EXTRACTION OF EIGEN-PAIRS FROM BEAM STRUCTURES USING AN EXACT ELEMENT BASED ON A CONTINUUM FORMULATION AND THE FINITE ELEMENT METHOD

by

J. Jara-Almonte

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Mechanical Engineering

APPROVED:

L. D. Mitchell, Chairman

C. Beattie

N. S. Eiss

R. H. Fries

C. E. Knight

R. G. Leonard

October, 1985

Blacksburg, Virginia

EXTRACTION OF EIGEN-PAIRS FROM BEAM STRUCTURES USING AN EXACT ELEMENT BASED ON A CONTINUUM FORMULATION AND THE FINITE ELEMENT METHOD

by

James Jara-Almonte

Committee Chairman: Larry D. Mitchell
Mechanical Engineering

(ABSTRACT)

Studies of numerical methods to decouple structure and fluid interaction have reported the need for more precise approximations of higher structure eigenvalues and eigenvectors than are currently available from standard finite elements. The purpose of this study is to investigate hybrid finite element models composed of standard finite elements and exact-elements for the prediction of higher structure eigenvalues and eigenvectors.

An exact beam-element dynamic-stiffness formulation is presented for a plane Timoshenko beam with rotatory inertia. This formulation is based on a converted continuum transfer matrix and is incorporated into a typical finite element program for eigenvalue/vector problems. Hybrid models using the exact-beam element generate transcendental, nonlinear eigenvalue problems. An eigenvalue extraction technique for this problem is also implemented. Also presented is a post-processing capability to reconstruct the mode shape each of exact element at as many discrete locations along the element as desired.

The resulting code has advantages over both the standard transfer matrix method and the standard finite element method. The advantage

over the transfer matrix method is that complicated structures may be modeled with the converted continuum transfer matrix without having to use branching techniques. The advantage over the finite element method is that fewer degrees of freedom are necessary to obtain good approximations for the higher eigenvalues. The reduction is achieved because the incorporation of an exact-beam-element is tantamount to the dynamic condensation of an infinity of degrees of freedom.

Numerical examples are used to illustrate the advantages of this method. First, the eigenvalues of a fixed-fixed beam are found with purely finite element models, purely exact-element models, and a closed-form solution. Comparisons show that purely exact-element models give, for all practical purposes, the same eigenvalues as a closed-form solution. Next, a Portal Arch and a Verdeel Truss structure are modeled with hybrid models, purely finite element, and purely exact-element models. The hybrid models do provide precise higher eigenvalues with fewer degrees of freedom than the purely finite element models. The purely exact-element models were the most economical for obtaining higher structure eigenvalues. The hybrid models were more costly than the purely exact-element models, but not as costly as the purely finite element models.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my wife

for her support and encouragement throughout the pursuit of this degree. I must also thank my son , who with his toothless smile, taught me to place things in their proper perspective, and assign them their proper priority. I am also indebted to my advisor Larry D. Mitchell for his patience throughout this study and the several drafts of this dissertation.

There are others, too numerous to list, people who have assisted me these past three years with their help and/or guidance. I feel that there are a few who I must mention. First, Dr. Chris Beattie who guided me in the way of handling poles. Second, Drs. M.P. Kamat, and W.L. Hallauer for sharing their computer codes with me. Third, Mr. E. Sitler and Ms. Z. Terry who got paperwork through the university bureaucracy. Fourth, Dr. C. E. Knight for his comments throughout this work.

Last, but not least I would like to thank the Union Carbide Corporation for their financial support during my graduate education. It was an honor for me to become a Union Carbide Fellow during my M. S. work. Without their continued financial support it may have been impossible to pursue this degree immediately after my M. S..

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The finite element method (FEM)[+] has become a commonly accepted problem solving tool in many different fields of study. The basic premise behind this method is the ability to approximate a continuous solution by a set of piecewise continuous functions over the same domain. Hence, the large domain is broken down to a set of small regions called elements and the governing differential equations of the entire system are solved over each element. Theoretically, as the elements become infinitesimally small, the solution obtained with the finite element method will converge on the solution obtained by solving the differential equations of the continuous system.

The idea of a finite element solution, the approximation of the continuous solution by piecewise continuous functions, was first postulated by Courant [1][*] in 1943. He presented the solution to a continuum problem by using continuous functions defined only over finite triangular regions. The union of all the triangular regions represented the solution to the continuum problem. At the time, Courant's work could not be applied directly to engineering applications. The solution of the large system of simultaneous equations needed to model engineering problems was intractable by hand calculations.

It was not until 1956 that the finite element method came into the realm of applicable mathematics, thanks to second-generation computers.

[+] Acronyms in parentheses will be used in the remainder of this dissertation to represent the phrase preceding the parentheses.

[*] Numbers in square brackets refer to references at the end of this dissertation.

The paper by Turner, et al [2] is considered the cornerstone in the development of the finite element method. This work presented two elements for the analysis of aircraft structures, a truss element and an inplane load plate element. However, the nature of the method presented by Turner, et al is such that several boundary value problems can be solved:

a) equilibrium problems (static, time-independent problems),
b) eigenvalue problems, and
c) transient or propagation problems (time dependent).

The work to be presented in the next sections deals with the second class of problems, the eigenvalue problems.

Since the publication of the Turner et al paper, the number of applications for the FEM has steadily increased through the years, as evidenced by the hundreds of studies published each year either on the FEM or having used the FEM. Recent work has tried to couple the effects of fluids on structures [3-5]. This coupling created a set of nonlinear simultaneous equations. One way to linearize the set of equations is to use modal analysis on the fluid and structure. For this linearization to provide acceptable results, fairly precise higher modes of the structure are required. Unfortunately, the standard eigenextraction algorithms in the FEM are not as effective in extracting higher eigenvalues as they are in extracting lower eigenvalues [6].

