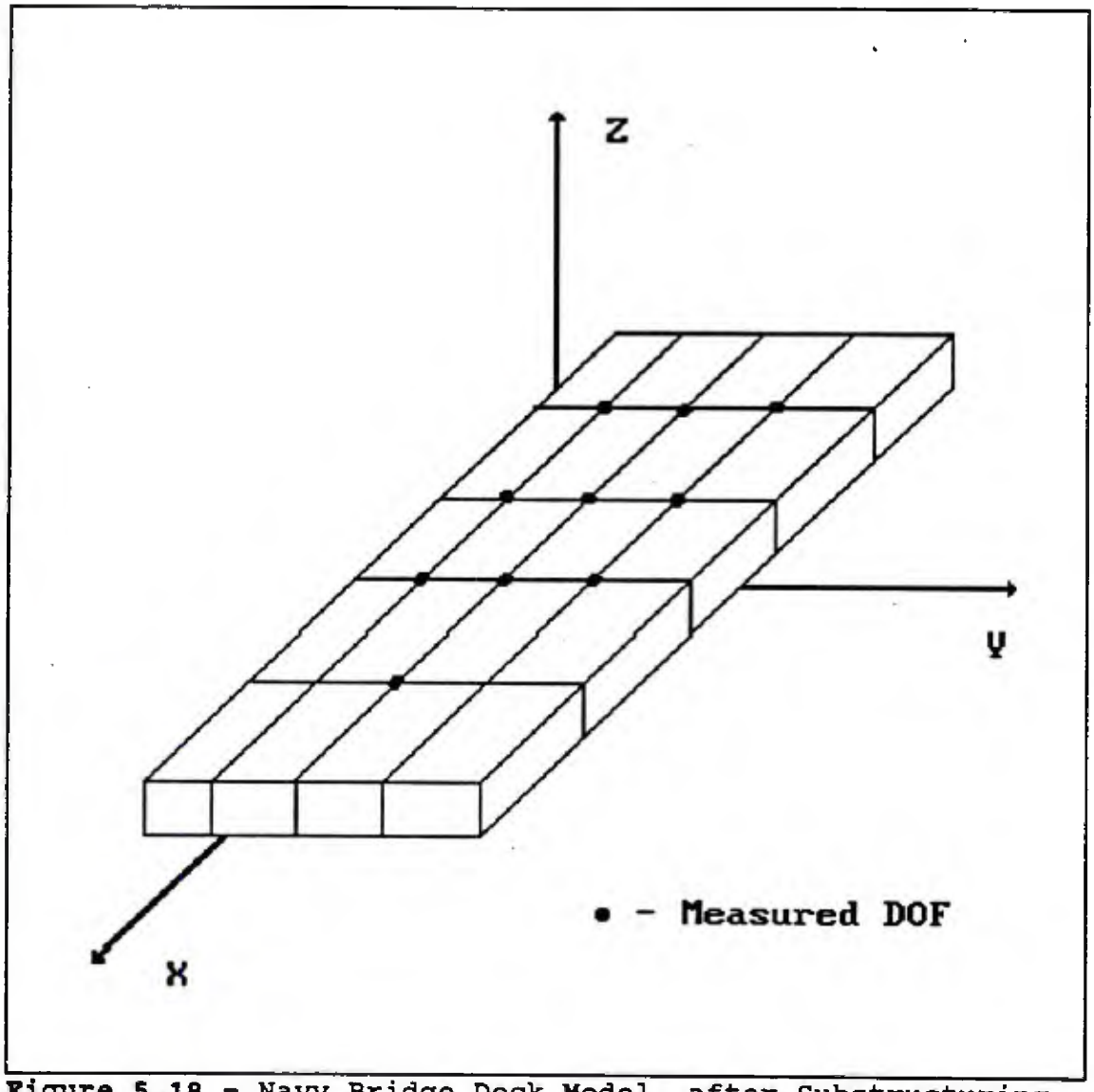

Figure 5.18 - Navy Bridge Deck Model, after Substructuring

5.3 Equivalent Moments of Inertia

The stiffness parameters of the portion of the structure which is condensed out of the finite element model are held constant (known) throughout the identification process. Therefore, it is important to have a good understanding of the properties of the condensed out portion of the structure for successful finite element modelling.

The Navy Bridge Deck was separated into two areas of reinforcement; zones A, and B (Fig. 5.19). The equivalent moments of inertia were calculated for the two zones by considering both the transformed cracked section, and the uncracked section of the doubly reinforced slab in orthogonal directions. The equivalent moments of inertia for both positive and negative bending moments of each zone were computed (Ref. [22]). The results of the calculations are presented in table 5.4.

Table 5.4

Equivalent Moments of Inertia

Transformed Cracked Section:

Zone A

Moment	I_{eq} (in.^{4}/ft.) Y-dir.	I_{eq} (in.^{4}/ft.) X-dir.
Positive	52.04	24.74
Negative	18.81	12.88

Zone B

Moment	I_{eq} (in.^{4}/ft.) Y-dir.	I_{eq} (in.^{4}/ft.) X-dir.
Positive	52.44	13.01
Negative	33.02	12.07

Uncracked Section:

Zone A

Moment	I_{eq} (in.^{4}/ft.) Y-dir.	I_{eq} (in.^{4}/ft.) X-dir.
Positive	168.77	159.88
Negative	168.77	159.88

Zone B

Moment	I_{eq} (in.^{4}/ft.) Y-dir.	I_{eq} (in.^{4}/ft.) X-dir.
Positive	172.44	158.33
Negative	172.44	158.33

If the bridge deck model is cracked at the time of parameter identification testing the equivalent moments of inertia for the transformed cracked section must be used in the appropriate locations of positive and negative moments. The equivalent moments of inertia used for parameter identification analysis are the uncracked equivalent moments of inertia.

Once the values of the uncracked equivalent moments of inertia are found the plate bending stiffnesses for an orthotropic reinforced concrete plate can be computed from equation 4.19 in chapter four of this report.

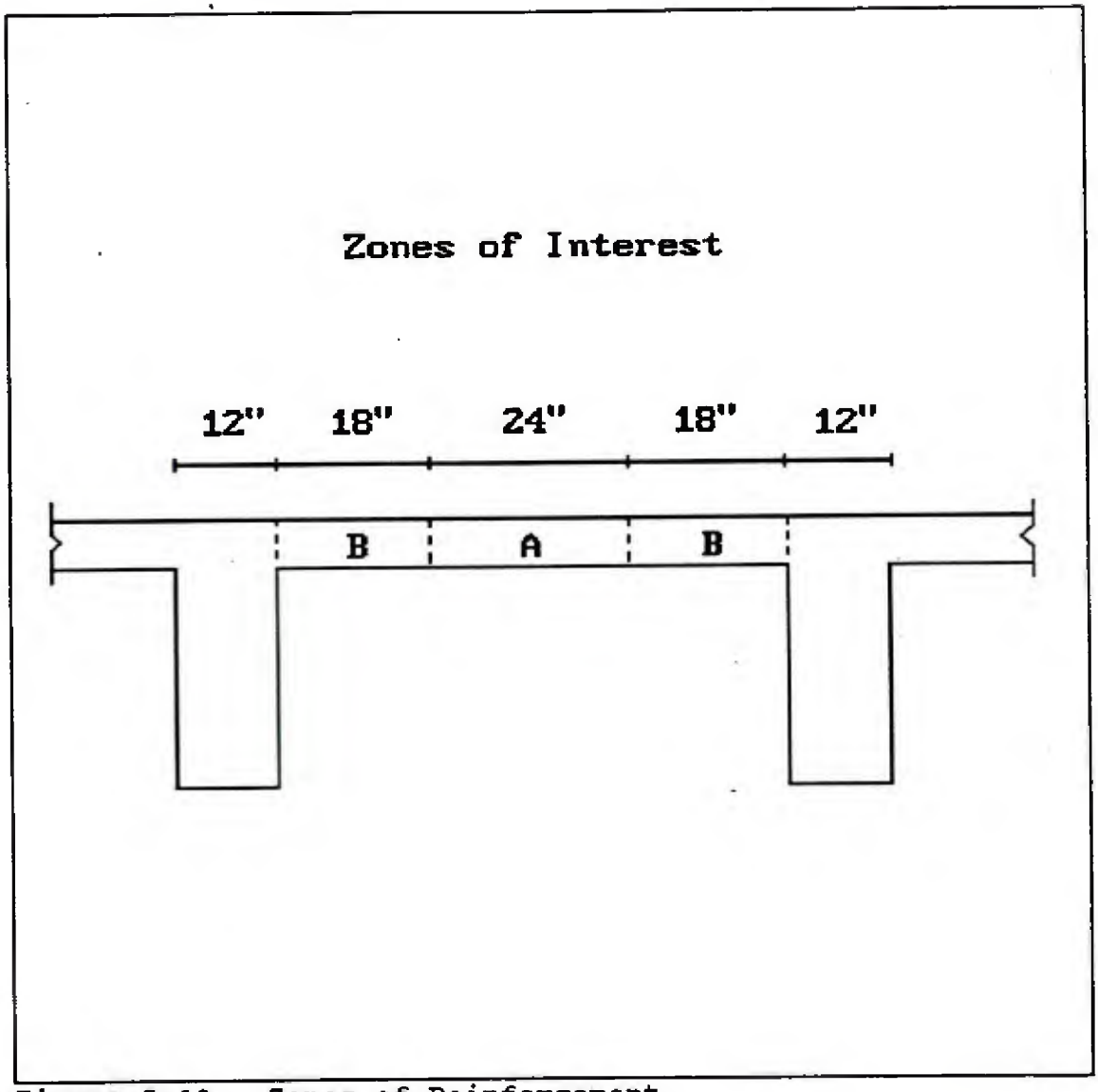

Figure 5.19 - Zones of Reinforcement

5.4 Substructuring

In order to reduce the number of active degrees of freedom in the parameter identification program (PARIS), substructing (Ref. [18], [24] and [25]) is performed. A super-element is generated by condensing out all the internal degrees of freedom of a substructure of the three span finite element model. The substructure consists of the three span model without the instrumented test span (Fig 5.20). The global stiffness matrix for the substructure is generated, and partitioned into active degrees of freedom, and internal degrees of freedom.

$$\left\{ \begin{array}{c} F_a \\ \hline F_i \end{array} \right\} = \left[\begin{array}{c|c} k_{aa} & k_{ai} \\ \hline k_{ia} & k_{ii} \end{array} \right] \left\{ \begin{array}{c} u_a \\ \hline u_i \end{array} \right\} \tag{5.1}$$

Since, there are no applied forces at the internal degrees of freedom ($F_i = 0$), the above matrix equation can be solved for the active degrees of freedom.

$$\{F_a\} = ([k_{aa}] - [k_{ai}][k_{ii}]^{-1}[k_{ia}])\{u_a\} \tag{5.2}$$

The condensed substructure (super-element) stiffness matrix is equivalent to

$$[k_{sub}] = [k_{aa}] - [k_{ai}][k_{ii}]^{-1}[k_{ia}] \tag{5.3}$$

Thus, in the parameter identification program only the stiffness matrix for the instrumented test span is generated (Fig 5.18). To complete the finite element model, the substructure stiffnesses (Eq. 5.3) are added to the correct degrees of freedom of the instrumented test span stiffness matrix.

The substructure (Fig. 5.20) contains 1189 degrees of freedom after applying the boundary conditions. The number of active degrees of freedom is 90, thus the number of internal degrees of freedom to be condensed out in 1089. To perform static condensation on the substructure's internal degrees of freedom, requires the inversion of $[k_{ii}]$ (Eq. 5.3), which is a 1089 x 1089 matrix . This inversion can be accomplished within the core memory of the VAX 8550 at Tufts University.

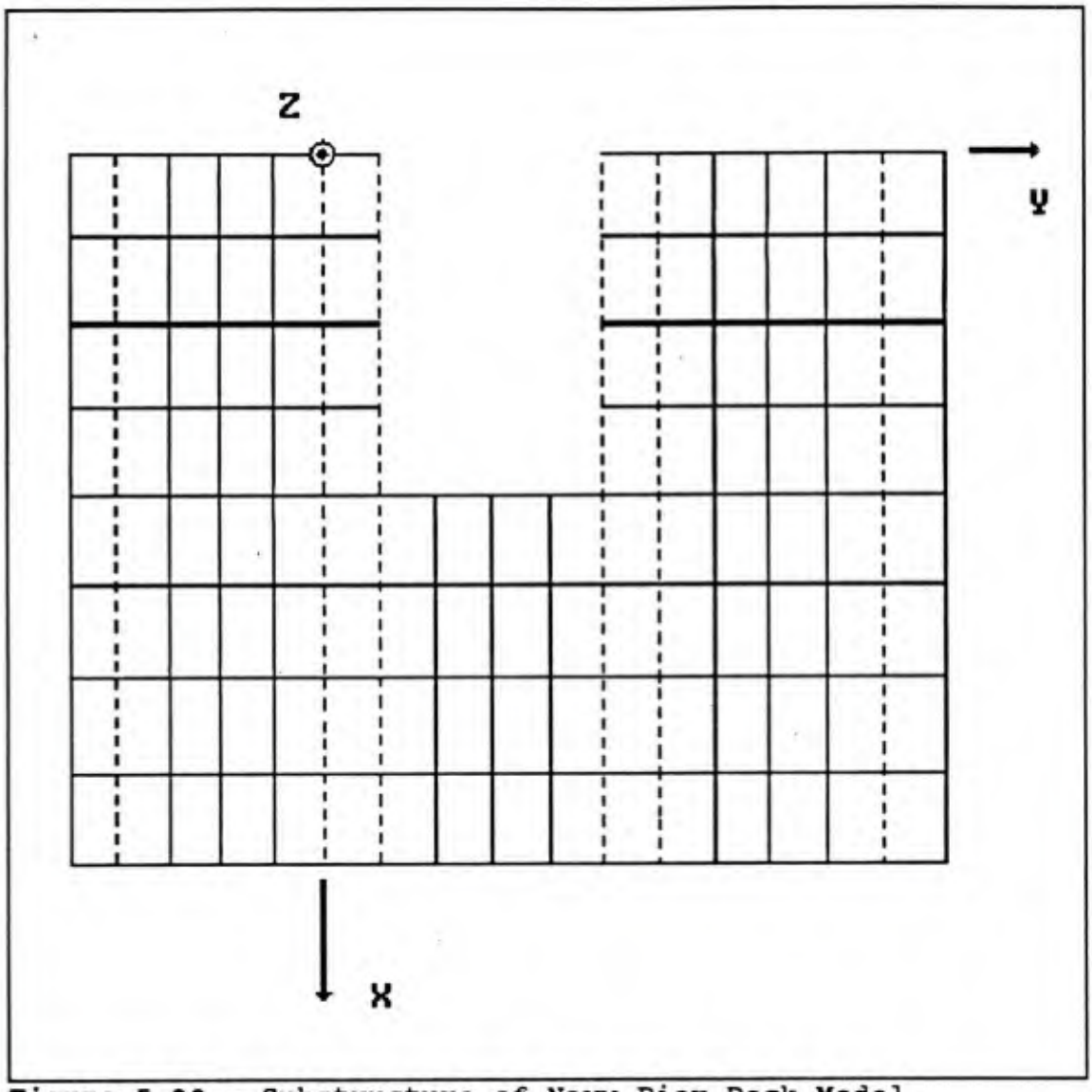

Figure 5.20 - Substructure of Navy Pier Deck Model

6.0 PARAMETER IDENTIFICATION WITH SIMULATED DATA

In this chapter, parameter identification is performed on the instrumented test span of the Navy Pier deck. The goal is to relate the simulated behavior of the Navy Pier Deck to the mathematical relations derived in chapter two. The initial values of the unknown parameters are derived from the entries of the constitutive matrix for plate bending for an uncracked slab, and the transverse shear terms of the [C] matrix as defined in chapter five. The true values of the unknown parameters used to simulate damage in the slab were taken to be approximately fifty percent of the initial values. In all cases the simulated damage was limited to local stiffness deterioration of one or two elements, this was done to simulate a degradation of bending stiffness or transverse shear stiffness due to localized cracks.

In the first section of this chapter, parameter identification is performed without measurement error, and in the second section uncertainties are incorporated in the force and displacement measurements. In both sections test data is simulated for the damaged slab.

6.1 Parameter Identification without Measurement Error

In this section the goal is to perform parameter identification on different subsets of elements to identify different types of localized damage in the test span. The purpose

of investigating different subsets of parameters is to identify the entire test span with a series of parameter identification runs. There are four bending stiffness and/or two transverse shear stiffnesses per element. Ten measurement stations provide 55 independent measurements, thus a maximum of 55 unknown parameters can be identified. Thus, not enough data is available to identify all the unknown parameters in the test span with one parameter identification run. Figure 6.1 illustrates the element numbering and the instrumentation points of the test span. Table 6.1 lists each of the parameter identification examples and the results.