The objective of this work was to explore an enhancement to the FEM that would allow a more accurate extraction of higher eigenvalues and eigenvectors. The FEM is an entrenched analytical tool in structural analysis. Therefore, an enhancement to the FEM would allow practicing engineers to use it without much retraining. The enhancement to be

studied is the incorporation of a so-called "exact" element to the FEM library. Thus, the enhancement is in reality a hybrid method. Work leading up to this hybrid method is presented in the Literature Review section.

An "exact" element arises whenever the boundary value problem is solved exactly over the element domain, and not by approximating functions as proposed by Courant. In particular, the element studied here is a beam. The development of an "exact" beam element may be obtained directly from continuum mechanics equations. However, in this study the equations for the "exact" beam element are obtained from transforming a continuum transfer matrix
representation of a beam into a equivalent dynamic stiffness matrix. This route was chosen because the transfer matrix method is also commonly used by practicing engineers. Furthermore, extensive libraries of continuum transfer matrices have been developed through the years. Practically anyone of these continuum transfer matrices may be converted to a dynamical stiffness matrix and added to the library of a FEM program. The transformation technique employed is presented in the Theory section, along with the FEM beam element matrices.

The addition of an "exact" element to a FEM model makes the resulting eigenvalue problem nonlinear. Therefore, the models using "exact" elements require a special eigenvalue extraction if simplifying assumptions are to be avoided. A rather crude eigenvalue extraction method is presented in the Program Description section. Although this extraction method is inefficient, it is still possible to obtain the higher eigenvalues at a reduced computational cost, for the examples presented here.

Three examples will be presented in the Numerical Examples section, a fixed-fixed beam, a Portal Arch and a Verdeel Truss. In the fixed-fixed beam example, the accuracy of the exact element is established empirically by comparing it with a closed-form solution. Also in this example, the finite element beam formulation used is compared against a commercial code. In the second and third examples, the Portal Arch and Verdeel Truss, the structures are first modeled with purely FEM beam elements and purely exact elements, to establish how many finite element beams are required to produce acceptable eigenvalues. Then, the structures are modeled with a combination of exact and finite element beams. The results from these hybrid models are compared against the purely finite element results and the exact element results to establish the accuracy and usefulness of the hybrid model.

LITERATURE REVIEW

This section is a summary of the recent work leading up to the proposed enhancement of the FEM eigenvalue solution. This enhancement is the incorporation of a continuum transfer matrix into a standard FEM set of equations. Therefore, recent work in the standard FEM eigenvalue formulation, the exact displacement eigenvalue problem, and the finite element-transfer matrix method is reviewed . Work in each one of these areas is presented below, preceded by a brief introduction to each area.

A. The Standard Finite Element Eigenvalue Problem:

The standard FEM eigenvalue problem in structural analysis is expressed as $[K]\mathbf{d}=\omega^2[M]\mathbf{d}$, where, [K] and [M] are, respectively, the global stiffness and mass matrices, $\mathbf{d}$ is the nodal displacement vector, and ω is the circular frequency. The mass and stiffness matrices are assembled from element matrices, which are dependent on the element formulation (discretization scheme) chosen by the analyst. The common way to formulate these matrices is through either variational methods, such as the Rayleigh-Ritz method; through weighted residual methods, such as the Galerkin method; or the least squares methods [6-11]. The variational formulation yields a functional that characterizes the solution as being stationary at the eigenvalues. For example, if Rayleigh's quotient is chosen as the functional, then the stationary values of Rayleigh's quotient:

$$R = \frac{\mathbf{d}^T[K]\mathbf{d}}{\mathbf{d}^T[M]\mathbf{d}} \tag{1}$$

would be in the vicinity of the natural frequencies of the continuous structure approximated by [K] and [M].

Recent work in the FEM has tried to take into account the effect of fluids upon structures and vice-versa [3-5]. This type of problem generates nonlinear equations. Rather than solving complicated nonlinear equations Sung and Nefske [3] linearized the problem through modal analyses of the structure and the fluid independently. For this method to provide accurate results, fairly precise higher modes of the structure are required. Sung and Nefske found that the extraction of higher eigenvalues and eigenvectors in large structures was a difficulty for their linearization scheme. Using NASTRAN [12], they were unable to obtain sufficiently accurate eigenvalues beyond 110 Hz to provide satisfactory results from their linearization scheme.

Strang and Fix [13] have quantified the deviation between the eigenvalues of a large FEM discretization (Λ_i) from those of the continuous system (λ_i). Let the continuous system eigenvalue problem be expressed by,

$$L\,\mathbf{u} = \lambda\, m\,\mathbf{u}\,, \quad \mathbf{u} \in \mathbf{S} \tag{2}$$

where, L is a linear, homogeneous, self-adjoint differential operator of order 2p, $\mathbf{u}$ is the displacement function, m is the mass per unit length, and $\mathbf{S}$ is the domain over which Eq. (2) applies. The boundary constraints have not been included, but it is understood that they are needed for the solution of Eq. (2). In a variational solution to Eq. (2) the admissible or energy functions would belong to a space κ^p; that is, they would need to be at least p times differentiable. If the continuous system described by Eq. (2) is discretized by a FEM model with a

large number of degrees of freedom, n, and with interpolation function of order k, then Strang and Fix have proven that,

$$\Lambda_i - \lambda_i = c\, n^{-2(k-p)}, \quad i=1, 2, 3, \ldots \tag{3}$$

where, c is a real constant. Thus, theoretically the eigenvalues of a FEM model should approximate the actual eigenvalues as n goes to infinity. Although it is reassuring to know that FEM models are guaranteed to converge on the true eigenvalues of a continuous system, it is not feasible nor desirable to have extremely large models. Large FEM models require more storage space, more computation time, and have increased computational truncation errors.

Since it became obvious that large models were less desirable, much work has been done to reduce the number of degrees of freedom of FEM models. One of the first, and presently more popular, techniques [14, 15] to reduce degrees of freedom is known in the static structural analysis area as "substructuring", "partitioning", or "nodal condensation". This technique was introduced by Kron [16, 17] in the late 1950's to condense out nodes in electrical networks. Kron's Diakoptics technique was adapted to static structural analysis in the early 1960's [18-21], and used to develop so-called "mixed methods" [22, 23]. In these methods, the substructure may be analyzed by the force method, while the rest of the structure is analyzed by the displacement method. Later, in 1965, Guyan [24] and Irons [25] extended the technique to dynamic structural analysis. This technique is commonly referred to as "dynamic substructuring", and "dynamic condensation".

Suppose there is a structure of (m+s) degrees of freedom. If there is a distinct substructure of s degrees of freedom then the standard FEM

eigenvalue problem can be rewritten as,

$$\begin{bmatrix} K_{mm} & K_{ms} \\ K_{ms}^T & K_{ss} \end{bmatrix} \begin{Bmatrix} \mathbf{d}_m \\ \mathbf{d}_s \end{Bmatrix} = \omega^2 \begin{bmatrix} M_{mm} & M_{ms} \\ M_{ms}^T & M_{ss} \end{bmatrix} \begin{Bmatrix} \mathbf{d}_m \\ \mathbf{d}_s \end{Bmatrix} \tag{4}$$

where, K_{mm}, M_{mm} are the master stiffness and mass matrices,

K_{ss}, M_{ss} are the slave stiffness and mass matrices,

K_{ms}, M_{ms} are the coupling mass and stiffness matrices,

$\mathbf{d}_m$ is the vector of master degrees of freedom, and

$\mathbf{d}_s$ is the vector of slave degrees of freedom.

If the second set of equations is solved for d_s, then,

$$\mathbf{d}_s = -(K_{ss} - \omega^2 M_{ss})^{-1} (K_{ms} - \omega^2 M_{ms})^T \mathbf{d}_m \tag{5}$$

provided $(K_{ss} - \omega^2 M_{ss})$ is nonsingular. When Eq. (5) is substituted back into the first set of equations, in Eq. (4), the result is,

$$[(K_{mm} - \omega^2 M_{mm}) - (K_{ms} - \omega^2 M_{ms})(K_{ss} - \omega^2 M_{ss})^{-1} (K_{ms} - \omega^2 M_{ms})^T] \mathbf{d}_m = \mathbf{0} \tag{6}$$

The order of the eigenvalue problem in Eq. (6) is described by the number of master degrees of freedom, m. The remaining s degrees of freedom have been eliminated from the eigenvalue problem. These s degrees of freedom, however, can be reconstructed from Eq. (5). Because the reconstruction of the s degrees of freedom depends on the m degrees of freedom, the former are usually called the "slaves" and the latter the "masters".

The reduction of the size of the matrices in Eq. (6) is accompanied by a more complicated eigenvalue extraction, since the reduced system of equations does not resemble an algebraic eigenvalue problem. The second matrix in Eq. (6) contains frequency dependent terms which are off the main diagonal. For this reason several methods have been presented in the literature to facilitate the solution of the eigenvalue problem in

Eq. (6), by eliminating these off-diagonal, frequency-dependent terms.

In his original paper, Irons [25] argued that if the slaves were chosen such that they were the ones with the smallest inertia forces then the terms $\omega^2 M_{ss}$ could be set to zero. Furthermore, if the FEM discretization generates a diagonal mass matrix ($M_{ms}=\mathbf{0}$), then Eq. (6) can be rewritten as,

$$[(K_{mm} - K_{ms}K_{ss}^{-1}K_{ms}^{T}) - \omega^2 M_{mm}]\ \mathbf{d}_m = \mathbf{0} \tag{7}$$

This reduction of Eq. (6) is commonly used in practice as evidenced by Rao [11] and commercial codes such as ANSYS [14] and SUPERB [15]. This technique is usually referred to as the "Guyan Reduction" or "Static Condensation". This second name arises from the fact that Eq. (7) can also be obtained if ω is set to zero for the slaves. Equation (7) can also be obtained if the slave degrees of freedom are considered massless. In this case, Meirovitch [6] points out that Eq. (6) is in reality a fourth order polynomial in ω, whereas Eq. (7) is only a second order polynomial. Thus, a static condensation is tantamount to discarding the higher order terms in ω. This is possible as long as the coefficients of ω^4 are small. They are only small whenever the slaves are chosen from areas of the structure with large stiffness and low mass.

To improve the accuracy of the dynamic substructuring some attempts [26, 27] have been made to take into account the higher order terms in ω. The increased accuracy of the approximations made to solve Eq. (6) were offset by the increased computational effort. Thus, it proved more efficient to establish guidelines for choosing master and slave degrees

of freedom. Henshell and Ong [28] devised an automatic selection of the masters and Thomas [29] developed an approximate error bound for the eigenvalues based on these selected masters.

B. The Exact Method (Dynamic Stiffness Method):

Dynamic (dynamical) stiffness matrices are matrices whose elements contain transcendental functions. These matrices arise in structural analysis whenever a structure is modeled by continuous elements using the displacement method. Continuous elements are often called the exact elements since the boundary value problem is solved exactly over each element without a-priori assumptions of the shape functions (interpolation functions) and without the use of admissible functions. This is why the library of exact elements is limited to trusses, rods in torsion, beams, and simple plate geometries [30].