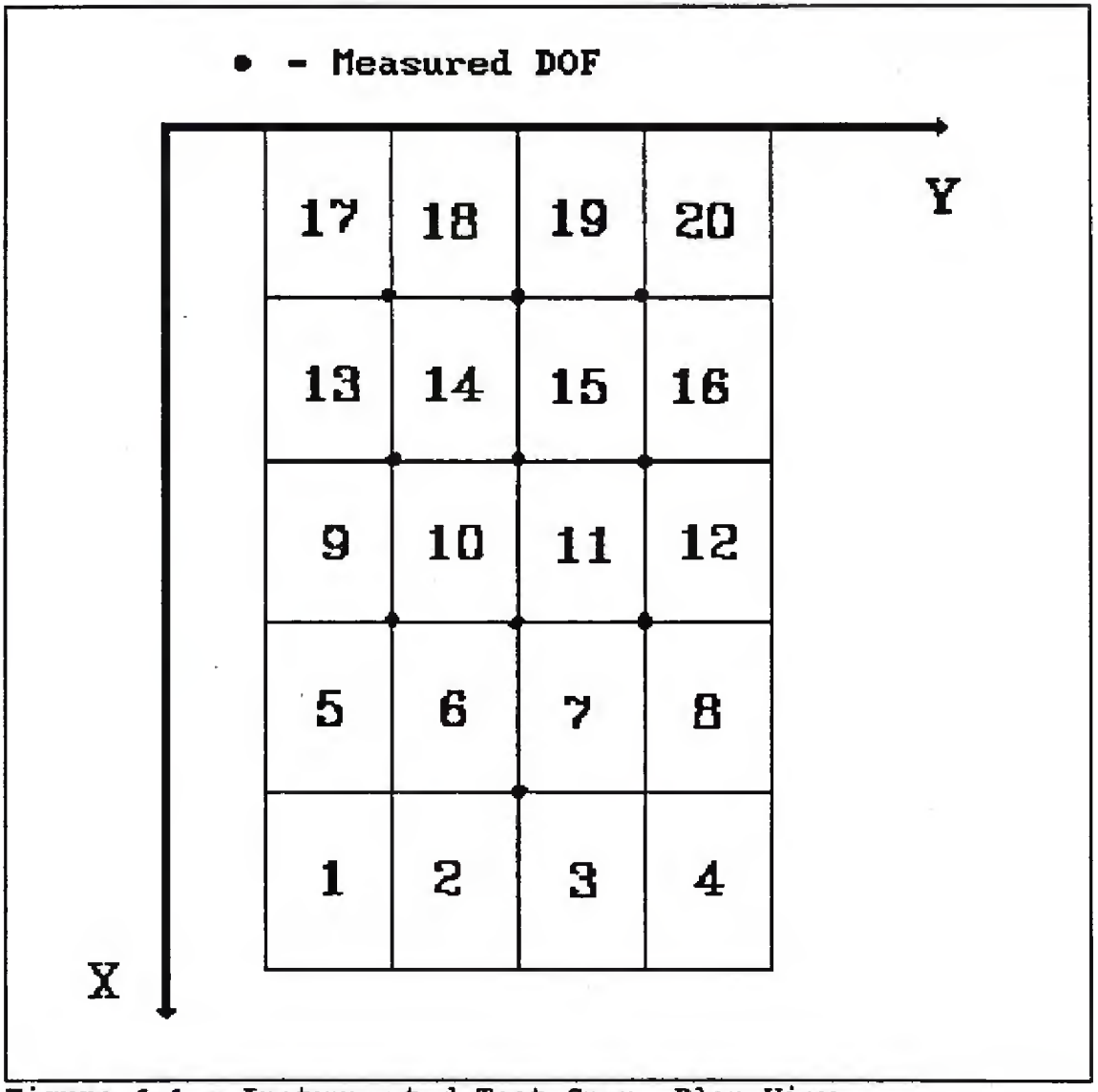

Figure 6.1 - Instrumented Test Span, Plan View

Table 6.1

Summary of Parameter Identification Runs

NUE - Number of Unknown Parameters per Element
NUP - Number of Unknown Parameters
ITER - Number of Iterations for Convergence

Example	Elements	NUE	NUP	ITER	Result
PARIS1	6,7,9-20	4*	55	10	Diverged
PARIS2	9-20	4	48	10	Converged
PARIS3	1-12	4	48	7	Converged
PARIS4	5-16	4	48	10	Converged
PARIS5	2,3,6,7,10,11,14,15,17-20	4	48	10	Converged
PARIS6	6,7,9-12,13-16	4	40	6	Converged
PARIS7	10,11,13-20	4	40	6	Converged
PARIS8	2,3,6,7,10,11,14,15,18,19	4	40	7	Converged
PARIS9	1-4,6,7,10,11,14,15,18,19	4	48	10	May Converge
PARIS10	17-20	6	24	8	Converged
PARIS11	1,4,5,8,9,12,13,16	6	48	6	Converged
PARIS12	5,8,9,12,13,16,17,20	6	48	6	Converged

*Element 6 of PARIS1 only contains 3 unknown parameters, as 55 is not divisible by 4.

The above examples can be categorized by the types of identified damage, PARIS1 through PARIS9 are attempting to identify only the elements of the [D] matrix. In these examples, there are four unknown bending stiffnesses per element. The purpose of these examples is to identify bending stiffness degradation in the center of the plate because of localized cracking. Whereas, PARIS10 through PARIS12 are attempting to identify both elements of the [D] matrix and [C] matrix. In these examples, there are six unknowns per element, four bending stiffnesses and two transverse shear. The purpose of these examples is to identify degradation of bending stiffness and transverse shear stiffness.

The first example PARIS1 contains fifty-five unknown parameters with damage to elements ten and fifteen. PARIS1 contains the maximum number of unknown parameters that can be derived from the test measurements. There are no apparent linear dependencies in this problem, however this case did not converge because of an ill-conditioned sensitivity matrix. The smallest eigen value for this example is 4.6×10^{-15}, which is very close to zero. The condition number of the sensitivity matrix for this example is 1.7×10^{11}. The ill-conditioning of the sensitivity matrix causes the gradient search to diverge, as error function is not minimized for this problem. The error function increases by an order of magnitude within the first three iterations.

To reduce the ill-conditioning of the sensitivity matrix the number of unknown parameters is reduced to forty-eight. This number was selected because it reduces the number of unknown parameters, and it allows the number of unknown elements to be a whole number. It was concluded that there is no physical basis for allowing some of the parameters of an element to be constant and other parameters variable. Thus, elements of the remaining examples are either completely variable or constant.

PARIS2 contains forty-eight unknowns and damage to elements ten and fifteen. For this example the smallest eigen value is 4.8×10^{-12}, this number is close to zero, but larger than the smallest eigen value for PARIS1. The condition number for PARIS2 is 1.6×10^{8}. The sensitivity matrix for PARIS2 is not as ill-conditioned as for PARIS1, as a result the error function for this example is

minimized within ten iterations. PARIS2 illustrates that by reducing the set of unknown parameters, the ill conditioning of the sensitivity matrix can be alleviated. Based on the performance of PARIS2 it is concluded that forty-eight unknown parameter is an acceptable number of parameters to identify for the remaining parameter identification examples.

PARIS3 contains forty-eight unknown parameters and damage to elements two and seven. For this example, the condition number is 6.4×10^{7}, and the smallest eigen value is 7.7×10^{-12}. This example converges very quickly as the sensitivity matrix is not ill-conditioned. In this example the unknown parameters are more sensitive to the test measurements than in the previous two examples. Small changes in the stiffnesses of this subset of unknown parameters produce larger changes in the test measurements, this facilitates a faster convergence of the algorithm.

PARIS4 contains forty-eight unknown parameters with damage to the bending stiffness of elements ten and fifteen. For this example, the condition number is 5.6×10^{7}, and the smallest eigen value is 1.2×10^{-11}. This example also converges quickly as the sensitivity matrix is not ill-conditioned.

PARIS5 contains forty-eight unknown parameters with damage to the bending stiffness of elements ten and fifteen. For this example, the condition number is 8.7×10^{6} and the smallest eigen value is 6.4×10^{-11}. This example also converges quickly as the sensitivity matrix is not ill-conditioned.

Both PARIS6 and PARIS7 are examples that contain subsets of

parameters of PARIS4 AND PARIS2, respectively. The purpose of these two examples is to illustrate how the convergence properties of the parameter identification algorithm improve with fewer unknown parameters. Both PARIS6 and PARIS7 converge in only six iterations while locating the same damage as PARIS4 and PARIS2. Both examples have well conditioned sensitivity matrices, PARIS6 has a condition number of 9.2×10^{5} and PARIS7 has a condition number of 8.1×10^{4}. In both of these cases, there is more independent information available to speed the convergence of the parameter identification algorithm.

PARIS8 contains forty unknown parameters and damage to elements ten and fifteen. The smallest eigen value is 1.0×10^{-9} and the condition number is 5.3×10^{6}. The subset of unknown parameter in PARIS8 is sensitive to the test measurements, thus the algorithm converges very quickly. This example also illustrates how reducing the number of unknowns increases the speed of convergence of the algorithm.

PARIS9 contains forty-eight unknown parameters and damage to the bending stiffnesses of elements ten and fifteen. The smallest eigen value is 4.9×10^{-12}, and the condition number is 1.1×10^{8}. The solution to this example is oscillatory, it does not converge within ten iterations. However, this example may converge if allowed to continue for more iterations. PARIS9 illustrates that this subset of parameters is not as sensitive to the test measurements as the other examples. The elements included in PARIS9 are farther away from the measurement stations, thus changes

in the stiffnesses of these elements do not produce large changes in test measurements. Thus, resulting in slow convergence of the parameter identification algorithm.

The purpose of PARIS10 is to investigate the ability of the algorithm to perform parameter identification on a localized area of the instrumented test span. It was suspected that the edge elements of the test span may create a problem for the program, as the edge elements are not highly stressed by the test loads. The number of unknown parameters for this run is twenty-four with localized bending and transverse shear damage to element nineteen. The smallest eigen value for this example is 3.6×10^{-9}, and the condition number is 9.6×10^{6}. The solution for PARIS10 is very well behaved, the algorithm converges in 8 iterations.

PARIS11 contains forty-eight unknown parameters with damage to the bending stiffness and transverse shear terms of element twelve. The smallest eigen value for this example is 9.1×10^{-12} and the condition number is 2.4×10^{10}. The subset of unknown parameters for PARIS11 is very sensitive to the test measurements, thus the algorithm quickly detects the damage to element twelve and converges in six iterations.

PARIS12 contains forty-eight unknown parameters and damage to the bending stiffness and transverse shear terms of element twelve. In this case, the smallest eigen value is 1.3×10^{-11} and the condition number is 1.6×10^{10}. Similar to PARIS11 the subset of unknown parameters is very sensitive to the test measurements, thus this example converges quickly.

The results of these twelve parameter identification runs illustrate the capability of the PARIS program to locate damaged areas of the test span successfully. Forty eight parameters were identified for a variety of test cases. The program is capable of detecting localized bending and shear stiffness degradation. In the next section the effect of measurement errors will be investigated.

6.2 Level of Acceptable Measurement Errors

Past experience [Ref. 17] has shown that an acceptable level of error is dependent on the structural model, level of indeterminacy, the set of force and displacement measurements, and the subset of parameters identified. Therefore, the level of acceptable measurement error is a function of the topology of the structure. Thus, the level of acceptable measurement errors is different for various problems. In this section the goal is to determine the level of acceptable error for the Navy Pier Deck problem. There are three possible outcomes of the input/output error relationship. The first is the linear range, small errors in the measurements correspond to small errors in the identified parameters. The second in the nonlinear range, small errors in the measurements correspond to large errors in the identified parameters. The third is the divergence range, small errors in the measurements result in algorithmic failure. Thus, the objective is to find the level of error for the Navy Pier Deck that induces linear errors in the identified parameters. For this purpose three different error levels were investigated, 0.002%, 0.02%, and 0.2% measurement errors.

PARIS7 was selected as a test case for the Monte Carlo experiments. This example was selected as a fair representation of the behavior of the parameter identification algorithm. Three Monte Carlo experiments were performed for each of the specified levels of error. The reason so few Monte Carlo observations were

selected, was the considerable amount of computer time necessary for one parameter identification run. The parameter identification algorithm takes approximately three hours of CPU time on the VAX 8550 at Tufts University to go ten iterations with 40 unknown parameters. The number of Monte Carlo observations selected will result in a poor estimate of the mean. If the number of Monte Carlo observations is increased the estimate of the mean will move closer to the true value of the mean. The results of this error investigation are presented in table 6.2.

Table 6.2

Summary of Monte Carlo Experiments

EL - Damaged Element Number
True Value - True Value of Bending Stiffness Parameters
OBS - Monte Carlo Observation Number
Mean - Mean of the Monte Carlo Experiments
% Error - Percent Error of the Mean with Respect to the True Value
ITER - Number of Iterations

Experiment 1, Measurement Error = 0.002%

EL	True Value	OBS 1	OBS 2	OBS 3	Mean	% Error
10	24915.0	24905.5	24917.8	24995.4	24939.6	0.14
	4863.5	4879.3	4868.8	4840.9	4863.0	-0.01
	26300.0	26307.7	26299.0	26307.3	26304.7	0.02
	10367.0	10350.9	10343.9	10346.8	10347.2	-0.19
15	22423.5	22458.3	22458.5	22368.5	22428.4	0.02
	4377.2	4427.2	4411.6	4481.6	4440.1	1.44
	25290.0	25296.4	25292.4	25248.0	25278.9	-0.04
	9330.3	9305.0	9317.1	9335.3	9319.1	-0.12
ITER		7	9	8		

Experiment 2, Measurement Error = 0.02%

EL	True Value	OBS 1	OBS 2	OBS 3	Mean	% Error
10	24915.0	24806.0	24553.8	25746.4	25035.4	0.48
	4863.5	5021.0	4882.6	4632.5	4845.4	-0.37
	26300.0	26374.1	26433.2	26368.4	26391.9	0.35
	10367.0	10209.3	10346.5	10164.8	10240.2	-1.22
15	22423.5	22730.7	22992.1	21876.7	22533.2	0.48
	4377.2	4875.4	4543.2	5412.4	4943.7	12.9
	25290.0	25345.1	25802.3	24872.1	25339.8	0.20
	9330.3	9083.3	9178.8	9391.0	9217.7	-1.21
ITER		9	10	10		

Experiment 3, Measurement Error = 0.2%

EL	True Value	OBS 1	OBS 2	OBS 3
10	24915.0	23018.8	31334.7	34343.0
	4863.5	6359.1	5794.7	2384.0
	26300.0	26763.8	27934.4	26941.5
	10367.0	9051.6	7493.3	8881.0
15	22423.5	22246.3	42974.6	17007.9
	4377.2	9176.9	6707.8	13211.1
	25290.0	25099.9	30374.0	21576.1
	9330.3	7378.0	4415.2	10408.1
ITER		10	10	10

The results presented in the above table illustrate the sensitivity of the identified parameters to measurement error. The first Monte Carlo experiment (0.002% measurement error) shows the first range of input/output error relationship, small measurement errors induce in small errors in the identified parameters. The range of errors in the identified parameters is 0.01% to 1.5%, which is acceptable. However, this level of measurement error is too small for practical applications.

The second Monte Carlo experiment (0.02% measurement error) is also in the linear region of the input/output relationship. Small errors in the measurements result in small errors in the identified parameters. The range of errors in the identified parameters is 0.4% to 13%, which is acceptable. This level of measurement error requires very sensitive instruments.

The third Monte Carlo experiment (0.2% measurement error) shows a nonlinear relationship between measurement error and the identified parameters. The parameter identification algorithm failed to converge for the second and third observations, thus a mean and percent error could not be calculated. This level of error is on the border line of nonlinear behavior and algorithmic failure. If the measurement error is increased beyond this limit of 0.2% the algorithm will probably diverge.

Thus, it is concluded that an acceptable level of measurement error for the above example is 0.02%, which corresponds to a standard deviation of 0.0001 for force and displacement measurements (Ref. [17]). This is a low limit, however it is recommended for algorithmic convergence that the error limit not be close to the nonlinear range of the input/output relationship.

7.0 CONCLUSIONS

In this report, a methodology was devised for the systematic identification of the plate bending stiffnesses of the reinforced concrete Navy's one third scale pier deck model. Bending stiffnesses and transverse shear terms were successfully identified from simulated incomplete static test data. The level of acceptable error was also investigated for this structure. Through this research several conclusions were reached:

The simplified hourglass plate bending element proved to be suitable for modelling plate bending behavior. The element formulation is both easy to program and computationally efficient. The examples presented in this report illustrate the modelling capabilities of this element.

Parameter identification was performed successfully using the hourglass plate bending elements with simulated data. The program PARIS.SAS is capable of detecting large changes in element stiffness properties.

The error tolerance for the Navy's scale model was determined to be 0.02%, which corresponds to a standard deviation of 0.0001 in the force and displacement measurements. The identified parameters are sensitive to measurement errors for this problem. Thus, it is important to have very accurate force and displacement measurements.

It is recommended first to perform parameter identification on the undamaged structure. This would allow analysts to adjust

the stiffness properties of the finite element model to represent the physical model more accurately. Thus, when performing parameter identification on the damaged structure the adjusted stiffness values could be used.

Continued development of this approach of static parameter identification can lead to a practical procedure for the instrumentation, testing and identification of full scale civil engineering structures. The scope of future work for this method of static parameter identification could include: The implementation of this method to a full scale steel or reinforced concrete structure. Load capacity determination of an existing structure based on the identified parameters. Extension of the static parameter identification algorithm to accept two different subsets of degrees of freedom for force and displacement measurements.

REFERENCES

1. U.S. Department of Transportation, Federal Highway Administration, Seventh Annual Report to Congress, "Highway Bridge Replacement and Rehabilitation Program," U.S. Department of Transportation, Washington, D.C., October 1986.

2. Minor, J., White, K.R., and Busch, R.S., "Condition Surveys of Concrete Bridge Components - User's Manual," National Cooperative Highway Research Program Report 312, Transportation Research Board, Washington, D.C., December 1988.

3. Manning, D.G., "Detecting Defects and Deterioration in Highway Structures," National Cooperative Highway Research Program Report Synthesis of Highway Practice 118, Transportation Research Board, National Research Council, Washington, D.C., July 1985.

4. American Society for Testing and Materials, "Recommended Practice for Examination and Sampling of Hardened Concrete in Construction," *Annual Book of ASTM Standards*, ASTM Designation: C-823, Section 4, Volume 04.02, Philadelphia, PA., 1986.

5. Hart, G.C., and Yao, J.T.P., "System Identification in Structural Dynamics," *Journal of the Engineering Mechanics Division*, ASCE, Vol. 103, No. EM6, Proc. Paper 13443, December 1977.

6. Beck, J.L., "Determining Models of Structures from Earthquake Records," *EERL Report No. 78-01*, California Institute of Technology, Pasadena, CA, June 1978.

7. Hudson, D.E., "Dynamic Tests of Full-Scale Structures," *Journal of the Engineering Mechanics Division*, ASCE, Vol. 103, No. EM6, Proc. Paper 13446, December 1977.

8. Liu, S.C., and Yao, J.T.P., "Structural Identification Concept", *Journal of the Structural Division*, ASCE, Vol. 104, No. ST12, Proc. Paper 14207, December 1978.