The free-vibration problem in the exact method is expressed as,

$$[K(\omega)]\ \mathbf{d} = \mathbf{0} \tag{8}$$

where, $[K(\omega)]$ is the dynamic stiffness matrix whose elements are functions of the circular frequency ω, and $\mathbf{d}$ is the displacement vector of the juncture points at which elements meet. Theoretically, Eq. (8) is the limit of Eq. (6) when the number of slave degrees of freedom s tends toward infinity. In Eq. (3) it was seen that the eigenvalues of the discretized model will converge on the eigenvalues of the continuous system as n goes to infinity. It follows then that if s is allowed to go to infinity then Eqs. (6) and (8) will give the same eigenvalues. For this reason, Eq. (8) may be viewed as a substructure technique in which an infinity of degrees of freedom have been condensed out every

time an exact element is used.

The eigenvalue problem in Eq. (8) is a nonlinear eigenvalue problem. This precludes the use of algorithms developed for the linear eigenvalue problem, such as those incorporated into commercial FEM codes [12, 14, 15]. Therefore, it is desirable to linearize Eq. (8) in some way while still retaining the essence of the eigenvalue problem. Several linearization attempts have been studied [31-34]. Richards and Leung [34] proved that a linearization of the form,

$$(\mathbf{B}_A - \omega^2 \mathbf{P}_A)\,\mathbf{d} = \mathbf{0} \tag{9}$$

where, $\mathbf{B}_A = [K(\omega_A)] + \omega_A{}^2 \mathbf{P}_A$,

$$\mathbf{P}_A = \frac{1}{2\omega}\left[\frac{\partial [K(\omega)]}{\partial \omega}\right]_{\omega = \omega_A},$$

ω_A is an arbitrary frequency,

will generate positive definite matrices $\mathbf{B}_A$ and $\mathbf{P}_A$ as long as the system being modeled is constrained against rigid-body motion. Positive definite matrices are desirable since there are efficient algorithms for the extraction of eigenvalues from a system of positive definite matrices [6]. In Eq. (9), the trial circular frequency ω_A will be equal to ω only at an eigenvalue of Eq. (8). Hopper, Simpson and Williams [32] converted Eq. (9) to a form better suited for computer implementation by evaluating the partial derivative in Eq. (9) at two trial frequencies ω_1 and ω_2. This linearization gave excellent results by providing bounds on all the eigenvalues below the first pole of Eq. (8), as long as ω_1 and ω_2 are also below the first pole. Unfortunately, Hopper et al were unable to establish any bound criteria above the first pole of Eq. (8).

A pole of Eq. (8) is a nonzero frequency at which Eq. (8) is satisfied and det[K(ω)] is either indeterminate or nonzero. Since det[K(ω)] $\neq$ 0, Eq.(8) can only be satisfied if $\mathbf{d}=\mathbf{0}$; that is, when the juncture points of the continuous element model are stationary. This does not imply that the structure is stationary, it means the connection points are fixed while the rest of the structure vibrates. In other word, the juncture points become the vibration nodes (not nodes in the FEM sense) of the mode shape. Since these poles satisfy Eq. (8) they are in reality system eigenvalues. They will not, however, be called eigenvalues here, since the equivalent linear eigenvalue problem is not satisfied at these frequencies. The nontrivial solution of a linear eigenvalue problem of positive definite matrices must have $\det[K - \omega^2 M] = 0$, and $\mathbf{d}\neq\mathbf{0}$, at an eigenvalue.

In the FEM dynamic substructuring technique the slave degrees of freedom correspond to a physical substructure. The eigenvalue problem for this substructure alone is $(K_{ss} - \omega^2 M_{ss})\mathbf{d}_s=\mathbf{0}$, where the terms have been defined in Eq. (4). Naturally, a substructure eigenvalue will make $(K_{ss}-\omega^2 M_{ss})$ singular. In the exact displacement formulation, the poles of Eq. (8) correspond to the frequencies which make $(K_{ss}-\omega^2 M_{ss})$ singular. For this reason, the poles of an exact displacement formulation are also referred to as substructure eigenvalues.

Another way of solving Eq. (8) is through frequency scanning methods. References [8-11] outline the frequency search method for the linear eigenvalue problem $[K-\omega^2 M]\mathbf{d}=\mathbf{0}$. They recommend that the eigenvalues be counted using the Sturm Sequence Theorem, since there is the possibility of missing eigenvalues in a frequency scan. Wittrick and

Williams [35, 36] have modified the Sturm Sequence theorem for a nonlinear eigenvalue problem to take into account the infinity of eigenvalues contained in Eq. (8). Their formula can determine the number of structure eigenvalues below an arbitrary frequency ω_A,

$$J(\omega_A) = J_o(\omega_A) + s[K(\omega_A)] \tag{10}$$

where, $J(\omega_A)$ is the number of structure eigenvalues below the chosen ω_A, $J_o(\omega_A)$ is the number of poles lying below ω_A, $s[K(\omega_A)]$ is the sign count of the dynamic stiffness matrix in Eq. (8). The sign count is the number of negative terms along the main diagonal of $[K(\omega_A)]$ after it has been upper-triangularized, without pivoting. This counting technique has been used in several studies [37-39], where a system of equations similar to Eq. (8) had to be solved.

Equation 10 has some interesting consequences. For example, suppose a cantilever beam is to be analyzed with one exact beam element for in-plane vibration. Such a model has 3 active degrees of freedom at the free end. They are the longitudinal displacement, the transverse displacement, and the rotation about the neutral axis. Thus, the order of $[K(\omega)]$ is 3, and at most there will be 3 eigenvalues below the first pole of $\det[K(\omega)]$. Supposing that there are 3 eigenvalues in this range, then between two poles there can be at most 1 eigenvalue. This latter conclusion, that poles must sandwich eigenvalues, is easily verified by considering the characteristic equations for the eigenvalues and poles. According to Refs. [40, 41], the eigenvalues of a simple cantilever beam are given by the roots of,

$$\cos(\beta\ell)\cosh(\beta\ell) = -1 \tag{a}$$

where, $\beta^4 = \omega^2\mu/(EI)$,

μ is the mass per unit length,

E is the modulus of elasticity,

I is the cross-section second area moment of inertia.

The poles of det[K(ω)] are given by the roots of,

$$\cos(\beta \ell)\cosh(\beta \ell) = 1 \qquad \text{(b)}$$

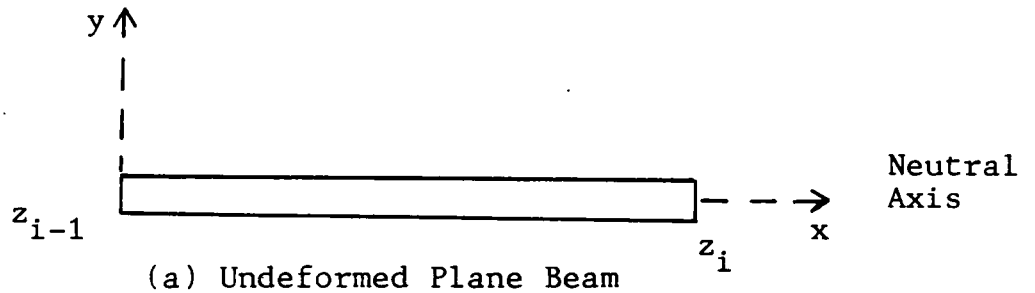
Since the cosh function is never less than 1, then between two roots of Eq. (a), eigenvalues, there must be a root of Eq. (b), a pole of det[K(ω)].

C. The Finite Element-Transfer Matrix Method:

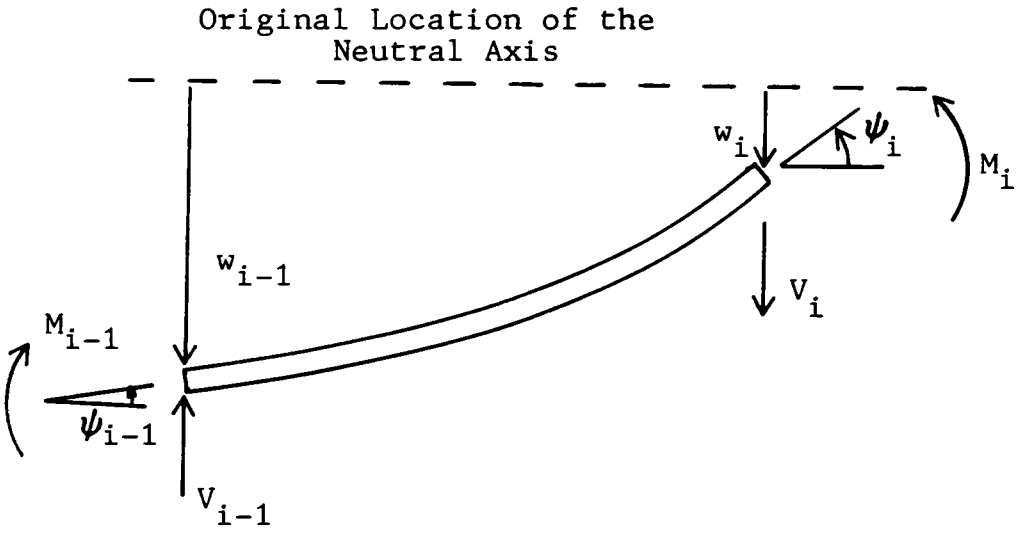
The finite element-transfer matrix (FETM) method is a hybrid method as its name implies. The method uses both the FEM and the Transfer Matrix Method (TMM). The FEM eigenvalue problem was presented above, and the TMM method is presented below.

The general theory of the TMM can be found in Ref. [42]. In the TMM certain variables of interest are defined at different stations in the system. For example, in structural analysis, the system may be a plane beam, as shown in Fig. 1. A beam usually has four variables of interest: deflection of the neutral axis, slope of the neutral axis, internal moment, and internal shear. Naturally, the stations would be the points along the neutral axis of the beam. These variables are placed in a vector which is called the state vector. The TMM then seeks to relate the state vectors at two stations by solving the boundary value problem between the two stations. This leads to the general equation,

$$\mathbf{z}_i = [TM]\ \mathbf{z}_{i-1} \qquad (11)$$



(a) Undeformed Plane Beam



(b) Deformed Plane Beam

Notation:

z is the state vector
w is the deflection of the neutral axis
ψ is the rotation about the translated neutral axis
M is the moment
V is the shear
i-1 and i are the left and right ends of the beam

Figure 1. Schematic and Notation for a Plane Beam in the TMM.

where, [TM] is the transfer matrix,

$\mathbf{z}_i$ is the state vector at station i, see Fig. 1,

$\mathbf{z}_{i-1}$ is the state vector at station i-1.

This equation is obviously simple to apply between stations. The difficulty in Eq. (11) is the derivation of the transfer matrix. However, through the years, extensive transfer matrix libraries have been developed, Refs. [42, 43].

A system normally contains many stations. To obtain the solution, Eq. (11) must be applied between all the stations, following a certain path. Thus, any solution using the TMM is inherently path dependent. If the path is a smooth, continuous path, along which the number of state variables is constant, then the TMM may be used with ease. However, if there are forks in the path, called "branching", or if the number of state variables changes along the path, then Eq. (11) becomes cumbersome to use. Pestel and Leckie [42] say that although these type of problems can be handled with modifications to Eq. (11), they recommend avoiding the TMM in favor of a more general analysis method, such as the FEM.

In the analysis of long chain-like structures, the FEM has been used extensively. A sample chain-like structure is shown in Fig. 2. As a chain-like structure gets longer, the number of degrees of freedom increases. This causes higher order matrices in a FEM model, creating more complex and expensive solutions [44, 45]. One way to reduce the size of the matrices is to couple the FEM and TMM techniques. This coupling leads to the FETM technique [44-52].