9. Berman, A., "System Identification in Structural Dynamic Models - Theoretical and Practical Models," *AIAA Structures*, Structural Dynamics and Materials Conference, Paper 84-0929, Palm Springs, CA, 1984.

10. Adelman, H.M., and Haftka R.T., "Sensitivity Analysis of Discrete Structural Systems," *AIAA Journal*, Vol. 24, No. 5, May 1986.

11. Douglas, B.M., and Reid, W.H., "Dynamic Tests and System Identification of Bridges," *Journal of the Structural Division*, ASCE, Vol. 108, No. ST10, October 1982.

12. Werner, S.D., Douglas, B.M., and Crouse, C.B., "System Identification of the Meloland Road Overcrossing", Seismic Engineering: Research and Practice, Proceedings of the Sessions Related to Seismic Engineering at the Structures Congress '89, ASCE, San Francisco, CA, May 1989.

13. Buckle, I.G. Richardson, J.A., and Sveinsson, B.I., "Vertical Response Data from the Dominion Road Bridge Test," Seismic Engineering: Research and Practice, Proceedings of the Sessions Related to Seismic Engineering at the Structures Congress '89, ASCE, San Francisco, CA, May 1989.

14. Sheena, Z., Zalmanovitch, A., and Unger, A., "Theoretical Stiffness Matrix Correction by Using Static Test Results," 24th Israel Annual Conference of Aviation and Astronautics, February 1982.

15. Sanayei, M., and Nelson, R.B., "Identification of Structural Element Stiffnesses from Incomplete Static Test Data", SAE-861793, Aerospace Technology Conference and Exposition, Long Beach, CA, October 1986.

16. Sanayei, M., and Schmolze, J.G., "An Expert System that Identifies Structural Element Stiffnesses for Failure Detection," IASTED-109-083, *Applied Simulation and Modelling*, Santa Barbara, CA, May 1987.

17. Sanayei, M., Identification of Structural Element Stiffnesses from Incomplete Static Test Data, Ph.D. Dissertation, University of California, Los Angeles, June 1986.

18. Bathe, K., Finite Element Procedures in Engineering Analysis, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

19. Curtis, G., and Wheatley, P., Applied Numerical Analysis, Addison-Wesley Publishing Co., Reading, Massachusetts, 1985.

20. Timoshenko, S., and Goodier, J., Theory of Elasticity, McGraw-Hill Book Co., New York, 1970.

21. Nelson, R.B., and Yongjiang, S., "A 3-D Finite Element for Thin Plate/Shell Analysis Based on Hourglass Concepts", *In Review*, University of California, Los Angeles, 1988.

22. Timoshenko, S., and Woinowsky-Krieger, S., Theory of Plates and Shells, McGraw-Hill Book Co., New York, 1970.

23. ANSYS Engineering Analysis System User's Manual, Swanson Analysis Systems, Inc., Houston, Pennsylvania, March 1983.

24. Zienkiewicz, O.C., and Taylor, R.L., The Finite Element Method, McGraw-Hill Book Co., New York, 1989.

25. Wilson, E., "The Static Condensation Algorithm," *The International Journal for Numerical Methods in Engineering*, Vol. 8, 1974.

26. SAS/IML Software, An Interactive Matrix Language, Version 5, SAS Institute, Carry, North Carolina, 1985.

Appendix A PARIS.SAS USER MANUAL

A.1 Introduction

PARIS.SAS (PARameter Identification of Structures) is a three dimensional static parameter identification program based on the algorithm described in chapter two. The program is written in SAS/IML (Ref. [26]), an Interactive Matrix Language, which is a part of the larger package, Statistical Applications Software. SAS/IML was selected for several reasons. The first is the ease of matrix manipulation. SAS/IML has many built in linear algebra operations, such as inverse, transpose, matrix multiplication, etc. The second is that SAS/IML has many built in mathematical and statistical functions; such as cubic spline, eigenvalues, linear regression, etc. The relative ease of programming in SAS/IML makes this an ideal language for finite element research.

A.2 Finite Element Model Input Data

The input data is separated into two parts, the first section requires the input data necessary to generate the finite element model of the structure. SAS/IML accepts input data in free format. There is no need to define the variable dimensions and type. Data may be entered as a matrix, vector, or scalar. A data matrix should start off with a left brace and finish with a right brace, and each row of the matrix should be separated by a comma. A data

vector should also start and end with a brace, blank spaces should separate each entry of the vector. A scalar can be input without braces. A line of IML code must always ends with a semicolon. The contents of the finite element input matrices are as follows:

A.2.1 NDOF = n;

NDOF is the number of kinematic degrees of freedom.

A.2.2 NELEM = n;

NELEM is the total number of finite elements used to define the structural model.

A.2.3 MAT = {1 E_1 ν_1 I_{x1} I_{y1} ,
2 E_2 ν_2 I_{x2} I_{y2} ,
. . . . . ,
n E_n ν_n I_{xn} I_{yn} };

MAT is the matrix that contains the material properties of the elements used to describe the structural model. The first column contains the material number (ascending order). The second column contains the modulus of elasticity. The third column contains Poisson's ratio. The fourth and fifth columns contain the equivalent moments of inertia for a reinforced concrete slab a describe in chapter five.

A.2.4 COORD = {1 X_1 Y_1 Z_1 ,
2 X_2 Y_2 Z_2 ,
. . . . . ,
n X_n Y_n Z_n };

The coordinate matrix, COORD, contains the coordinates of the nodes used to define the structure in a three dimensional cartesian coordinate system. The first column contains the node number (ascending order). The second column contains the X-coordinate. The third column contains the Y-coordinate. The fourth column

contains the Z-coordinate.

A.2.5 CONN = {1 ET_1 MN_1 N_{i1} N_{j1} N_{k1} N_{l1} N_{m1} N_{n1} N_{o1} N_{p1} ,
2 ET_2 MN_2 N_{i2} N_{j2} N_{k2} N_{l2} N_{m2} N_{n2} N_{o2} N_{p2} ,
. . . . . ,
n ET_n MN_n N_{in} N_{jn} N_{kn} N_{ln} N_{mn} N_{nn} N_{on} N_{pn} };

CONN is the matrix that contains the element connectivity for each finite element used. The first column contains the element number in ascending order. Please note, that for parameter identification the hourglass elements must appear first in the connectivity matrix. The second column contains the element type; 5 for a three dimensional isoparametric element, 6 for an hourglass plate bending element. The third column contains the material number corresponding to the row of the material matrix, MAT. Columns four through eleven contain the nodes used to define the finite element. Please note, that the node numbering for both the three dimensional isoparametric element and the hourglass element is counterclockwise (Figure 3.1, Chaper 3). Also note, that for the hourglass plate bending element the node numbering must start from the first quadrant and proceed counterclockwise on the positive Z face of the element (Fig. A.1).

A.2.6 BC = { N_i BC_x BC_y BC_z ,
N_j BC_x BC_y BC_z ,
N_k BC_x BC_y BC_z };

The boundary condition matrix contains the boundary conditions of the structural model. Each row of the boundary condition matrix contains the boundary conditions for a given node, i. Please note, that only the nodes that contain a boundary condition need to be entered in this matrix. Also note, the nodes that are entered in

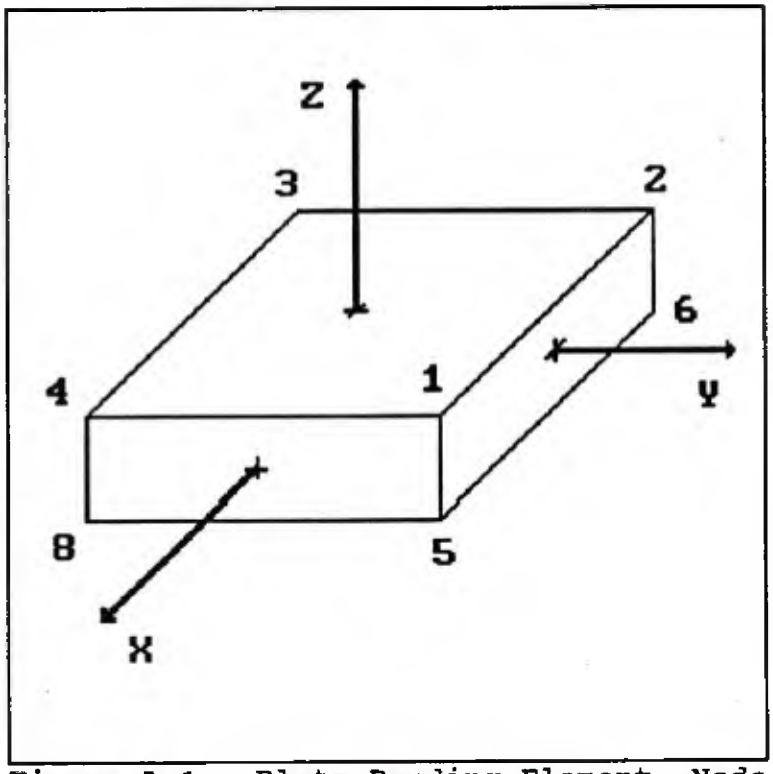

Figure A.1 - Plate Bending Element, Node Numbering

this matrix need to be in ascending order. Columns two, three, and four contain the X, Y, and Z boundary conditions, respectively, for the given node i. A one represents a constrained degree of freedom, a zero represents an unconstrained degree of freedom.

A.3 Parameter Identification Input Data

A.3.1 NJM = n;

NJM is the number of measured degrees of freedom for parameter identification.

A.3.2 JM = {i j k . . . };

JM is the vector that contains the measured degrees of freedom for parameter identification (ascending order).

A.3.3 NPT = n;

NPT is the total number of unknown stiffness parameters, eight per element for a completely anisotropic model.

A.3.4 NPU = n;

NPU is the total number of unknown parameters for parameter identification.

A.3.5 PU = $\{p_i\ p_j\ p_k\ .\ .\ .\ \}$;

PU is the vector that contains the unknown parameters p_i. The entries of PU are determined from the following equation

$$p_i = 8(EN - 1) + i \qquad (A.1)$$

where EN is the element number, and i is the location of the variable [C] matrix or [D] matrix value determined as follows,

$1 = D_{11}$ $2 = D_{12}$ $3 = D_{16}$ $4 = D_{22}$ $5 = D_{26}$ $6 = D_{66}$

$7 = C_{44}$ $8 = C_{55}$

Note, that for the Navy Pier Deck problem an orthotropic material is used. Thus, D_{16} and D_{26} are held constant and equal to zero for all parameter identification examples.

A.3.6 FMEFLAG = f;

FMEFLAG is the force measurement error flag, f = 0 for no force measurement error, f = 1 for force measurement error.

A.3.7 FSD = $\{SD_i\ SD_j\ SD_k\ .\ .\ .\ \}$;

FSD is the vector of pressure gauge standard deviations. The size of this vector is equal to the number of measurement stations.

A.3.8 JMEFLAG = f;

JMEFLAG is the displacement error flag, f = 0 for no displacement measurement error, f = 1 for displacement measurement error.

A.3.7 JSD = {SD_i SD_j SD_k . . . };

JSD is the vector of LVDT standard deviations. The size of this vector is equal to the number of measurement stations.

A.3.8 NOBS = n;

NOBS is the total number of Monte Carlo experiments, n must be greater than or equal to one.

A.3.9 NSTART = n;

NSTART is the restart option for a series of Monte Carlo experiments, n must be greater than or equal to one.

A.3.10 UAVE = u;

UAVE is the average absolute displacement u. This value is only necessary when performing Monte Carlo experiments, for the average displacement is used to scale the random numbers.

A.3.11 PRIFLAG = f;

PRIFLAG is the print flag, f = n print Monte Carlo experiment number n, f = -1 print all experiments.

A.3.12 KPRFLAG = f;

KPRFLAG is the global stiffness matrix print flag, f = 1 print the global stiffness matrix, f = 0 do not print the global stiffness matrix.

A.3.13 OUTFLAG = f;

OUTFLAG is the SAS dataset flag, f = 0 no dataset, f = 1 create a dataset. The SAS dataset is used for computing summary

statistics for each set of Monte Carlo Experiments.

A.3.14 STATFLAG = f;

STATFLAG is the static check of the global stiffness matrix flag, f = 0 no static check, f = 1 perform the static check. The static check consists of applying unit forces at all the measured degrees of freedom as specified in the vector JM. The purpose of the static check is simply to perform a quick check of the overall behavior of the finite element model.

A.3.15 LLCF = n;

LLCF is the lower limit convergence factor n. This factor is multiplied by the initial value of each parameter and is used to constrain the gradient search.

A.3.16 ULCF = n;

ULCF is the upper limit convergence factor n. This factor is multiplied by the initial value of each parameter and is used to constrain the gradient search.

A.3.17 LLPER = n;

LLPER is the lower limit percentage factor n. This percentage factor is multiplied by the initial value of each parameter and is used to reset the unknown parameters when, for the given iteration the step size ($p + \Delta p$) is outside of the lower limit set by LLCF.

A.3.18 ULPER = n;

ULPER is the upper limit percentage factor n. This percentage factor is multiplied by the initial value of each parameter and is used to reset the unknown parameters when, for the given iteration the step size ($p + \Delta p$) is outside of the upper limit set by ULCF.

A.3.19 DKLIM = n;

DKLIM is the convergence limit for Δk. The convergence limit is defined as the absolute change in the stiffness vector per each iteration.

A.3.20 DPLIM = n;

DPLIM is the convergence limit for Δp. The convergence limit is defined as the absolute change in the unknown parameters per each iteration.

A.3.21 FDLIM = n;

FDLIM is the failure detection limit for the parameter identification algorithm. If an unknown parameter becomes less than FDLIM the program stops and a failure detection message is printed.

A.3.22 DIVLIM = n;

DIVLIM is the divergence limit for the inverse of the sensitivity matrix.

A.3.23 CONDN = n;

CONDN is the maximum acceptable condition number of the sensitivity matrix. This parameter is to prevent the inversion of an ill-conditioned sensitivity matrix.

A.3.24 MNI = n;

MNI is the maximum number of iterations for the parameter identification algorithm.

A.3.25 EINT = n;

EINT is the eigen solution interval for the sensitivity matrix. If EINT = 0 the eigen solution is not executed.

A.3.26 DT = {EN_1 i (D_{ij} or C_{ij}) ,
EN_2 i (D_{ij} or C_{ij}) ,
. . . . . ,
EN_n i (D_{ij} or C_{ij}) };

DT is the matrix that contains the true values of the unknown parameters. The first column of the DT matrix contains the element number. The second column contains the location of the variable [C] matrix or [D] matrix value, as defined in section A.3.5. The third column contains the true value of the parameter.

A.4 Output Data

For every run the finite element input data and the parameter identification input data is printed at the beginning of the output. The simulated force and displacement data is printed before and after the addition of measurement errors.

If the print flag is not set equal to zero, the iterative matrices will be printed for each iteration, Δk, Δp, etc. If the eigen interval is active, for every EINT iterations the eigen solution will be performed. The eigenvalues, condition number, row echelon form, and the rank of the sensitivity matrix is printed. Finally at the end of the parameter identification run, a summary table is printed, which contains the values of the unknown parameters for each iteration, and the value of the error function for each iteration.

Appendix B **PARIS.SAS**

```
*OPTIONS LINESIZE=80;

LIBNAME PATH 'CIVILENG:[SSCAMPOL.NAV.COND]';
PROC IML;
PRINT "*****************************************************************";
PRINT "*                          PARIS                                *";
PRINT "*              A program to perform PARameter                   *";
PRINT "*               Identification of Structures                    *";
PRINT "*                                                               *";
PRINT "*                      Developed by                             *";
PRINT "*                                                               *";
PRINT "*           Masoud Sanayei, Assistant Professor                 *";
PRINT "*          Stephen Scampoli, Research Assistant                 *";
PRINT "*                  Tufts University                             *";
PRINT "*                Medford, Massachusetts                         *";
PRINT "*                   August 31, 1989                             *";
PRINT "*                                                               *";
PRINT "*                    DISCLAIMER:                                *";
PRINT "*     The developers of this program make no guarantees         *";
PRINT "*    with respect to the accuracy and reliability of this       *";
PRINT "*    program.  Users must verify their own results.             *";
PRINT "*****************************************************************";


START INPDAT;
*************************************************************************;
*                          INPUT DATA                                   ;
*************************************************************************;
* ENTER THE TITLE FOR THIS RUN;
PRINT "TITLE: NAVY BRIDGE DECK MODEL ";
PRINT "ELEMENTS TEN AND FIFTEEN ARE BADLY DAMAGED";
PRINT "FEM FOR PARAMETER IDENTIFICATION";
PRINT "    >>>>>>>>>>>>>>>>>>> INPUT DATA <<<<<<<<<<<<<<<<<<<<<<<<<";
**>>>ENTER THE NUMBER OF DOFs AND NUMBER OF ELEMENTS;
NDOF = 180;
NELEM = 20;

**>>>ENTER MATERIAL PROPERTIES AS FOLLOWS:
                    COL.#1 MATERIAL NUMBER
                    COL.#2 MODULUS OF ELASTICITY OF CONCRETE
                    COL.#3 POISSON'S RATIO FOR CONCRETE
                    COL.#4 MOMENT OF INERTIA IN THE X-DIRECTION
                    COL.#5 MOMENT OF INERTIA IN THE Y-DIRECTION
                           NOTE: 1 - ZONE A
                                 2 - ZONE B;


MAT = {1 3605 0.19 13.323667 14.06425,
       2 3605 0.19 13.19375  14.370333};
```

```
*>>>>ENTER NODAL COORDINATES AS FOLLOWS:
                    COL.#1 NODE NUMBER
                    COL.#2, COL.#3 & COL.#4 X,Y,Z IN GLOBAL COORDINATES;

COORD = {
1     135     12    30.625,
2     135     12    36,
3     135     27    30.625,
4     135     27    36,
5     135     42    30.625,
6     135     42    36,
7     135     57    30.625,
8     135     57    36,
9     135     72    30.625,
10    135     72    36};

*>>>GENERATE THE REMAINING NODES FROM ABOVE PATTERN;

PTC = COORD;
DO I = 1 TO 5;
   PTC[,1] = PTC[,1] + 10;
   PTC[,2] = PTC[,2] - 27;
   COORD = COORD//PTC;
END;

*>>>ENTER ELEMENT CONNECTIVITY AS FOLLOWS:
                    COL.#1 ELEMENT NUMBER
                    COL.#2 ELEMENT TYPE:
                           - 5 FOR 3D ISOPARAMETRIC BRICK
                           - 6 FOR HOURGLASS PLATE BENDING ELEMENT
                    COL.#3 MATERIAL NUMBER
                    COL.#4 - #11 NODES I J K L M N O P

        NOTE:  NUMBER THE HOURGLASS ELEMENTS FIRST
                      THE MAPPING SUBROUTINES ARE BASED ON THIS
                      ASSUMPTION;

CONN = {
1  6  2  4 14 12 2 3 13 11 1,
2  6  1  6 16 14 4 5 15 13 3,
3  6  1  8 18 16 6 7 17 15 5,
4  6  2  10 20 18 8 9 19 17 7};

*>>>GENERATE THE REMAINING ELEMENTS FROM ABOVE PATTERN;

PTE = CONN;
DO I = 1 TO 4;
   PTE[,1] = PTE[,1] + 4;
   PTE[,{4 5 6 7 8 9 10 11}] = PTE[,{4 5 6 7 8 9 10 11}] + 10;
   CONN = CONN//PTE;
END;
```

```
*>>>   BOUNDARY CONDITION MATRIX
        COL #1  NODE NUMBER
        COL #2 - #4 FX FY FZ
                1 - CONSTRAINED DEGREE OF FREEDOM
                0 - UNCONSTRAINED DEGREE OF FREEDOM
                NOTE:  ONLY THE NODES THAT INVOLVE CONSTRAINTS
                       NEED TO BE IN THE BOUNDARY CONDITION MATRIX;
BC= {
1 0 0 0};

*************************************************************************;
*              GENERATE SYSTEM DEGREE OF FREEDOM MATRIX                 *;
*************************************************************************;
START SYSDOF(COORD,CONN,NELEM,SYDOF,BC,NDF);
NN = NROW(COORD);
NDF = J(NN,4,0);
SYDOF = J(NELEM,26,0);
 DOF = 1;
 I = 1;
 DO J = 1 TO NN;
    IF COORD[J,1] ^= BC[I,1] THEN DO;
       NDF[J,1] = J;
       NDF[J,2] = DOF;
       NDF[J,3] = DOF + 1;
       NDF[J,4] = DOF + 2;
       DOF = DOF + 3;
    END;
    IF COORD[J,1] = BC[I,1] THEN DO;
       NDF[J,1] = J;
       DO L = 2 TO 4;
          IF BC[I,L] = 1 THEN NDF[J,L] = 0;
          IF BC[I,L] = 0 THEN DO;
             NDF[J,L] = DOF;
             DOF = DOF + 1;
          END;
       END;
       I = I + 1;
       IF I > NROW(BC) THEN I = NROW(BC);
    END;
 END;
 DO I = 1 TO NELEM;
    SYDOF[I,1] = I;
    SYDOF[I,2] = 24;
    N = 3;
    M = 5;
    DO J = 1 TO 8;
       SYDOF[I,N:M] = NDF[CONN[I,J+3],2:4];
       N = N + 3;
       M = M + 3;
    END;
 END;
```

```
FINISH;
RUN SYSDOF(COORD,CONN,NELEM,SYDOF,BC,NDF);
*****************************************************************;
*                          END OF ROUTINE                        *;
*****************************************************************;
*>>> ECHO PRINT FEM INPUT DATA;

MT={' MAT.NO' 'E-MODLS' '--NU--' '--Ix--' '--Iy--'};
CD={'NODE #' 'X-COORD' 'Y-COORD' 'Z-COORD'};
CN={'ELEMENT' '   TYPE' '   MAT' ' NODE i' ' NODE j' ' NODE k'
              ' NODE l' ' NODE m' ' NODE n' ' NODE o' ' NODE p'};
SD={'EL' 'DOF'};
FS={'  NODE' ' -Ux-  ' ' -Vy- ' ' -Wz- '};
PRINT NDOF,NELEM,
      MAT  (|COLNAME=MT |),
      COORD(|COLNAME=CD |),
      CONN (|COLNAME=CN FORMAT=7.0|);
      PRINT BC (|COLNAME=FS|);
      PRINT NDF (|COLNAME=FS|);
      PRINT SYDOF (|COLNAME=SD FORMAT=3.0|);

*****************************************************************;
*                    END OF FEM INPUT DATA                      *;
*****************************************************************;

PRINT , ;
PRINT ">>>>>>>>>> INPUT DATA FOR PARAMETER IDENTIFICATION <<<<<<<<<<";
************************* 4/1/89  ********************************** ;

PRINT "INPUT DATA RELATED TO D.O.F.";

NJM  =  10;                          ; * NUMBER OF MEASURED D.O.F. ;
NJU  = NDOF - NJM                    ; * NUMBER OF UNMEASURED D.O.F. ;
JM = {48 72 78 84 102 108 114 132 138 144};
                                       * VECTOR OF MEASURED D.O.F. ;

START DOJU;
NJU = NDOF - NJM;
JU = J(1,NJU,0);
K = 1;
J = 1;
  DO I = 1 TO NDOF;
     IF I ^= JM[,K] THEN DO;          * ROUTINE TO GENERATE;
        JU[,J] = I;                   * VECTOR OF UNMEASURED D.O.F. ;
        J = J + 1;
     END;
     IF I = JM[,K] THEN K = K + 1;
     IF K > NJM THEN K = NJM;
   END;
FINISH;
RUN DOJU;
```

```
PRINT  NJM, NJU, JM ;

PRINT "INPUT DATA RELATED TO PARAMETERS";
NPT  = 160                          ; * NO. OF TOTAL PARAMETERS ;
NPU  = 48                           ; * NO. OF UNKNOWN PARAMETERS ;
PU = {65 66 68 70 73 74 76 78 81 82 84 86 89 90 92 94
      97 98 100 102 105 106 108 110 113 114 116 118 121 122 124 126 129 130
     132 134 137 138 140 142 145 146 148 150 153 154 156 158};
                * VECTOR OF UNKNOWN PARAMETERS ;
PRINT NPT NPU, PU                   ;
PRINT "INPUT DATA RELATED TO FORCE & DISP. ERRORS AT EACH D.O.F.";

FMEFLAG= 0     ; * FORCE MEASUR. ERROR FLAG: 0=NO ERROR  1=FORCE ERROR;
FSD    ={ .0001 .0001 .0001 .0001 .0001 .0001 .0001 .0001
          .0001 .0001  }     ;
                * PRESS GAGE ERROR STAND. DIV.    ;
JMEFLAG= 0     ; * DISP. MEASUR. ERROR FLAG: 0=NO ERROR  1=DISP. ERROR;
JSD    ={ .0001 .0001 .0001 .0001 .0001 .0001 .0001 .0001
         .0001 .0001  }     ;
               * LVDT. ERROR STAND. DIV.         ;
NOBS   = 1     ; * NO. OF TOTAL MONTE CARLO EXPER.              .GE.1  ;
NSTART = 1     ; * RESTART OPTION FOR  MONTE CARLO .LE.NOBS  & .GE.1  ;
UAVE   = 1.0   ; * AVE.(ABS. DISP.) BASED ON LVDT STROKE & UMEAN      ;
PRIFLAG= -1    ; * PRINT FLAG    # = PRINT OBSERV. #   -1 = PRINT ALL ;
KPRFLAG= 0     ; * KBIG PRINT FLAG 0=DO NOT PRINT KBIG 1=PRINT KBIG   ;
OUTFLAG= 0     ; * DATASET FLAG 0=NO DATASET 1=DATASET                ;
STATFLAG= 0    ; * STATFLAG 0=NO STATIC CHECK 1=STATIC CHECK          ;
PRINT FMEFLAG, FSD, JMEFLAG, JSD, NOBS NSTART UAVE, PRIFLAG KPRFLAG
      OUTFLAG, STATFLAG;

PRINT "CONSTRAINTS PLUS CONVERGENCE AND DIVERGENCE  LIMITS";

LLCF    = -1.0      ; * LOWER LIM. CONV. FAC. TIMES PI AS CONSTRAINT  ;
ULCF    =  3.0      ; * UPPER LIM. CONV. FAC. TIMES PI AS CONSTRAINT  ;
LLPER   =  50.      ; * LOWER LIM. PERCENTAGE TIMES PF USED AS PI(^0) ;
ULPER   = 100.      ; * UPPER LIM. PERCENTAGE TIMES PF USED AS PI     ;
DKLIM   = 1.D-8     ; * CONVERGENCE LIMIT FOR DELTAK = KMV - KBARV    ;
DPLIM   = 1.D-8     ; * CONVERGENCE LIMIT FOR DELTAP = P(I+1) - P(I)  ;
FDLIM   = 1.D-06    ; * FAILURE DETECTION LIMIT                       ;
DIVLIM  = 1.D+16    ; * DIVERGECE LIMIT FOR SINV                      ;
CONDN   = 1.D+20    ; * CONDITION NUMBER LIMIT FOR EIGENVALUE PROBLEM ;
MNI     = 10        ; * MAX. NO. OF ITERATIONS                        ;
EINT    = 5         ; * EIGEN SOLUTION INTERVAL FOR |S|T|S| MATRIX    ;
                      * IF EINT = 0  ==> NO EIGEN SOLUTION            ;
PRINT LLCF  ULCF, LLPER  ULPER                                        ;
PRINT DKLIM  DPLIM  FDLIM, DIVLIM  CONDN, MNI, EINT                   ;

*>>>  D AND C MATRIX TRUE VALUES USED TO GENERATE PT VECTOR
       COL. 1  HOURGLASS ELEMENT NUMBER
       COL. 2  LOCATION OF VARIABLE D OR C VALUE
               1 - D11   2 - D12   3 - D16
               4 - D22   5 - D26   6 - D66
```

```
                7 - C44   8 - C55
        COL. 3  VALUE OF Dij or Cij;

DT = {
9    1 49344.8166304,
9    2 9784.63052653,
9    4 53745.2541395,
9    6 20856.7124381,
10   1 24915,
10   2 4863.5,
10   4 26300,
10   6 10367,
11   1 49830.6819172,
11   2 9727.40390869,
11   4 52600.4992738,
11   6 20734.7293843,
12   1 49344.8166304,
12   2 9784.63052653,
12   4 53745.2541395,
12   6 20856.7124381,
13   1 49344.8166304,
13   2 9784.63052653,
13   4 53745.2541395,
13   6 20856.7124381,
14   1 49830.6819172,
14   2 9727.40390869,
14   4 52600.4992738,
14   6 20734.7293843,
15   1 22423.5,
15   2 4377.15,
15   4 25290,
15   6 9330.3,
16   1 49344.8166304,
16   2 9784.63052653,
16   4 53745.2541395,
16   6 20856.7124381,
17   1 49344.8166304,
17   2 9784.63052653,
17   4 53745.2541395,
17   6 20856.7124381,
18   1 49830.6819172,
18   2 9727.40390869,
18   4 52600.4992738,
18   6 20734.7293843,
19   1 49830.6819172,
19   2 9727.40390869,
19   4 52600.4992738,
19   6 20734.7293843,
20   1 49344.8166304,
20   2 9784.63052653,
20   4 53745.2541395,
20   6 20856.7124381};
```

```
PRINT DT (|COLNAME={ 'ELEMENT' 'LOCATION' 'VALUE' }|);

PRINT ">>>>>>>>>>>>>>>>>>>> END OF INPUT DATA <<<<<<<<<<<<<<<<<<<<<<";


PRINT ">>>>>>>>>>>>>>>>>>>>>> DATA CHECK >>>>>>>>>>>>>>>>>>>>>>>>>>>";

NPC  = 0.5*NJM*(NJM+1)                ;
PRINT  NPC NPU                        ;

  IF NPU  > NPC   THEN DO             ;
     PRINT "NOT ENOUGH DATA AVAILABLE TO COMPUTE THE UNKNOWN PARAMETERS";
     PRINT ">>> INPUT ERROR <<<"      ;
  END;

NRPU = NCOL(PU)                       ;
PRINT NPU NRPU                        ;
IF NPU ^= NRPU  THEN PRINT ">>> INPUT ERROR <<<" ;

NRJM = NCOL(JM)                       ;
PRINT NJM NRJM                        ;
IF NJM ^= NRJM  THEN PRINT ">>> INPUT ERROR <<<" ;

NRJU = NCOL(JU)                       ;
PRINT  NRJU                           ;
IF NJU > 0      THEN  IF NJU ^= NRJU  THEN PRINT ">>> INPUT ERROR <<<";

NRFSD= NCOL(FSD)                                    ;
IF FMEFLAG ^= 0 THEN  PRINT  NRFSD                  ;
IF FMEFLAG ^= 0 THEN  IF NJM ^= NRFSD THEN PRINT ">>> INPUT ERROR <<<";

NRJSD= NCOL(JSD)                                    ;
IF JMEFLAG ^= 0 THEN  PRINT NRJSD                   ;
IF JMEFLAG ^= 0 THEN  IF NJM ^= NRJSD THEN PRINT ">>> INPUT ERROR <<<";
PRINT FMEFLAG JMEFLAG;

  IF FMEFLAG ^= 0 | JMEFLAG ^=0  THEN DO            ;
    NRAN = NJM # NOBS                               ;
    PRINT NRAN                                      ;
*S   IF NRAN > NTRAN THEN  PRINT ">>> INPUT ERROR <<<" ;

    NJM2= 2.0#NJM                                   ;
    RAN = J(NRAN,NJM2,0)                            ;

    DO INJM = 1 TO NJM                              ;
      DO N = 1 TO NRAN                              ;
        RAN[N,INJM] = RANNOR(234252)                ;
       END                                          ;
    END                                             ;
    DO INJM = NJM+1 TO NJM2;
```

```
      DO N = 1 TO NRAN;
         RAN[N,INJM] = RANNOR(865785);
      END;
    END;
   PRINT RAN                                                    ;
 END                                                            ;

PRINT LLPER;
IF LLPER = 0   THEN PRINT ">>> INPUT ERROR <<<"   ;
LLPER = LLPER/100.                  ;
ULPER = ULPER/100.                  ;

DO I = 1 TO NRPU;
  IF PU[1,I] ^= (DT[I,1] - 1)#8 + DT[I,2] THEN PRINT I,">>> INPUT ERROR <<<";

END;


FINISH;
RUN INPDAT;


*************************************************************************;
*                                CONSTANTS                                *;
*************************************************************************;
START CONST;

T1 = {
1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0,
0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0,
0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1,
1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0,
0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0  1  0,
0  0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0  1,
1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0,
0  1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0,
0  0  1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1,
1  0  0  1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0  0,
0  1  0  0  1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0,
0  0  1  0  0  1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1,
1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0  0,
0  1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0,
0  0  1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1,
1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0,
0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0  0  1  0  0  1  0,
0  0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0  0  1  0  0  1,
1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0,
0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0  1  0  0 -1  0,
0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0  1  0  0 -1,
1  0  0 -1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0,
0  1  0  0 -1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0,
0  0  1  0  0 -1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1};