In the FETM a transfer matrix representing the periodic substruc-

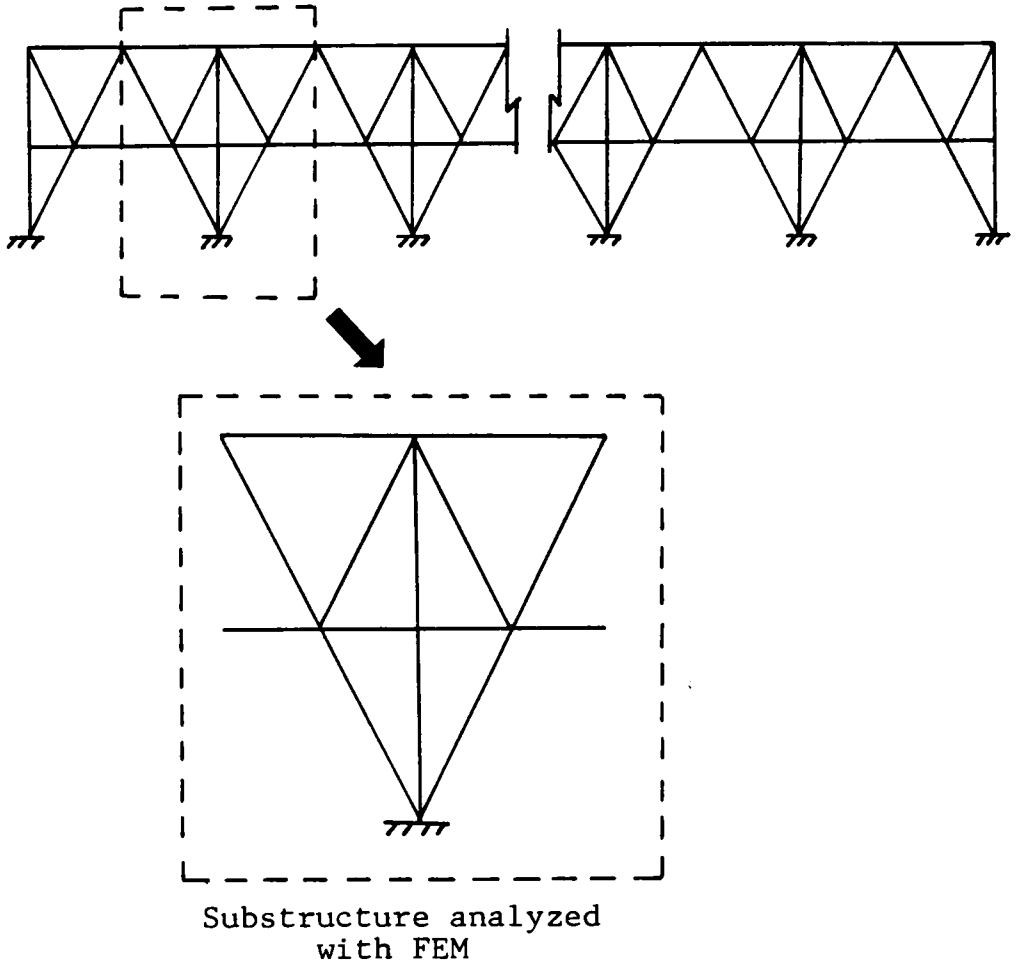


Figure 2. Typical Periodic Structure that May Be Analyzed with the FETM

ture is found with the aid of an FEM analysis. For example, the periodic substructure in Fig. 2 is the substructure in the square box. The FEM can then be used to develop the transfer matrix for this substructure. This transfer matrix is necessarily a lumped-parameter transfer matrix because of the discretization imposed by the FEM. Once the transfer matrix for the periodic structure is known, Eq. (11) is used to march from station to station. As the structure gets longer more stations are added.

The enhancement to the FEM studied here is the incorporation of an element matrix derived from a continuum transfer matrix. A continuum transfer matrix is obtained by solving the boundary value problems exactly between two stations [42]. Therefore, no discretization assumptions are made regarding the material property or mass distribution. This is in contrast to the lumped-parameter transfer matrices, which as their name implies, lump properties to facilitate the solution of the boundary value problem.

Of the cited work in the FETM area only Nagamatsu and Nagaike [52] considered coupling the two techniques in a dynamic stiffness formulation. The standard FEM eigenvalue problem may be written as $([K]-\omega^2[M])\mathbf{d} = \mathbf{0}$. The matrix in parentheses is commonly called the dynamic stiffness matrix. Nagamatsu and Nagaike termed this matrix the "impedance" matrix of the structure. In their work, they converted the transfer matrix of a beam with neither shear deformation nor rotatory inertia into an impedance formulation. They then coupled this to the master degrees of freedom of a finite element model, obtained with static substructuring. Next, they extracted the lower eigenvalues and

the associated mode shape for the master degrees of freedom. They reported good agreement between computed and experimental results for their technique, although not as good as for a pure TMM eigenvalue extraction.

The enhancement proposed here is similar to Nagamatsu and Nagaike's work because the enhancement is a transfer matrix conversion added to a FEM set of equations. But, the work here differs in the type of transfer matrix used, the purpose of the addition to the FEM, and the type of eigenvector extracted. The transfer matrix to be used here will include rotatory inertia and shear deformation. Nagamatsu and Nagaike's work sought the lower eigenvalues of a structure, whereas the purpose here is to improve the prediction of midrange and higher eigenvalues of a structure. The proposed method will not only find the master eigenvalues and eigenvectors but it will also reconstruct the eigenvector of every continuum element with any degree of precision required. This new technique will be presented in the theory section that follows.