```

```
T1 = T1/8;
FINISH;
RUN CONST;

START PROGID;
**********************SUBROUTINES*****************************************;

*************************************************************************;
*                  STRESS - STRAIN RELATIONSHIP, C MATRIX               *;
*************************************************************************;
START CMAT(E,PR,C);
C=J(6,6,0);
* 3D Stress Strain Relationship for Isotropic Material;
C[1,1] = 1;
C[1,2] = PR/(1 - PR);
C[1,3] = C[1,2];
C[2,2] = C[1,1];
C[2,3] = C[1,2];
C[3,3] = C[1,1];
C[4,4] = (1 - 2#PR)/(2#(1 - PR));
C[5,5] = C[4,4];
C[6,6] = C[4,4];
C = C + C` - DIAG(C);
C = ((E#(1 - PR))/((1 + PR)#(1 - 2#PR)))#C;
FINISH;

*************************************************************************;
*         JACOBIAN AND B MATICES FOR 3D ISOPARMETRIC BRICK              *;
*************************************************************************;
START JCBIJ(COORD,CONN,I,K,JCB,DJCB,BIJ);
JCB=J(3,3,0);
DDRST=J(3,8,0);
BIJ=J(6,24,0);

* For a 2 x 2 x 2 Lattice of Integration;
SPWI={ 0.577350269189626   1.0 };

X1=COORD[CONN[I,4],2];  Y1=COORD[CONN[I,4],3];  Z1=COORD[CONN[I,4],4];
X2=COORD[CONN[I,5],2];  Y2=COORD[CONN[I,5],3];  Z2=COORD[CONN[I,5],4];
X3=COORD[CONN[I,6],2];  Y3=COORD[CONN[I,6],3];  Z3=COORD[CONN[I,6],4];
X4=COORD[CONN[I,7],2];  Y4=COORD[CONN[I,7],3];  Z4=COORD[CONN[I,7],4];
X5=COORD[CONN[I,8],2];  Y5=COORD[CONN[I,8],3];  Z5=COORD[CONN[I,8],4];
X6=COORD[CONN[I,9],2];  Y6=COORD[CONN[I,9],3];  Z6=COORD[CONN[I,9],4];
X7=COORD[CONN[I,10],2]; Y7=COORD[CONN[I,10],3]; Z7=COORD[CONN[I,10],4];
X8=COORD[CONN[I,11],2]; Y8=COORD[CONN[I,11],3]; Z8=COORD[CONN[I,11],4];


*Numerical Integration in R, S, and T ;

IF K=1 THEN DO; R=SPWI[1,1] ;  S=R  ;  T=R ;   END;
IF K=2 THEN DO; R=-SPWI[1,1];  S=-R ;  T=-R;   END;
```

```
IF K=3 THEN DO; R=-SPWI[1,1];  S=R  ;  T=-R;   END;
IF K=4 THEN DO; R=SPWI[1,1] ;  S=-R ;  T=R ;   END;
IF K=5 THEN DO; R=SPWI[1,1] ;  S=R  ;  T=-R;   END;
IF K=6 THEN DO; R=-SPWI[1,1];  S=-R ;  T=R ;   END;
IF K=7 THEN DO; R=-SPWI[1,1];  S=R  ;  T=R ;   END;
IF K=8 THEN DO; R=SPWI[1,1] ;  S=-R ;  T=-R;   END;

RP = 1+R;
SP = 1+S;
TP = 1+T;
RM = 1-R;
SM = 1-S;
TM = 1-T;

JCB[1,1]=SP#TP#X1 -SP#TP#X2 -SM#TP#X3 +SM#TP#X4
        +SP#TM#X5 -SP#TM#X6 -SM#TM#X7 +SM#TM#X8;
JCB[2,1]=RP#TP#X1 +RM#TP#X2 -RM#TP#X3 -RP#TP#X4
        +RP#TM#X5 +RM#TM#X6 -RM#TM#X7 -RP#TM#X8;
JCB[3,1]=RP#SP#X1 +RM#SP#X2 +RM#SM#X3 +RP#SM#X4
        -RP#SP#X5 -RM#SP#X6 -RM#SM#X7 -RP#SM#X8;
JCB[1,2]=SP#TP#Y1 -SP#TP#Y2 -SM#TP#Y3 +SM#TP#Y4
        +SP#TM#Y5 -SP#TM#Y6 -SM#TM#Y7 +SM#TM#Y8;
JCB[2,2]=RP#TP#Y1 +RM#TP#Y2 -RM#TP#Y3 -RP#TP#Y4
        +RP#TM#Y5 +RM#TM#Y6 -RM#TM#Y7 -RP#TM#Y8;
JCB[3,2]=RP#SP#Y1 +RM#SP#Y2 +RM#SM#Y3 +RP#SM#Y4
        -RP#SP#Y5 -RM#SP#Y6 -RM#SM#Y7 -RP#SM#Y8;
JCB[1,3]=SP#TP#Z1 -SP#TP#Z2 -SM#TP#Z3 +SM#TP#Z4
        +SP#TM#Z5 -SP#TM#Z6 -SM#TM#Z7 +SM#TM#Z8;
JCB[2,3]=RP#TP#Z1 +RM#TP#Z2 -RM#TP#Z3 -RP#TP#Z4
        +RP#TM#Z5 +RM#TM#Z6 -RM#TM#Z7 -RP#TM#Z8;
JCB[3,3]=RP#SP#Z1 +RM#SP#Z2 +RM#SM#Z3 +RP#SM#Z4
        -RP#SP#Z5 -RM#SP#Z6 -RM#SM#Z7 -RP#SM#Z8;
JCB = JCB/8;

DJCB=DET(JCB);
IF DJCB<=0.0000001 THEN DO;
PRINT ">>>THE FOLLOWING ELEMENT HAS EXCESSIVE TWISTING<<<"
        I(|COLNAME='        ' ROWNAME='ELEMENT'|);
STOP;
END;

DDRST[1,1]=SP#TP;
DDRST[1,2]=-SP#TP;
DDRST[1,3]=-SM#TP;
DDRST[1,4]=SM#TP;
DDRST[1,5]=SP#TM;
DDRST[1,6]=-SP#TM;
DDRST[1,7]=-SM#TM;
DDRST[1,8]=SM#TM;
DDRST[2,1]=RP#TP;
DDRST[2,2]=RM#TP;
DDRST[2,3]=-RM#TP;
```

```
DDRST[2,4]=-RP#TP;
DDRST[2,5]=RP#TM;
DDRST[2,6]=RM#TM;
DDRST[2,7]=-RM#TM;
DDRST[2,8]=-RP#TM;
DDRST[3,1]=RP#SP;
DDRST[3,2]=RM#SP;
DDRST[3,3]=RM#SM;
DDRST[3,4]=RP#SM;
DDRST[3,5]=-RP#SP;
DDRST[3,6]=-RM#SP;
DDRST[3,7]=-RM#SM;
DDRST[3,8]=-RP#SM;
DRST=.125#(SOLVE(JCB,DDRST)) ;

DUXYZ=J(3,24,0);
DVXYZ=J(3,24,0);
DWXYZ=J(3,24,0);

DUXYZ[,{1 4 7 10 13 16 19 22}]=DRST;
DVXYZ[,{2 5 8 11 14 17 20 23}]=DRST;
DWXYZ[,{3 6 9 12 15 18 21 24}]=DRST;

BIJ[1,]=DUXYZ[1,];
BIJ[2,]=DVXYZ[2,];
BIJ[3,]=DWXYZ[3,];
BIJ[4,]=DUXYZ[2,]+DVXYZ[1,];
BIJ[5,]=DVXYZ[3,]+DWXYZ[2,];
BIJ[6,]=DUXYZ[3,]+DWXYZ[1,];

FINISH ;
*************************************************************************;
*    STIFFNESS MATRIX FOR A 3D ISOPARAMETRIC BRICK ELEMENT              *;
*************************************************************************;
START TDISOPEL(E,PR,COORD,CONN,I,KEL);
 KEL=J(24,24,0);
 RUN CMAT(E,PR,C);
*For 2 x 2 x 2 Lattice of Integration WF=1;
DO K=1 TO 8 ;
  RUN JCBIJ(COORD,CONN,I,K,JCB,DJCB,BIJ);
  KELIJ=DJCB#(XMULT(BIJ`,XMULT(C,BIJ)));
  KEL=KEL+KELIJ;
END;
 FREE  JCB BIJ C FIJ KELIJ;
FINISH;


*************************************************************************;
*             JACOBIAN FOR HOURGLASS ELEMENT   (Eq. 44)                 *;
*************************************************************************;
START JCBI(COORD,CONN,I,A,B,H,JCB,DJCB);
JCB=J(3,3,0);
```

```
X1=COORD[CONN[I,4],2];  Y1=COORD[CONN[I,4],3];  Z1=COORD[CONN[I,4],4];
X2=COORD[CONN[I,5],2];  Y4=COORD[CONN[I,7],3];  Z5=COORD[CONN[I,8],4];

A = ABS((X1-X2)/2);
B = ABS((Y1-Y4)/2);
H = ABS((Z1-Z5)/2);

JCB[1,1] = A;
JCB[2,2] = B;
JCB[3,3] = H;

DJCB = DET(JCB);

FINISH;
*************************************************************************;
*                          Bo MATRIX  (Eq. 31)                          *;
*************************************************************************;

START BOMAT(BO);
BO = J(6,12,0);
BO[1,2] = 1;
BO[2,7] = 1;
BO[3,12] = 1;
BO[4,8] = 1;
BO[4,11] = 1;
BO[5,4] = 1;
BO[5,10] = 1;
BO[6,3] = 1;
BO[6,6] = 1;
FINISH;
*************************************************************************;
*                    N INVERSE MATRIX   (Eq. 55)                        *;
*************************************************************************;
START NIMAT(A,B,H,NI);
NI = J(5,5,0);
NI[1,2] = 1/(H#A);
NI[2,3] = 1/(H#B);
NI[3,5] = -2/(A#B);
NI[4,1] = 1;
NI[4,5] = H/A;
NI[5,4] = 1;
NI[5,5] = H/B;
FINISH;
*************************************************************************;
*                    A INVERSE MATRIX  (Eq. 25)                         *;
*************************************************************************;
START AIMAT(COORD,CONN,T1,I,A,B,H,AI);
AI = J(24,24,0);
T2 = J(24,24,0);

P1 = 0;
```

```
P2 = 0;
P3 = 0;

DO J = 4 TO 11 BY 1;
 P1 = P1 + COORD[CONN[I,J],2];
 P2 = P2 + COORD[CONN[I,J],3];
 P3 = P3 + COORD[CONN[I,J],4];
END;

P1 = P1/8;
P2 = P2/8;
P3 = P3/8;

T2[1,1] = 1;
T2[1,4] = -P1/A;
T2[1,7] = -P2/B;
T2[1,10] = -P3/H;

T2[2,4] = 1/A;
T2[3,7] = 1/B;
T2[4,10] = 1/H;

T2[5,2] = 1;
T2[5,5] = -P1/A;
T2[5,8] = -P2/B;
T2[5,11] = -P3/H;

T2[6,5] = 1/A;
T2[7,8] = 1/B;
T2[8,11] = 1/H;

T2[9,3] = 1;
T2[9,6] = -P1/A;
T2[9,9] = -P2/B;
T2[9,12] = -P3/H;

T2[10,6] = 1/A;
T2[11,9] = 1/B;
T2[12,12] = 1/H;


T2[13,13] = 1;
T2[14,16] = 1;
T2[15,19] = 1;
T2[16,22] = 1;
T2[17,14] = 1;
T2[18,17] = 1;
T2[19,20] = 1;
T2[20,23] = 1;
T2[21,15] = 1;
T2[22,18] = 1;
T2[23,21] = 1;
```

```
T2[24,24] = 1;

AI = XMULT(T2,T1);
FREE T2;

FINISH;


*************************************************************************;
*                          D MATRIX    (Eq. 62)                         *;
*************************************************************************;
START DMAT(E,PR,IX,IY,D);
D = J(5,5,0);

* Orthotropic material stress stain relationship for reinforced;
* concrete, Theory of Plates and Shells, Timoshenko, page 366;

D[1,1] = (E/(1-PR##2))#IX;
D[2,2] = (E/(1-PR##2))#IY;
D[1,2] = PR#((D[1,1]#D[2,2])##0.5);
D[3,3] = ((1-PR)/2)#((D[1,1]#D[2,2])##0.5);
K14 = 0.01#MAX(D);
D[4,4] = K14;
D[5,5] = K14;

D = D + D` - DIAG(D);
FINISH;
*************************************************************************;
*                  GENERATE D MATRIX FROM P VECTOR                      *;
*************************************************************************;
START GENDMAT(PV,D,I,K14);
D = J(5,5,0);
K = (I - 1)#8 + 1;
DO L = 1 TO 3;
   DO J = L TO 3;
      D[L,J] = PV[K];
      K = K + 1;
    END;
END;
D[4,4] = K14[I];
D[5,5] = K14[I];
D = D + D` - DIAG(D);
FINISH;

*************************************************************************;
*STIFFNESS MATRIX FOR A 3D HOURGLASS PLATE BENDING ELEMENT (p. 16 & 28)*;
*************************************************************************;

START HOUR(E,PR,PV,COORD,CONN,T1,I,K14,KS,KEL);
 KEL = J(24,24,0);
 RUN CMAT(E,PR,C);
   C[4,4] = PV[((I-1)#8+7),1];
```

```
  C[5,5] = PV[((I-1)#8+8),1];
 RUN JCBI(COORD,CONN,I,A,B,H,JCB,DJCB);
 RUN BOMAT(BO);
 RUN NIMAT(A,B,H,NI);
 RUN AIMAT(COORD,CONN,T1,I,A,B,H,AI);
 RUN GENDMAT(PV,D,I,K14);

 K11=8#DJCB#(XMULT(BO`,XMULT(C,BO)));
 KQ=4#A#B#(XMULT(NI`,XMULT(D,NI)));
 KEL[1:12,1:12] = K11;
 KEL[{14 15 18 19 21},{14 15 18 19 21}] = KQ;
 KEL[13,13] = KS[I];
 KEL[16,16] = KS[I];
 KEL[17,17] = KS[I];
 KEL[20,20] = KS[I];
 KEL[22,22] = 10#KS[I];
 KEL[23,23] = 10#KS[I];
 KEL[24,24] = 10#KS[I];
 KEL = XMULT(AI`,XMULT(KEL,AI));
 FREE JCB BO C D AI NI;

FINISH;

*************************************************************************;
*          ASSEMBLING KFF FOR THE STRUCTURE                             ;
*************************************************************************;
START STIFF(NDOF,NELEM,MAT,PV,COORD,CONN,SYDOF,KFF,T1,K14,KS);
 KFF=J(NDOF,NDOF,0);
DO I=1 TO NELEM;
 IF CONN[I,2]=5  THEN RUN
   TDISOPEL(MAT[CONN[I,3],2],MAT[CONN[I,3],3],COORD,CONN,I,KEL);
 IF CONN[I,2]=6 THEN RUN
   HOUR(MAT[CONN[I,3],2],MAT[CONN[I,3],3],PV,COORD,CONN,T1,I,K14,KS,KEL);

 EN=SYDOF[I,1];
 NDOFE=SYDOF[I,2];
  DO M=1 TO NDOFE;
  DO N=M TO NDOFE;
   AI=SYDOF[EN,M+2];
   BJ=SYDOF[EN,N+2];
   IF AI^=0 & BJ^=0 THEN KFF[AI,BJ]=KFF[AI,BJ]+KEL[M,N] ;
  END;
  END;
END;
KFF=KFF+KFF`-DIAG(KFF)  ;
FINISH;

*************************************************************************;
*     PARTIAL DERIVATIVES FOR 3D HOURGLASS PLATE BENDING ELEMENT      *;
*************************************************************************;

START PHOUR(PDER,COORD,CONN,T1,I,K14,KEL);
```

```
 KEL = J(24,24,0);
 C = J(6,6,0);
    C[4,4] = PDER[((I-1)#8+7),1];
    C[5,5] = PDER[((I-1)#8+8),1];
 RUN JCBI(COORD,CONN,I,A,B,H,JCB,DJCB);
 RUN BOMAT(BO);
 RUN NIMAT(A,B,H,NI);
 RUN AIMAT(COORD,CONN,T1,I,A,B,H,AI);
 RUN GENDMAT(PDER,D,I,K14);
 D[4,4] = 0;
 D[5,5] = 0;

 K11=8#DJCB#(XMULT(BO`,XMULT(C,BO)));
 KQ=4#A#B#(XMULT(NI`,XMULT(D,NI)));
 KEL[1:12,1:12] = K11;
 KEL[{14 15 18 19 21},{14 15 18 19 21}] = KQ;
 KEL = XMULT(AI`,XMULT(KEL,AI));
 FREE JCB BO C D AI NI;

FINISH;

*************************************************************************;
*                COMPUTE THE PARTIAL DERIVATIVE Pi                      ;
*************************************************************************;
START PSTIFF(NDOF,NELEM,MAT,PDER,COORD,CONN,SYDOF,PKBIG,T1,K14);
 PKBIG=J(NDOF,NDOF,0);
DO I=1 TO NELEM;
 IF CONN[I,2]=6 THEN DO;
 RUN PHOUR(PDER,COORD,CONN,T1,I,K14,KEL);
 EN=SYDOF[I,1];
 NDOFE=SYDOF[I,2];
  DO M=1 TO NDOFE;
  DO N=M TO NDOFE;
   AI=SYDOF[EN,M+2];
   BJ=SYDOF[EN,N+2];
   IF AI^=0 & BJ^=0 THEN PKBIG[AI,BJ]=PKBIG[AI,BJ]+KEL[M,N] ;
  END;
  END;
  END;
END;
PKBIG = PKBIG + PKBIG` - DIAG(PKBIG);
FINISH;

*************************************************************************;
*                        PARTITION KBIG                                 *;
*************************************************************************;
START PARTI(KBIG,KAA,KAB,KBA,KBB,JM,JU,NJU) ;
          KAA = KBIG[JM ,JM ]     ;
          IF NJU > 0 THEN  DO     ;
            KAB = KBIG[JM ,JU ]   ;
            KBA = KAB`            ;
            KBB = KBIG[JU ,JU ]   ;
```

```
          END                              ;
FINISH                                     ;

*************************************************************************;
*     PUT THE LOWER TRIANGLE OF A MATRIX IN A VECTOR                    *;
*************************************************************************;
START VECTOR(PKBAR,NJM,SV,NPC)             ;
          SV   = J(NPC,1,0.)               ;
          NN   = 0                         ;
          DO K = 1 TO NJM                  ;
          DO L = K TO NJM                  ;
             NN   = NN + 1                 ;
             SV[NN,]= PKBAR[K,L]           ;
          END                              ;
          END                              ;
FINISH                                     ;
*************************************************************************;
*          EXPAND THE SIZE OF A VECTOR FROM NPU TO NPT                  *;
*************************************************************************;
START EXPAND(DELTAP,NPU,DELTAPE,NPT,PU);
          DELTAPE= J(NPT,1,0.)             ;
          DO  O = 1 TO NPU                 ;
          LPU = PU[,O]                     ;
          DELTAPE[LPU,]=DELTAP[O,];
          END                              ;
FINISH                                     ;

*************************************************************************;
*             STATIC CHECK OF THE GLOBAL STIFFNESS MATRIX               *;
*************************************************************************;
START STATCHK(KBIG,NDOF,JM);
 DISP=J(NDOF,1,0);
 LOAD=J(NDOF,1,0);
 LOAD(|JM`,|)=1;
 DISP=SOLVE(KBIG,LOAD);
 PRINT "DISPLACEMENTS DUE TO UNIT LOADS AT MEASURED DOF";
 PRINT LOAD(|COLNAME='-LOAD-'|);
 PRINT DISP(|COLNAME='-DISP-'|);
FINISH;

*************************************************************************;
*              GENERATE PI, K14 AND KS VECTORS                          *;
*************************************************************************;
START GENPIVEC(NELEM,CONN,MAT,NPT,PI,K14,KS);
PI = J(NPT,1,0);
K14 = J(NELEM,1,0);
KS = J(NELEM,1,0);
L = 1;
DO I=1 TO NELEM;
  IF CONN[I,2]=6 THEN DO;
    RUN DMAT(MAT[CONN[I,3],2],MAT[CONN[I,3],3],MAT[CONN[I,3],4],
             MAT[CONN[I,3],5],D);
```

```
    RUN CMAT(MAT[CONN[I,3],2],MAT[CONN[I,3],3],C);
    DO J = 1 TO 3;
       DO K = J TO 3;
          PI[L] = D[J,K];
          L = L + 1;
       END;
    END;
    K14[I] = 0.01#MAX(D);
    KS[I] = 0.01#MAX(D);
    PI[L] = C[4,4];
    PI[L+1]=C[5,5];
    L = L+2;
  END;
END;
FINISH;
*************************************************************************;
*                      GENERATE THE PT VECTOR                           *;
*************************************************************************;
START GENPTVEC(DT,PT);
 DO I = 1 TO NROW(DT);
    K = (DT[I,1] - 1)#8 + DT[I,2];
    PT[K] = DT[I,3];
 END;
FINISH;
*************************************************************************;
*                          END OF SUBROUTINES                           *;
*************************************************************************;

START PROG3;
USE PATH.KCON;

PRINT "=======================  PROG3  ============================";
PRINT ">>>>>>>>>>>>>>>> COMPUTE PSUEDO-DISPLACEMENTS <<<<<<<<<<<<<<<<";

RUN GENPIVEC(NELEM,CONN,MAT,NPT,PI,K14,KS);
PT = PI;
RUN GENPTVEC(DT,PT);

PFMAT = J(NPT,1,0);
PFMAT = PI||PT;

ERRMAT = J(1,2,0);

RUN STIFF(NDOF,NELEM,MAT,PT,COORD,CONN,SYDOF,KBIG,T1,K14,KS);
READ ALL INTO KCD;
KBIG = KBIG + KCD;
FREE KCD;

IF KPRFLAG^=0 THEN PRINT PT ,KBIG;
IF STATFLAG^=0 THEN RUN STATCHK(KBIG,NDOF,JM);
ABIG = INV(KBIG)                    ;
FREE KBIG                           ;
```

```
IF KPRFLAG^=0 THEN PRINT ABIG      ;
PRINT "MEASURED  FORCES INCLUDING NO ERROR";
F     = I(NJM)                     ;
PRINT F                            ;
PRINT "MEASURED  DISP. INCLUDING NO ERROR";
U     = ABIG[JM,JM]                ;
FREE ABIG                          ;
UTRUE= U                           ;
PRINT  U (|FORMAT=8.6|)            ;
  UABS = ABS(U)                    ;
  UMEAN= SUM(UABS)/(NROW(U)#NCOL(U));
  UABSMIN = MIN(UABS)              ;
  UMIN = MIN( U )                  ;
  UMAX = MAX( U )                  ;
  PRINT UAVE (|FORMAT=8.6|), UMEAN (|FORMAT=8.6|),
        UABSMIN (|FORMAT=8.6|), UMIN (|FORMAT=8.6|),
        UMAX (|FORMAT=8.6|);

  IF JMEFLAG ^= 0 THEN DO          ;
    JME  = J(NJM,NJM,0)            ;
  END                              ;

  IF FMEFLAG ^= 0 | JMEFLAG ^=0  THEN DO;
     KMVR= J(1,NPC  ,0)            ;
     PFR = J(1,NPT+4,0)            ;
     FME = J(1,NJM  ,0)            ;
     PRINT ">>>>>>>>>>>>>>> MONTE CARLO ANALYSIS <<<<<<<<<<<<<<<";
  END                              ;

IF FMEFLAG=0 & JMEFLAG=0 THEN NOBS=1;
* NEWFLAG = -1                     ;
PRINT NSTART NOBS;

IF OUTFLAG ^= 0 THEN DO;
   PRF = J(1,NPT+4,1.0);
END;

FINISH;
RUN PROG3;
PRINT "OUT OF PROG3";


*************************************************************************;
*                        MONTE CARLO ANALYSIS                           *;
*************************************************************************;

START DOMC                         ;

DO MC = NSTART TO NOBS             ;
PRINT MC                           ;
IF PRIFLAG >= 0  THEN NEWFLAG = 0  ;
IF PRIFLAG = MC  |  MC = NSTART  |  MC = NOBS THEN NEWFLAG = -1;
```

```
IF FMEFLAG ^=0 | JMEFLAG ^=0 THEN DO;
      MCU1 = NJM*(MC-1) + 1          ;
      MCU2 = MCU1 + NJM - 1          ;
END                                  ;

*********************************************************************;
*     ADD RANDOM ERROR TO FORCE AND DISPLACEMENT MEASUREMENTS       *;
*********************************************************************;

  IF FMEFLAG ^= 0 THEN DO            ;
  IF NEWFLAG = -1 THEN PRINT "MEASURED  FORCES INCLUDING ERROR";
    DO N = 1 TO NJM                  ;
      FME[1,N] =RAN[MCU1,N]`#FSD[,N];
    END                              ;
  IF NEWFLAG = -1 THEN PRINT FME     ;
    F = I(NJM)                       ;
    DO M = 1 TO NJM                  ;
      F[M,M] = F[M,M] + FME[,M]      ;
    END                              ;
  IF NEWFLAG = -1 THEN PRINT F       ;
  END                                ;

  IF JMEFLAG ^= 0 THEN DO            ;
    IF NEWFLAG = -1 THEN PRINT "MEASURED  DISP. INCLUDING ERROR";
    DO N = 1 TO NJM                  ;
      NPN = N + NJM                  ;
      JME[N,] = RAN[MCU1:MCU2,NPN]`#JSD[,N];
    END                              ;
    IF NEWFLAG = -1 THEN PRINT JME   ;
    DO M = 1 TO NJM                  ;
    DO N = 1 TO NJM                  ;
      U[M,N] = UTRUE[M,N] + UAVE#JME[M,N];
    END                              ;
    END                              ;
    IF NEWFLAG = -1 THEN PRINT "NON-SYMMETRIC MEASURED DISPLACEMENTS";
    IF NEWFLAG = -1 THEN PRINT  U (|FORMAT=8.6|)   ;
  END                                ;

************************************ ;
FINV = INV(F)                        ;
IF NEWFLAG = -1 THEN PRINT FINV      ;
AM = U*FINV                          ;
IF NEWFLAG = -1 THEN PRINT "MEASURED FLEXIBILITY";
IF NEWFLAG = -1 THEN PRINT  AM  (|FORMAT=8.6|)     ;
************************************ ;

  IF FMEFLAG ^= 0 | JMEFLAG ^=0  THEN DO;
   IF NEWFLAG = -1 THEN PRINT "SYMMETRIC MEASURED FLEXIBILITY";
       AM = 0.5#(AM+AM`)             ;
   IF NEWFLAG = -1 THEN PRINT  AM (|FORMAT=8.6|)   ;
  END                                ;
```

```
KM    = INV(AM)                              ;
RUN VECTOR(KM,NJM,KMV,NPC)                   ;
IF NEWFLAG = -1 THEN PRINT  KM               ;
IF NEWFLAG = -1 THEN DO                      ;
   KTEST = XMULT(KM,AM)                      ;
   DETKT = DET(KTEST)                        ;
   PRINT KTEST, DETKT                        ;
END;
IF NEWFLAG = -1 THEN PRINT  KMV              ;
************************************ ;

  IF FMEFLAG ^=0 ¦ JMEFLAG ^=0 THEN DO;
    DO IKMV = 1 TO NPC                       ;
      KMVR[1,IKMV] = KMV[IKMV,1]             ;
    END                                      ;
    PRINT KMVR                               ;
  END                                        ;

*************************************************************************;
*                                ITERATIONS                               *;
*************************************************************************;

IF NEWFLAG = -1 THEN PRINT "SENSIT. ANAL. FOR ID. USING SIM. MEAS.";

PF    = PI                                   ;
DKMAX = 1.                                   ;
DPMAX = 1.                                   ;

IF NEWFLAG = -1 THEN PRINT ">>>>>>>>>>>>  ITERATIONS  <<<<<<<<<<<<";

START ITERS                                  ; * START ITER;
DO  ITER = 1 TO MNI   WHILE ( DKMAX > DKLIM  &  DPMAX > DPLIM );
PRINT ITER;
IF NEWFLAG = -1 THEN PRINT "SET INIT. VAL. TO FINAL VAL. IN CONSTR";

  P = PF                                     ;

START DOIC2                                  ; * Check if Pi is within the range;
  DO IC = 1 TO NPT                           ; * of acceptable values           ;
    IF P[IC,] > ULCF#PI[IC,] THEN DO;
       P[IC,] = ULPER#PI[IC,]                ;
    END                                      ;
    IF P[IC,] < LLCF#PI[IC,] THEN DO;
       P[IC,] = LLPER#PI[IC,]                ;
    END                                      ;
  END                                        ;
FINISH ;
RUN    DOIC2;

*************************************************************************;
*                          COMPUTE DELTA K                                *;
```

```
*****************************************************************;

IF NEWFLAG = -1 THEN PRINT P;
IF NEWFLAG=-1 & KPRFLAG^=0 THEN PRINT "ANALYTICAL STIFFNESS MATRIX" ;
RUN STIFF(NDOF,NELEM,MAT,P,COORD,CONN,SYDOF,KBIG,T1,K14,KS) ;
READ ALL INTO KCD;
KBIG = KBIG + KCD;
FREE KCD;

IF NEWFLAG = -1 & KPRFLAG^=0 THEN PRINT KBIG ;
RUN PARTI(KBIG,KAA,KAB,KBA,KBB,JM,JU,NJU)  ;
FREE KBIG                          ;

IF NJU > 0 THEN  DO                ;
  ABB = INV(KBB)                   ;
  KBAR= KAA - KAB*ABB*KBA          ;
END                                ;

IF NJU = 0 THEN  KBAR= KAA         ;
RUN VECTOR(KBAR,NJM,KBARV,NPC)     ;
DELTAK= KMV - KBARV                ;
IF NEWFLAG = -1 THEN PRINT KBAR, KBARV, KMV, DELTAK   ;
IF NEWFLAG = -1 THEN PRINT "MAX ABS ERROR USED FOR CONV {KM}-{KBAR}";
DKMAX = MAX( ABS(DELTAK) )         ;
IF NEWFLAG = -1 THEN PRINT DKMAX   ;
IF NEWFLAG = -1 THEN PRINT "NORM OF CHANGE IN DELTAK FOR CONV";
NORMDK= SQRT( SSQ(DELTAK) )        ;
IF NEWFLAG = -1 THEN PRINT NORMDK ;
IF NEWFLAG = -1 THEN PRINT "EVALUATION OF ERROR FUNCTION FOR ITER";
ERR = DELTAK`*DELTAK;
ERRMAT = ERRMAT||ERR;
IF NEWFLAG = -1 THEN PRINT ERR;

*****************************************************************;
*      COMPUTE THE PARTIAL DERIV OF KBAR WITH RESPECT TO P(I)     *;
*****************************************************************;

IF NEWFLAG = -1 THEN PRINT "COMP. PARTIAL DERIV. OF KBAR W.R.T. P(I)";
PKBAR = J(NJM,NJM,0.)              ;
S     = J(NPC,NPU,0.)              ;

START DOJJ;
  DO  JJ = 1 TO NPU                ;
    JPU = PU[,JJ]                  ;
    PDER= J(NPT,1,0.)              ;
    PDER[JPU,1] = 1.0              ;
    RUN PSTIFF(NDOF,NELEM,MAT,PDER,COORD,CONN,SYDOF,PKBIG,T1,K14);
    RUN PARTI(PKBIG,PKAA,PKAB,PKBA,PKBB,JM,JU,NJU);
    FREE PKBIG                     ;

    IF NJU > 0 THEN
      PKBAR = PKAA - PKAB*ABB*KBA - KAB*ABB*PKBA + KAB*ABB*PKBB*ABB*KBA;
```

```
    IF NJU = 0 THEN  PKBAR= PKAA  ;
    RUN VECTOR(PKBAR,NJM,SV,NPC)  ;

    START DOII;
       DO II = 1 TO NPC           ;
           S[II,JJ]=SV[II,]       ;
       END                        ;
    FINISH ;
    RUN    DOII;

  END                             ;
FINISH ;
RUN    DOJJ;

IF NEWFLAG = -1 THEN DO           ;
   PRINT "NORM OF THE SENSITIVITIES ONE TO NPU";
   NORMS = J(NPU,1,0);
   DO LL = 1 TO NPU               ;
      NORMS[LL] = SQRT(SSQ(S[,LL])) ;
   END                            ;
   NORMS = (PU`)||NORMS           ;
   PRINT NORMS (|COLNAME = {' PARM ' ' NORM '}|);
END                               ;
     STS= XMULT(S`,S)             ;
*    IF NEWFLAG = -1 THEN PRINT S, STS ;

*************************************************************************;
*           EIGEN SOLUTION TO CHECK FOR LINEAR DEPENDENCIES             *;
*************************************************************************;

PRINT EINT;
START JUMPS;
     IF  EINT ^= 0   THEN DO          ;
     IF  EINT = 1  THEN  GOTO  EVERY  ;
     IF  MOD(ITER,EINT) = 1   THEN DO ;

     EVERY:
      IF NEWFLAG = -1 THEN PRINT "CHECK FOR LINEAR DEPENDENCIES";
      IF NEWFLAG = -1 THEN PRINT "SOLVE THE EIGEN PROBLEM FOR |S|T.|S|
                          MATRIX";

                *    CALL EIGEN (EVAL,EVEC,STS)          ;
                *    CONDNUM = ABS( MAX(EVAL) / MIN(EVAL) ) ;
                *    IF NEWFLAG = -1 THEN PRINT EVAL, CONDNUM, EVEC ;

                     EVAL = EIGVAL(STS)                  ;
                     CONDNUM = ABS( MAX(EVAL) / MIN(EVAL) ) ;
                     IF NEWFLAG = -1 THEN PRINT EVAL, CONDNUM;

    IF NEWFLAG = -1 THEN PRINT "CHECK THE RANK OF  |S|  MATRIX";
    IF NEWFLAG = -1 THEN PRINT "ROW-ECHELON FORM OF AUGMENTED
```

```
                              MATRIX S,DELTAK";

                        SDK = S¦¦DELTAK              ;
                        ESDK= ECHELON(SDK)           ;
                        IF NEWFLAG = -1 THEN PRINT ESDK ;
                        RANK = 0;
                        DO J = 1 TO NPU              ;
                            RANK = ESDK[J,J] + RANK;
                        END                          ;
                        IF NPU > RANK THEN DO        ;
                              PRINT "RANK OF SENSITIVITY MATRIX < NPU";
                              PRINT RANK             ;
                              STOP                   ;
                         END;
                         PRINT RANK;


  IF CONDNUM > CONDN THEN DO          ;
  IF NEWFLAG = -1 THEN PRINT ITER     ;
  IF MNI > ITER THEN PRINT ">>>>>>> ITERATIONS FAILED <<<<<<<<" ;
  STOP;
  END;


     END                              ; *END OF  IF MOD = 1  ;
     END                              ; *END OF IF EINT ^= 0 ;

*****************************************************************;
*                          SOLVE FOR DELTA P                    *;
*****************************************************************;

   IF NPU  = NPC THEN DO              ;
   IF NEWFLAG = -1 THEN PRINT "DIRECT SOLUTION OF  ¦S¦{DP}={DK}";
     SINV  = INV(S)                   ;
 * IF NEWFLAG = -1 THEN PRINT SINV    ;
     DELTAP= SOLVE(S,DELTAK)          ;
   END                                ;

     IF NPU  < NPC THEN DO            ;
     IF NEWFLAG = -1 THEN PRINT "LEAST SQUARES ¦ST¦¦S¦{DP}=¦ST¦{DK}`";
       SINV  = SOLVE(STS,S`)          ;
 *   IF NEWFLAG = -1 THEN PRINT SINV ;
         DELTAP= XMULT(SINV,DELTAK)  ;
     END                              ;

DPMAX = MAX( ABS(DELTAP) )            ;
IF NEWFLAG = -1 THEN PRINT "ERROR {DP} USED FOR CONV.", DPMAX ;

IF NEWFLAG = -1 THEN DO;
    STEST = XMULT(S,DELTAP);
    SERR = DELTAK - STEST;
    SPER = (SERR/DELTAK)#100;
```

```
    STEST = DELTAK||STEST||SERR||SPER;
    PRINT STEST (|COLNAME = {'DELTAK' '-S*DP-' 'ERROR' '-%_ERR-'}|);
END;

MAXSINV= MAX( ABS( SINV ) )           ;
IF MAXSINV > DIVLIM THEN DO           ;
  PRINT "***** SENSITIVITY MATRIX IS SINGULAR *****";
  PRINT MAXSINV DIVLIM                ;
  IF NEWFLAG = -1 THEN PRINT ITER     ;
  IF MNI > ITER THEN PRINT ">>>>>>>>> ITERATIONS FAILED <<<<<<<<<<";
  STOP;
END                                   ;

IF NEWFLAG = -1 THEN PRINT "COMPUTE THE NEW VALUES OF PARAMETERS";

RUN EXPAND(DELTAP,NPU,DELTAPE,NPT,PU)                              ;

PF = P + DELTAPE                      ;
IF NEWFLAG =-1 THEN PRINT DELTAPE, PF;
PFMAT = PFMAT||PF;

*************************************************************************;
*                          FAILURE DETECTION                            *;
*************************************************************************;

* IF MIN( ABS ( PF[PU,1] ) ) < FDLIM  THEN  DO   ;
* IF NEWFLAG = -1 THEN PRINT ITER MNI;
* IF NEWFLAG = -1 THEN PRINT "**** FAILURE DETECTION ***
ITERATION CONVERGED";
* IF MNI > ITER THEN PRINT ">>>>>>>>>>>>>> ITERATIONS CONVERGED
<<<<<<<<<<<<<" ;
* IF MNI = ITER THEN PRINT ">>>>>> ITERATIONS REACHED ITER LIM AND
 QUIT <<<<<" ;
* STOP;
* END;

FINISH                          ; * FINISH JUMPS                  ;
RUN JUMPS;

END                             ; * END OF ITERATIONS FOR "ITER" ;
FINISH                          ; * FINISH ITERS                  ;
RUN    ITERS                    ; * END OF ITER                   ;

ITER = ITER - 1                 ;
*    IF  EINT ^= 0   THEN DO    ;
*      IF NEWFLAG = -1 THEN PRINT "EIGEN SOLUTION FOR |S|T.|S|";
*                    CALL EIGEN(EVAL,EVEC,STS)                    ;
*                    CONDNUM = ABS( MAX(EVAL) / MIN(EVAL) )       ;
*                    IF NEWFLAG = -1 THEN PRINT ITER, STS, EVAL,
                                    CONDNUM, EVEC;
*    END                        ; * END OF IF EINT ^= 0           ;
```

```
*************************************************************************;
*   APPEND THE DATASET FOR SUMMARY STATISTICS FROM MONTE CARLO    *;
*************************************************************************;

START FORAND;
     IF FMEFLAG ^= 0 | JMEFLAG ^=0  THEN DO;
   START DOIPF;
         DO IPF = 1 TO NPT         ;
           PFR[1,IPF] = PF[IPF,1] ;
         END                       ;
   FINISH ;
   RUN    DOIPF;

       PFR[1,NPT+1] = MC           ;
       PFR[1,NPT+2] = ITER         ;
       PFR[1,NPT+3] = DKMAX        ;
       PFR[1,NPT+4] = DPMAX        ;

       IF OUTFLAG ^= 0  THEN DO    ;
          IF MC = 1 THEN PFRMAT = PFR;
          IF MC ^= 1 THEN PFRMAT = PFRMAT//PFR;
       END                         ;

     END                           ;
FINISH;
RUN FORAND;

PRINT PFMAT (|COLNAME = {'--PI--' '--PT--'}|);
PRINT ERRMAT;
PTEMP = PFMAT[,(1:2)];
ETEMP = ERRMAT[,(1:2)];
FREE PFMAT ERRMAT;
PFMAT = PTEMP;
ERRMAT = ETEMP;
FREE PTEMP ETEMP;
END                                ; * end of MONTE CARLO analysis;
FINISH                             ; * FINISH MONTE CARLO ANALYSIS NOBS;
RUN   DOMC                         ; * RUN MONTE CARLO ANALYSIS NOBS;
FINISH                             ; * FINISH PROGID;

RUN PROGID;
CLOSE PATH.KCON;                     * Close active dataset;
START DATASET;
IF OUTFLAG ^= 0 THEN DO;
   CREATE PARMID FROM PFRMAT;        * Open temporary dataset for;
   APPEND FROM PFRMAT;               * summary statistics;
END;
FINISH;
RUN DATASET;
QUIT;
    PROC PRINT DATA=PARMID;            * Print the identified parameters;
    PROC MEANS DATA=PARMID;             * Summary statistics;
```

```
        TITLE 'IDENTIFIED PARAMETERS SUMMARY STATISTICS';
ENDSAS;
```