THEORY

The FEM enhancement studied here is the addition of a continuous, exact element to the FEM. There are several ways to formulate continuous elements, as seen in the studies [33, 37, 57]. By definition, these elements satisfy the boundary value problem exactly and generate a transcendental, nonlinear eigenvalue problem. The eigenvalue extraction must, therefore, be carried out with different techniques than for lumped parameter or approximate models. The associated eigenvectors must also be obtained with different techniques [31]. Some of these techniques only approximate the eigenvectors, arguing that the eigenvector is no more than a visual aid.

For the fluid-structure linearization techniques that require accurate eigenvectors, approximations may prove unacceptable. Moreover, the entire ability of experimentally verifying theoretical FEM results depends on verifying the similarity of the experimental and theoretical mode shapes. The field of experimental modal analysis also depends upon accurate mode shapes to carry out structural modifications [54]. Mode shapes from experimental and theoretical approaches can be compared to help pin-point areas of weakness in the theoretical dynamic FEM model. For these reasons, it is considered necessary to obtain the total structure eigenvectors. The eigenvector extraction technique presented here uses the TMM to reconstruct the exact eigenvector description.

A. The Transfer Matrix Method:

The basic concept and definitions in the TMM which led to Eq. (11) were presented in the previous section. This equation summarized the TMM for all

fields of study which use this method. In structural analysis, the state vectors in Eq. (11) are usually made up of an even number of entries. Half of the entries are "displacements" and the remaining are the internal "forces" associated with the displacements. Thus, Eq. (11) may be rewritten as,

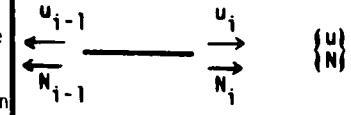
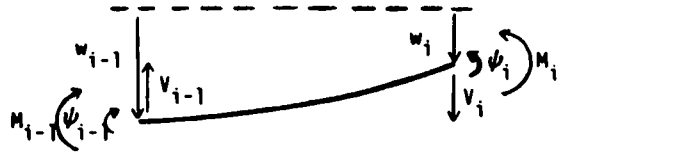
$$\begin{Bmatrix} \mathbf{d} \\ \mathbf{f} \end{Bmatrix}_i = [TM] \begin{Bmatrix} \mathbf{d} \\ \mathbf{f} \end{Bmatrix}_{i-1} \tag{12}$$

where **d** represents the displacements and **f** are the internal forces.

The transfer matrices for structural analysis are quite numerous [42, 43]. Many of these transfer matrices are exact solutions of the differential equations between two stations. These transfer matrices are referred to as "continuum transfer matrices", since the domain of the system between two stations is not discretized. The derivation of continuum transfer matrix for a Timoshenko beam with rotatory inertia is presented in Ref. [42] (section 5.1), and summarized in Table 1.

By definition, the interpolation functions in the FEM and the admissible functions of a Rayleigh-Ritz method are approximations to the solutions of a boundary value problem. That is, the set of solutions that may be obtained with the FEM or a variational method are only approximations of the solution obtained with continuum transfer matrices. Therefore, with a continuum transfer matrix and Eq. (12), once the state vector at one end of an exact element is known, then the displacements and internal loads along the element may be extracted with almost any degree of precision. This makes a continuum transfer matrix an ideal candidate for an exact element to be incorporated into a finite element code. The transformation of a typical catalog transfer matrix into a dynamical stiffness form follows. References will be made to Table 1 since this beam formulation will be used in later sections.

Table 1. Transfer Matrices for a Timoshenko Beam with Rotatory Inertia and Axial Deformation.

	Axial Vibration	Bending Vibration
Positive Sign Convention		
Transfer Matrix	$\cos\Omega$ $\quad \frac{\ell}{EA}\frac{\sin\Omega}{\Omega}$ $-\mu\omega^2\ell\frac{\sin\Omega}{\Omega}$ $\quad \cos\Omega$	$C_0-\sigma C_2$ $\quad \ell[C_1-(\sigma+\tau)C_3]$ $\quad aC_2$ $\quad \frac{a\ell}{\beta}[-\sigma C_1+(\beta^4+\sigma^2)C_3]$ $\frac{\beta^4}{\ell}C_3$ $\quad C_0-\tau C_2$ $\quad \frac{a}{\ell}(C_1-\tau C_3)$ $\quad aC_2$ $\frac{\beta^4}{a}C_2$ $\quad \frac{\ell}{a}[-\tau C_1+(\beta^4+\tau^2)C_3]$ $\quad C_0-\tau C_2$ $\quad \ell[C_1-(\sigma+\tau)C_3]$ $\frac{\beta^4}{a\ell}(C_1-\sigma C_3)$ $\quad \frac{\beta^4}{a}C_2$ $\quad \frac{\beta^4}{\ell}C_3$ $\quad C_0-\sigma C_2$
Variables used in Transfer Matrices	$\Omega=\ell\omega\sqrt{\frac{\rho}{E}}$ E=Modulus of Elasticity G=Modulus of Rigidity A=Cross section area A_s=Equivalent shear area I=Area moment of inertia i=Radius of gyration ℓ=Length of beam μ=Mass per unit length	$a=\frac{\ell^2}{EI}$ $\beta^4=\frac{\mu\omega^2}{EI}\ell^4$ $\sigma=\frac{\mu\omega^2}{GA_s}\ell^4$ $\tau=\frac{\mu i^2\omega^2}{EI}\ell^2$ $\lambda_{1,2}=\sqrt{\sqrt{\beta^4+\frac{1}{4}(\sigma-\tau)^2}\mp\frac{1}{2}(\sigma+\tau)}$
		$\Lambda=(\lambda_1^2+\lambda_2^2)^{-1}$ $C_0=\left(\lambda_2^2\cosh\lambda_1+\lambda_1^2\cos\lambda_2\right)\Lambda$ $C_2=(\cosh\lambda_1-\cos\lambda_2)\Lambda$ $C_3=\left(\frac{1}{\lambda_1}\sinh\lambda_1-\frac{1}{\lambda_2}\sin\lambda_2\right)\Lambda$ $C_1=\left(\frac{\lambda_2^2}{\lambda_1}\sinh\lambda_1+\frac{\lambda_1^2}{\lambda_2}\sin\lambda_2\right)\Lambda$