Appendix C SUBSTRUCT.SAS USER MANUAL

C.1 Introduction

SUBSTRUCT.SAS is program written to perform substructuring on the finite element model of the Navy Bridge Deck. The program is written in SAS/IML (Ref. [26]) and is based on the substructuring algorithm described in Ref. [23], [24], and [25].

C.2 Finite Element Input

The finite element input for the substructuring program is very similar to the finite element input of PARIS.SAS. Only the differences in the input structure of the two programs will be noted here, for the remaining input structure see Appendix A.

C.2.1 JI = {i,j,k . . . };

JI is the vector that will contain the active degrees of freedom of the substructure. The vector JI is generated from the NI vector, thus initially all the program requires as input is the size of this vector.

C.2.2 JC = {i j k . . . };

JC is the vector that will contain the internal degrees of freedom to be condensed out of the substructure. The vector JC is also generated, thus initially all the program requires as input is the size of this vector.

C.2.3 CONN = {1 ET_1 MN_1 N_{i1} N_{j1} N_{k1} N_{l1} N_{m1} N_{n1} N_{o1} N_{p1} ,
2 ET_2 MN_2 N_{i2} N_{j2} N_{k2} N_{l2} N_{m2} N_{n2} N_{o2} N_{p2} ,
. . . . . ,
n ET_n MN_n N_{in} N_{jn} N_{kn} N_{ln} N_{mn} N_{nn} N_{on} N_{pn} };

CONN is the matrix that contains the element connectivity for each finite element used. This format of this matrix is identical to the connectivity matrix presented in Appendix A, expect the hourglass plate bending elements do not need to be numbered first.

C.2.4 NI = {i j k . . . };

NI is the vector that contains the nodes (ascending order), which include the active degrees of freedom of the substructure. This vector of nodes is used to generate the vector of degrees of freedom of interest, JI.

C.3 Output Data

For each run the input data is echo printed. A SAS dataset is generated, which contains the substructure stiffness. This dataset is then used in the parameter identification program to provide the super-element stiffness matrix.

Appendix D **SUBSTRUCT.SAS**

```
*OPTIONS LINESIZE=80;

LIBNAME PATH 'CIVILENG:[SSCAMPOL.NAV.COND]';
PROC IML;
PRINT "*****************************************************************";
PRINT "*                        SUBSTRUCT.SAS                          *";
PRINT "*              A program to perform substructuring              *";
PRINT "*                    of the Navy Pier Deck                      *";
PRINT "*                                                               *";
PRINT "*                        Developed by                           *";
PRINT "*                                                               *";
PRINT "*              Masoud Sanayei, Assistant Professor              *";
PRINT "*             Stephen Scampoli, Research Assistant              *";
PRINT "*                      Tufts University                         *";
PRINT "*                   Medford, Massachusetts                      *";
PRINT "*                      August 31, 1989                          *";
PRINT "*                                                               *";
PRINT "*                       'DISCLAIMER:                            *";
PRINT "*     The developers of this program make no guarantees         *";
PRINT "*    with respect to the accuracy and reliability of this       *";
PRINT "*    program.  Users must verify their own results.             *";
PRINT "*****************************************************************";


START INPDAT;
*****************************************************************************;
*                            INPUT DATA                                     ;
*****************************************************************************;
* ENTER THE TITLE FOR THIS RUN;
PRINT "TITLE: FIVE SPAN NAVY BRIDGE DECK MODEL ";
PRINT "FEM FOR SUBSTRUCTURING";
PRINT "    >>>>>>>>>>>>>>>>>>>>> INPUT DATA <<<<<<<<<<<<<<<<<<<<<<<<";
**>>>ENTER THE TOTAL NUMBER OF DOFs AND NUMBER OF ELEMENTS;
NDOF = 1179;
NELEM = 172;


**>>>ENTER THE SIZE OF THE VECTOR OF ACTIVE DOFs;

JI = J(1,90,0);

**>>>ENTER THE SIZE OF THE VECTOR OF DOFs TO BE CONDENSED OUT OF
          THE GLOBAL STIFFNESS MATRIX;

JC = J(1,1089,0);

**>>>ENTER MATERIAL PROPERTIES AS FOLLOWS:
                    COL.#1 MATERIAL NUMBER
                    COL.#2 MODULUS OF ELASTICITY
```

```
                    COL.#3 POISSON'S RATIO
                    COL.#4 MOMENT OF INERTIA IN THE X-DIRECTION PER INCH
                    COL.#5 MOMENT OF INERTIA IN THE Y-DIRECTION PER INCH
                           1 - ZONE A
                           2 - ZONE B
                           3 - PURE CONCRETE;


MAT = {1 3605 0.19  13.323667 14.06425,
       2 3605 0.19  13.19375  14.370333,
       3 3605 0.19      0          0};


*>>>>ENTER NODAL COORDINATES AS FOLLOWS:
                    COL.#1 NODE NUMBER
                    COL.#2, COL.#3 & COL.#4 X,Y,Z IN GLOBAL COORDINATES;

COORD={
1    216    -72    0,
2    216    -60    0,
3    216    -72    15.3125,
4    216    -60    15.3125,
5    216    -72    30.625,
6    216    -72    36,
7    216    -60    30.625,
8    216    -60    36,
9    216    -45    30.625,
10   216    -45    36,
11   216    -30    30.625,
12   216    -30    36,
13   216    -15    30.625,
14   216    -15    36,
15   216      0    30.625,
16   216      0    36,
17   216     12    30.625,
18   216     12    36,
19   216      0    15.3125,
20   216     12    15.3125,
21   216      0    0,
22   216     12    0,
23   216     27    30.625,
24   216     27    36,
25   216     42    30.625,
26   216     42    36,
27   216     57    30.625,
28   216     57    36,
29   216     72    30.625,
30   216     72    36,
31   216     84    30.625,
32   216     84    36,
33   216     72    15.3125,
34   216     84    15.3125,
```

```
35   216    72    0,
36   216    84    0,
37   216    99    30.625,
38   216    99    36,
39   216   114    30.625,
40   216   114    36,
41   216   129    30.625,
42   216   129    36,
43   216   144    30.625,
44   216   144    36,
45   216   156    30.625,
46   216   156    36,
47   216   144    15.3125,
48   216   156    15.3125,
49   216   144    0,
50   216   156    0};

*>>>GENERATE THE REMAINING NODES FROM ABOVE PATTERN;

PTC = COORD;
DO I = 1 TO 3;
   PTC[,1] = PTC[,1] + 50;
   PTC[,2] = PTC[,2] - 27;
   COORD = COORD//PTC;
END;

COORD2={
201  108    -72    0,
202  108    -60    0,
203  108    -72    15.3125,
204  108    -60    15.3125,
205  108    -72    30.625,
206  108    -72    36,
207  108    -60    30.625,
208  108    -60    36,
209  108    -45    30.625,
210  108    -45    36,
211  108    -30    30.625,
212  108    -30    36,
213  108    -15    30.625,
214  108    -15    36,
215  108      0    30.625,
216  108      0    36,
217  108     12    30.625,
218  108     12    36,
219  108      0    15.3125,
220  108     12    15.3125,
221  108      0    0,
222  108     12    0,
223  108     72    30.625,
```

```
224   108     72     36,
225   108     84     30.625,
226   108     84     36,
227   108     72     15.3125,
228   108     84     15.3125,
229   108     72     0,
230   108     84     0,
231   108     99     30.625,
232   108     99     36,
233   108    114     30.625,
234   108    114     36,
235   108    129     30.625,
236   108    129     36,
237   108    144     30.625,
238   108    144     36,
239   108    156     30.625,
240   108    156     36,
241   108    144     15.3125,
242   108    156     15.3125,
243   108    144     0,
244   108    156     0};


*>>>GENERATE THE REMAINING NODES FROM ABOVE PATTERN;

COORD = COORD//COORD2;
PTC = COORD2;
DO I = 1 TO 4;
   PTC[,1] = PTC[,1] + 44;
   PTC[,2] = PTC[,2] - 27;
   COORD = COORD//PTC;
END;

*>>>ENTER ELEMENT CONNECTIVITY AS FOLLOWS:
                    COL.#1 ELEMENT NUMBER
                    COL.#2 ELEMENT TYPE:
                    - 5 FOR 3D ISOPARAMETRIC BRICK
                    - 6 FOR HOURGLASS PLATE BENDING ELEMENT
                    COL.#3 MATERIAL NUMBER
                    COL.#4 - #11 NODES I J K L M N O P;

CONN = {
1  5  3  4  54 53 3  2  52 51 1,
2  5  3  7  57 55 5  4  54 53 3,
3  5  3  8  58 56 6  7  57 55 5,
4  6  2  10 60 58 8  9  59 57 7,
5  6  1  12 62 60 10 11 61 59 9,
6  6  1  14 64 62 12 13 63 61 11,
7  6  2  16 66 64 14 15 65 63 13,
8  5  3  18 68 66 16 17 67 65 15,
9  5  3  17 67 65 15 20 70 69 19,
10 5  3  20 70 69 19 22 72 71 21,
```

```
11 6  2  24 74 68 18 23 73 67 17,
12 6  1  26 76 74 24 25 75 73 23,
13 6  1  28 78 76 26 27 77 75 25,
14 6  2  30 80 78 28 29 79 77 27,
15 5  3  32 82 80 30 31 81 79 29,
16 5  3  31 81 79 29 34 84 83 33,
17 5  3  34 84 83 33 36 86 85 35,
18 6  2  38 88 82 32 37 87 81 31,
19 6  1  40 90 88 38 39 89 87 37,
20 6  1  42 92 90 40 41 91 89 39,
21 6  2  44 94 92 42 43 93 91 41,
22 5  3  46 96 94 44 45 95 93 43,
23 5  3  45 95 93 43 48 98 97 47,
24 5  3  48 98 97 47 50 100 99 49};


*>>>GENERATE THE REMAINING ELEMENTS FROM THE ABOVE PATTERN;

PTE = CONN;
DO I = 1 TO 2;
   PTE[,{4 5 6 7 8 9 10 11}] = PTE[,{4 5 6 7 8 9 10 11}] + 50;
   PTE[,1] = PTE[,1] + 24;
   CONN = CONN//PTE;
END;

CONN2 = {
73  5  3 154 204 203 153 152 202 201 151,
74  5  3 157 207 205 155 154 204 203 153,
75  5  3 158 208 206 156 157 207 205 155,
76  6  2 160 210 208 158 159 209 297 157,
77  6  1 162 212 210 160 161 211 209 159,
78  6  1 164 214 212 162 163 213 211 161,
79  6  2 166 216 214 164 165 215 213 163,
80  5  3 168 218 216 166 167 217 215 165,
81  5  3 167 217 215 165 170 220 219 169,
82  5  3 170 220 219 169 172 222 221 171,
83  5  3 184 228 227 183 186 230 229 185,
84  5  3 181 225 223 179 184 228 227 183,
85  5  3 182 226 224 180 181 225 223 179,
86  6  2 188 232 226 182 187 231 225 181,
87  6  1 190 234 232 188 189 233 231 187,
88  6  1 192 236 234 190 191 235 233 189,
89  6  2 194 238 236 192 193 237 235 191,
90  5  3 196 240 238 194 195 239 237 193,
91  5  3 195 239 237 193 198 242 241 197,
92  5  3 198 242 241 197 200 244 243 199};


CONN = CONN//CONN2;

CONN3 = {
93   5  3  204 248 247 203 202 246 245 201,
```

```
94   5  3  207 251 249 205 204 248 247 203,
95   5  3  208 252 250 206 207 251 249 205,
96   6  2  210 254 252 208 209 253 251 207,
97   6  1  212 256 254 210 211 255 253 209,
98   6  1  214 258 256 212 213 257 255 211,
99   6  2  216 260 258 214 215 259 257 213,
100  5  3  218 262 260 216 217 261 259 215,
101  5  3  217 261 259 215 220 264 263 219,
102  5  3  220 264 263 219 222 266 265 221,
103  5  3  228 272 271 227 230 274 273 229,
104  5  3  225 269 267 223 228 272 271 227,
105  5  3  226 270 268 224 225 269 267 223,
106  6  2  232 276 270 226 231 275 269 225,
107  6  1  234 278 276 232 233 277 275 231,
108  6  1  236 280 278 234 235 279 277 233,
109  6  2  238 282 280 236 237 281 279 235,
110  5  3  240 284 282 238 239 283 281 237,
111  5  3  239 283 281 237 242 286 285 241,
112  5  3  242 286 285 241 244 288 287 243};


*>>>GENERATE THE REMAINING ELEMENTS FROM THE ABOVE PATTERN;

CONN = CONN//CONN3;
PTE = CONN3;
DO I = 1 TO 3;
   PTE[,{4 5 6 7 8 9 10 11}] = PTE[,{4 5 6 7 8 9 10 11}] + 44;
   PTE[,1] = PTE[,1] + 20;
   CONN = CONN//PTE;
END;

*>>>   BOUNDARY CONDITION MATRIX
        COL #1  NODE NUMBER
        COL #2 - #4 FX FY FZ
               1 - CONSTRAINED DEGREE OF FREEDOM
               0 - UNCONSTRAINED DEGREE OF FREEDOM
               NOTE:  ONLY THE NODES THAT INVOLVE CONSTRAINTS
                      NEED TO BE IN THE BOUNDARY CONDITION MATRIX;
BC= {
1   1  1  1,
2   1  1  1,
22  0  0  1,
35  0  0  1,
50  0  0  1};


*>>>GENERATE THE REMAINING BOUNDARY CONDITIONS FROM THE ABOVE PATTERN;

PTBC = BC;
DO I = 1 TO 3;
   PTBC[,1] = PTBC[,1] + 50;
   BC = BC//PTBC;
```

```
END;

BC2= {
201    1   1   1,
202    1   1   1,
222    0   0   1,
229    0   0   1,
244    0   0   1};


*>>>GENERATE THE REMAINING BOUNDARY CONDITIONS FROM THE ABOVE PATTERN;

BC = BC//BC2;
PTBC = BC2;
DO I = 1 TO 4;
   PTBC[,1] = PTBC[,1] + 44;
   BC = BC//PTBC;
END;

*************************************************************************;
*              GENERATE SYSTEM DEGREE OF FREEDOM MATRIX                 *;
*************************************************************************;
START SYSDOF(COORD,CONN,NELEM,SYDOF,BC,NDF);
NN = NROW(COORD);
NDF = J(NN,4,0);
SYDOF = J(NELEM,26,0);
 DOF = 1;
 I = 1;
 DO J = 1 TO NN;
    IF COORD[J,1] ^= BC[I,1] THEN DO;
       NDF[J,1] = J;
       NDF[J,2] = DOF;
       NDF[J,3] = DOF + 1;
       NDF[J,4] = DOF + 2;
       DOF = DOF + 3;
    END;
    IF COORD[J,1] = BC[I,1] THEN DO;
       NDF[J,1] = J;
       DO L = 2 TO 4;
          IF BC[I,L] = 1 THEN NDF[J,L] = 0;
          IF BC[I,L] = 0 THEN DO;
             NDF[J,L] = DOF;
             DOF = DOF + 1;
          END;
       END;
       I = I + 1;
       IF I > NROW(BC) THEN I = NROW(BC);
    END;
 END;
 DO I = 1 TO NELEM;
    SYDOF[I,1] = I;
    SYDOF[I,2] = 24;
```

```
    N = 3;
    M = 5;
    DO J = 1 TO 8;
       SYDOF[I,N:M] = NDF[CONN[I,J+3],2:4];
       N = N + 3;
       M = M + 3;
    END;
 END;
FINISH;
RUN SYSDOF(COORD,CONN,NELEM,SYDOF,BC,NDF);
*************************************************************************;
*                          END OF ROUTINE                                *;
*************************************************************************;

*>>> GENERATE THE DEGREES OF FREEDOM OF INTEREST;

NI = {167 168 173 174 175 176 177 178 179 180 217 218 223 224
      261 262 267 268 305 306 311 312 349 350 355 356 393 394
      399 400};

PJI = NDF[NI,{2 3 4}];
K = 1;
DO I = 1 TO NROW(PJI);
   DO J = 1 TO 3;
      IF PJI[I,J] ^= 0 THEN DO;
         JI[K] = PJI[I,J];
         K = K + 1;
      END;
   END;
END;

*>>> GENERATE THE DEGREES OF FREEDOM TO BE CONDENSED;

K = 1;
J = 1;
DO I = 1 TO NDOF;
   IF I ^= JI[,K] THEN DO;
      JC[,J] = I;
      J = J + 1;
   END;
   IF I = JI[,K] THEN K = K + 1;
   IF K > 90 THEN K = 90;
END;

*>>> ECHO PRINT FEM INPUT DATA;

MT={' MAT.NO' 'E-MODLS' '--NU--' };
CD={'NODE #' 'X-COORD' 'Y-COORD' 'Z-COORD'};
CN={'ELEMENT' '   TYPE' '   MAT' ' NODE i' ' NODE j' ' NODE k'
               ' NODE l' ' NODE m' ' NODE n' ' NODE o' ' NODE p'};
SD={'EL' 'DOF'};
FS={'  NODE' ' -Ux-  ' ' -Vy- ' ' -Wz- '};
```

```
PRINT NDOF,NELEM,
      MAT  (|COLNAME=MT |),
      COORD(|COLNAME=CD |),
      CONN (|COLNAME=CN FORMAT=7.0|);
      PRINT BC (|COLNAME=FS|);
      PRINT NDF (|COLNAME=FS|);
      PRINT SYDOF (|COLNAME=SD FORMAT=4.0|);
      PRINT JI;
      PRINT JC;
FREE NDF BC PJI CONN2 COORD2 BC2;
FINISH;
RUN INPDAT;

*************************************************************************;
*                      END OF FEM INPUT DATA                            ;
*************************************************************************;

*************************************************************************;
*                  STRESS - STRAIN RELATIONSHIP, C MATRIX                *;
*************************************************************************;
START CMAT(E,PR,C);
C=J(6,6,0)                    ;
* 3D Stress Strain Relationship for Isotropic Material;
C[1,1] = 1;
C[1,2] = PR/(1 - PR);
C[1,3] = C[1,2];
C[2,2] = C[1,1];
C[2,3] = C[1,2];
C[3,3] = C[1,1];
C[4,4] = (1 - 2#PR)/(2#(1 - PR));
C[5,5] = C[4,4];
C[6,6] = C[4,4];
C = C + C` - DIAG(C);
C = ((E#(1 - PR))/((1 + PR)#(1 - 2#PR)))#C;
FINISH;

*************************************************************************;
*                      JACOBIAN AND B MATICES                           *;
*************************************************************************;
START JCBIJ(COORD,CONN,I,K,JCB,DJCB,BIJ);
JCB=J(3,3,0);
DDRST=J(3,8,0);
BIJ=J(6,24,0);

* For a 2 x 2 x 2 Lattice of Integration;
SPWI={ 0.577350269189626   1.0 };

X1=COORD[CONN[I,4],2];  Y1=COORD[CONN[I,4],3];  Z1=COORD[CONN[I,4],4];
X2=COORD[CONN[I,5],2];  Y2=COORD[CONN[I,5],3];  Z2=COORD[CONN[I,5],4];
X3=COORD[CONN[I,6],2];  Y3=COORD[CONN[I,6],3];  Z3=COORD[CONN[I,6],4];
X4=COORD[CONN[I,7],2];  Y4=COORD[CONN[I,7],3];  Z4=COORD[CONN[I,7],4];
X5=COORD[CONN[I,8],2];  Y5=COORD[CONN[I,8],3];  Z5=COORD[CONN[I,8],4];
```

```
X6=COORD[CONN[I,9],2];  Y6=COORD[CONN[I,9],3];  Z6=COORD[CONN[I,9],4];
X7=COORD[CONN[I,10],2]; Y7=COORD[CONN[I,10],3]; Z7=COORD[CONN[I,10],4];
X8=COORD[CONN[I,11],2]; Y8=COORD[CONN[I,11],3]; Z8=COORD[CONN[I,11],4];


*Numerical Integration in R, S, and T ;

IF K=1 THEN DO; R=SPWI[1,1] ;  S=R  ;  T=R ;   END;
IF K=2 THEN DO; R=-SPWI[1,1];  S=-R ;  T=-R;   END;
IF K=3 THEN DO; R=-SPWI[1,1];  S=R  ;  T=-R;   END;
IF K=4 THEN DO; R=SPWI[1,1] ;  S=-R ;  T=R ;   END;
IF K=5 THEN DO; R=SPWI[1,1] ;  S=R  ;  T=-R;   END;
IF K=6 THEN DO; R=-SPWI[1,1];  S=-R ;  T=R ;   END;
IF K=7 THEN DO; R=-SPWI[1,1];  S=R  ;  T=R ;   END;
IF K=8 THEN DO; R=SPWI[1,1] ;  S=-R ;  T=-R;   END;

RP = 1+R;
SP = 1+S;
TP = 1+T;
RM = 1-R;
SM = 1-S;
TM = 1-T;

JCB[1,1]=SP#TP#X1 -SP#TP#X2 -SM#TP#X3 +SM#TP#X4
        +SP#TM#X5 -SP#TM#X6 -SM#TM#X7 +SM#TM#X8;
JCB[2,1]=RP#TP#X1 +RM#TP#X2 -RM#TP#X3 -RP#TP#X4
        +RP#TM#X5 +RM#TM#X6 -RM#TM#X7 -RP#TM#X8;
JCB[3,1]=RP#SP#X1 +RM#SP#X2 +RM#SM#X3 +RP#SM#X4
        -RP#SP#X5 -RM#SP#X6 -RM#SM#X7 -RP#SM#X8;
JCB[1,2]=SP#TP#Y1 -SP#TP#Y2 -SM#TP#Y3 +SM#TP#Y4
        +SP#TM#Y5 -SP#TM#Y6 -SM#TM#Y7 +SM#TM#Y8;
JCB[2,2]=RP#TP#Y1 +RM#TP#Y2 -RM#TP#Y3 -RP#TP#Y4
        +RP#TM#Y5 +RM#TM#Y6 -RM#TM#Y7 -RP#TM#Y8;
JCB[3,2]=RP#SP#Y1 +RM#SP#Y2 +RM#SM#Y3 +RP#SM#Y4
        -RP#SP#Y5 -RM#SP#Y6 -RM#SM#Y7 -RP#SM#Y8;
JCB[1,3]=SP#TP#Z1 -SP#TP#Z2 -SM#TP#Z3 +SM#TP#Z4
        +SP#TM#Z5 -SP#TM#Z6 -SM#TM#Z7 +SM#TM#Z8;
JCB[2,3]=RP#TP#Z1 +RM#TP#Z2 -RM#TP#Z3 -RP#TP#Z4
        +RP#TM#Z5 +RM#TM#Z6 -RM#TM#Z7 -RP#TM#Z8;
JCB[3,3]=RP#SP#Z1 +RM#SP#Z2 +RM#SM#Z3 +RP#SM#Z4
        -RP#SP#Z5 -RM#SP#Z6 -RM#SM#Z7 -RP#SM#Z8;
JCB = JCB/8;

DJCB=DET(JCB);
IF DJCB<=0.0000001 THEN DO;
PRINT ">>>THE FOLLOWING ELEMENT HAS EXCESSIVE TWISTING<<<"
        I(¦COLNAME='        ' ROWNAME='ELEMENT'¦);
STOP;
END;

DDRST[1,1]=SP#TP;
DDRST[1,2]=-SP#TP;
```

```
DDRST[1,3]=-SM#TP;
DDRST[1,4]=SM#TP;
DDRST[1,5]=SP#TM;
DDRST[1,6]=-SP#TM;
DDRST[1,7]=-SM#TM;
DDRST[1,8]=SM#TM;
DDRST[2,1]=RP#TP;
DDRST[2,2]=RM#TP;
DDRST[2,3]=-RM#TP;
DDRST[2,4]=-RP#TP;
DDRST[2,5]=RP#TM;
DDRST[2,6]=RM#TM;
DDRST[2,7]=-RM#TM;
DDRST[2,8]=-RP#TM;
DDRST[3,1]=RP#SP;
DDRST[3,2]=RM#SP;
DDRST[3,3]=RM#SM;
DDRST[3,4]=RP#SM;
DDRST[3,5]=-RP#SP;
DDRST[3,6]=-RM#SP;
DDRST[3,7]=-RM#SM;
DDRST[3,8]=-RP#SM;
DRST=.125#(SOLVE(JCB,DDRST)) ;

DUXYZ=J(3,24,0);
DVXYZ=J(3,24,0);
DWXYZ=J(3,24,0);

DUXYZ[,{1 4 7 10 13 16 19 22}]=DRST;
DVXYZ[,{2 5 8 11 14 17 20 23}]=DRST;
DWXYZ[,{3 6 9 12 15 18 21 24}]=DRST;

BIJ[1,]=DUXYZ[1,];
BIJ[2,]=DVXYZ[2,];
BIJ[3,]=DWXYZ[3,];
BIJ[4,]=DUXYZ[2,]+DVXYZ[1,];
BIJ[5,]=DVXYZ[3,]+DWXYZ[2,];
BIJ[6,]=DUXYZ[3,]+DWXYZ[1,];

FINISH ;
*************************************************************************;
*    STIFFNESS MATRIX FOR A 3D ISOPARAMETRIC BRICK ELEMENT              *;
*************************************************************************;
START TDISOPEL(E,PR,COORD,CONN,I,KEL);
 KEL=J(24,24,0);
 RUN CMAT(E,PR,C);
*For 2 x 2 x 2 Lattice of Integration WF=1;
DO K=1 TO 8 ;
  RUN JCBIJ(COORD,CONN,I,K,JCB,DJCB,BIJ);
  KELIJ=DJCB#(XMULT(BIJ`,XMULT(C,BIJ)));
  KEL=KEL+KELIJ;
END;
```

```
 FREE  JCB BIJ C FIJ KELIJ;
FINISH;


*************************************************************************;
*                              JACOBIAN                                 *;
*************************************************************************;
START JCBI(COORD,CONN,I,A,B,H,JCB,DJCB);
JCB=J(3,3,0);

X1=COORD[CONN[I,4],2];  Y1=COORD[CONN[I,4],3];  Z1=COORD[CONN[I,4],4];
X2=COORD[CONN[I,5],2];  Y4=COORD[CONN[I,7],3];  Z5=COORD[CONN[I,8],4];

A = ABS((X1-X2)/2);
B = ABS((Y1-Y4)/2);
H = ABS((Z1-Z5)/2);

JCB[1,1] = A;
JCB[2,2] = B;
JCB[3,3] = H;

DJCB = DET(JCB);

FINISH;
*************************************************************************;
*                            BO MATRIX                                  *;
*************************************************************************;

START BOMAT(BO);
BO = J(6,12,0);
BO[1,2] = 1;
BO[2,7] = 1;
BO[3,12] = 1;
BO[4,8] = 1;
BO[4,11] = 1;
BO[5,4] = 1;
BO[5,10] = 1;
BO[6,3] = 1;
BO[6,6] = 1;
FINISH;
*************************************************************************;
*                          N INVERSE MATRIX                             *;
*************************************************************************;

START NIMAT(A,B,H,NI);
NI = J(5,5,0);
NI[1,2] = 1/(H#A);
NI[2,3] = 1/(H#B);
NI[3,5] = -2/(A#B);
NI[4,1] = 1;
NI[4,5] = H/A;
NI[5,4] = 1;
```

```
NI[5,5] = H/B;
FINISH;
*************************************************************************;
*                          A INVERSE MATRIX                             *;
*************************************************************************;
START AIMAT(COORD,CONN,I,A,B,H,AI);
AI = J(24,24,0);
T1 = J(24,24,0);
T2 = J(24,24,0);

T1 = {
1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0,
0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0,
0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1  0  0  1,
1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0,
0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0  1  0,
0  0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0  1,
1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0,
0  1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1  0,
0  0  1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0 -1,
1  0  0  1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0  0,
0  1  0  0  1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0,
0  0  1  0  0  1  0  0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1,
1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0  0,
0  1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0 -1  0,
0  0  1  0  0 -1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1,
1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0  0  1  0  0  1  0  0,
0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0  0  1  0  0  1  0,
0  0  1  0  0  1  0  0 -1  0  0 -1  0  0 -1  0  0 -1  0  0  1  0  0  1,
1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0  1  0  0 -1  0  0,
0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0  1  0  0 -1  0,
0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0  1  0  0 -1,
1  0  0 -1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0  0,
0  1  0  0 -1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1  0,
0  0  1  0  0 -1  0  0  1  0  0 -1  0  0 -1  0  0  1  0  0 -1  0  0  1};

T1 = T1/8;

P1 = 0;
P2 = 0;
P3 = 0;

DO J = 4 TO 11 BY 1;
 P1 = P1 + COORD[CONN[I,J],2];
 P2 = P2 + COORD[CONN[I,J],3];
 P3 = P3 + COORD[CONN[I,J],4];
END;

P1 = P1/8;
P2 = P2/8;
P3 = P3/8;
```

```
T2[1,1] = 1;
T2[1,4] = -P1/A;
T2[1,7] = -P2/B;
T2[1,10] = -P3/H;

T2[2,4] = 1/A;
T2[3,7] = 1/B;
T2[4,10] = 1/H;

T2[5,2] = 1;
T2[5,5] = -P1/A;
T2[5,8] = -P2/B;
T2[5,11] = -P3/H;

T2[6,5] = 1/A;
T2[7,8] = 1/B;
T2[8,11] = 1/H;

T2[9,3] = 1;
T2[9,6] = -P1/A;
T2[9,9] = -P2/B;
T2[9,12] = -P3/H;

T2[10,6] = 1/A;
T2[11,9] = 1/B;
T2[12,12] = 1/H;


T2[13,13] = 1;
T2[14,16] = 1;
T2[15,19] = 1;
T2[16,22] = 1;
T2[17,14] = 1;
T2[18,17] = 1;
T2[19,20] = 1;
T2[20,23] = 1;
T2[21,15] = 1;
T2[22,18] = 1;
T2[23,21] = 1;
T2[24,24] = 1;

AI = XMULT(T2,T1);
FREE T1 T2;

FINISH;


*************************************************************************;
*                          D MATRIX                                     *
*************************************************************************;
START DMAT(E,PR,IX,IY,D);
D = J(5,5,0);
```

```
* Orthotropic material stress stain relationship for reinforced;
* concrete, Theory of Plates and Shells, Timoshenko, page 366;

D[1,1] = (E/(1-PR##2))#IX;
D[2,2] = (E/(1-PR##2))#IY;
D[1,2] = PR#((D[1,1]#D[2,2])##0.5);
D[3,3] = ((1-PR)/2)#((D[1,1]#D[2,2])##0.5);
K14 = 0.01#MAX(D);
D[4,4] = K14;
D[5,5] = K14;

D = D + D` - DIAG(D);

FINISH;
*************************************************************************;
*    STIFFNESS MATRIX FOR A 3D HOURGLASS PLATE BENDING ELEMENT          *;
*************************************************************************;

START HOUR(E,PR,IX,IY,COORD,CONN,I,KEL);
 KEL = J(24,24,0);
 RUN CMAT(E,PR,C);
 RUN JCBI(COORD,CONN,I,A,B,H,JCB,DJCB);
 RUN BOMAT(BO);
 RUN NIMAT(A,B,H,NI);
 RUN AIMAT(COORD,CONN,I,A,B,H,AI);
 RUN DMAT(E,PR,IX,IY,D);

 K11=8#DJCB#(XMULT(BO`,XMULT(C,BO)));
 KQ=4#A#B#(XMULT(NI`,XMULT(D,NI)));
 KS = 0.01#MAX(D);
 KEL[1:12,1:12] = K11;
 KEL[{14 15 18 19 21},{14 15 18 19 21}] = KQ;
 KEL[13,13] = KS;
 KEL[16,16] = KS;
 KEL[17,17] = KS;
 KEL[20,20] = KS;
 KEL[22,22] = 10#KS;
 KEL[23,23] = 10#KS;
 KEL[24,24] = 10#KS;
 KEL = XMULT(AI`,XMULT(KEL,AI));
 FREE JCB BO C D AI NI K11 KQ;

FINISH;


*************************************************************************;
*         ASSEMBLING KFF FOR THE STRUCTURE                              ;
*************************************************************************;
START STIFF(NDOF,NELEM,MAT,COORD,CONN,SYDOF,KFF);
 KFF=J(NDOF,NDOF,0);
DO I=1 TO NELEM;
```

```
 IF CONN[I,2]=5  THEN RUN
   TDISOPEL(MAT[CONN[I,3],2],MAT[CONN[I,3],3],COORD,CONN,I,KEL);
 IF CONN[I,2]=6 THEN RUN
   HOUR(MAT[CONN[I,3],2],MAT[CONN[I,3],3],MAT[CONN[I,3],4],
        MAT[CONN[I,3],5],COORD,CONN,I,KEL);
 EN=SYDOF[I,1];
 NDOFE=SYDOF[I,2];
  DO M=1 TO NDOFE;
  DO N=1 TO NDOFE;
   AI=SYDOF[EN,M+2];
   BJ=SYDOF[EN,N+2];
   IF AI^=0 & BJ^=0 THEN KFF[AI,BJ]=KFF[AI,BJ]+KEL[M,N] ;
  END;
  END;
END;
FREE SYDOF COORD CONN KEL MAT;
* KFF=KFF+KFF`-DIAG(KFF)  ;
FINISH;

*************************************************************************;
*                          MAIN PROGRAM                                 *;
*************************************************************************;

RUN STIFF(NDOF,NELEM,MAT,COORD,CONN,SYDOF,KBIG);
KAA = KBIG[JI, JI];
KAB = KBIG[JI ,JC];
FREE JI;
KBB = KBIG[JC ,JC ];
FREE KBIG JC;
KC  = KAA - XMULT(KAB,SOLVE(KBB,KAB`));
FREE KAB KBB KAA;
J1 = (1:36);
J2 = (55:66);
J3 = (85:96);
J4 = (115:126);
J5 = (145:156);
J6 = (175:180);
J = J1||J2||J3||J4||J5||J6;
KCD = J(180,180,0);
KCD[J,J] = KC;
CREATE PATH.KCON FROM KCD;          * Create permanent Sas Dataset to;
APPEND FROM KCD;                    * store the substructure stiffnesses;
QUIT;                               * for PARIS;
ENDSAS;
```