Partitioning the [TM] in Eq. (12) into,

$$[TM] = \begin{bmatrix} U_1 & U_2 \\ U_3 & U_4 \end{bmatrix}$$

Eq. (12) may be rewritten as,

$$\left\{\begin{matrix} \mathbf{d}_i \\ \hline \mathbf{f}_i \end{matrix}\right\} = \left[\begin{matrix} U_1 & U_2 \\ \hline U_3 & U_4 \end{matrix}\right] \left\{\begin{matrix} \mathbf{d}_{i-1} \\ \hline \mathbf{f}_{i-1} \end{matrix}\right\} \tag{13}$$

Solving the first set of equations for $\mathbf{f}_{i-1}$,

$$\mathbf{f}_{i-1} = U_2^{-1}\mathbf{d}_i - U_2^{-1}U_1\mathbf{d}_{i-1} \tag{14}$$

and substituting into the second set of equation to solve for f_i,

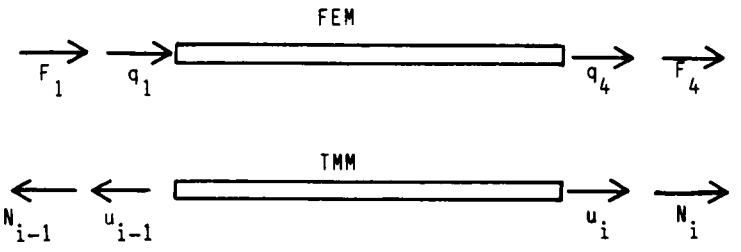
$$\mathbf{f}_i = (U_3 - U_4U_2^{-1}U_1)\mathbf{d}_{i-1} + U_4U_2^{-1}\mathbf{d}_i \tag{15}$$

Eqs. (14) and (15) can be assembled into:

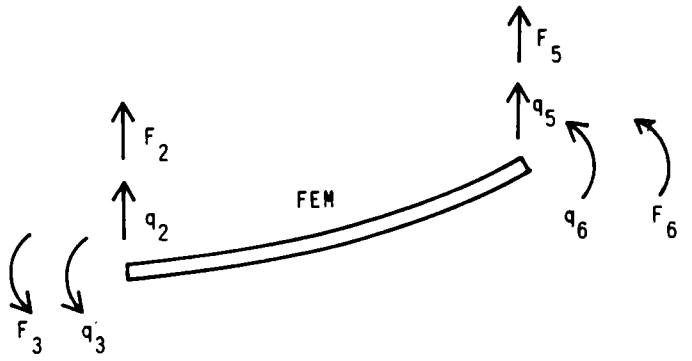
$$\left[\begin{matrix} -U_2^{-1}U_1 & U_2^{-1} \\ \hline U_3-U_4U_2^{-1}U_1 & U_4U_2^{-1} \end{matrix}\right] \left\{\begin{matrix} \mathbf{d}_{i-1} \\ \hline \mathbf{d}_i \end{matrix}\right\} = \left\{\begin{matrix} \mathbf{f}_{i-1} \\ \hline \mathbf{f}_i \end{matrix}\right\} \tag{16}$$

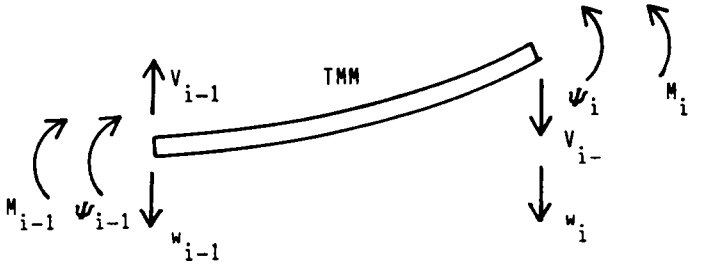
Equation (16) resembles the classical static stiffness formulation encountered in static FEM problems. Equation (16) must be incorporated into a finite element code with care since the sign conventions between the TMM and the FEM generally do not agree. For the beam element studied here, the differences in sign convention are summarized in Fig. 3. In this figure q_i is the i^{th} degree of freedom in the local coordinate system of a finite element beam element. The beam transfer matrix presented in Table 1 was converted into the stiffness matrix by using Eq. (16). The resulting stiffness matrix was algebraically complex to evaluate because of the many trigonometric and hyperbolic terms. The number of operations necessary to evaluate the converted stiffness matrix was greater than those required to evaluate and convert it via Eq. (16). For this reason the latter approach was implemented.

In Eq. (16), all four submatrices contain the submatrix U_2^{-1}. Should the matrix U_2 be singular then every term in the matrix in Eq. (16) will be



(a) Axial Conventions





(b) Bending Conventions

Figure 3. Positive Sign Conventions in the FEM and the TMM.

indeterminate because $\det(U_2)=0$ and $U_2^{-1}=[\mathrm{Cof}(U_2)^T]/\det(U_2)$, $U_2 \neq \mathbf{0}$. It is convenient at this point to ask when will the submatrix U_2 become singular? Perhaps the easiest way to answer this question is by performing a free-vibration analysis of a structure described by Eq. (16). Suppose that a structure has fixed-fixed boundary conditions (i.e. $\mathbf{d}_{i-1}=\mathbf{d}_i=\mathbf{0}$), and suppose that the structure is vibrating harmonically, then from Eq. (13),

$$\begin{Bmatrix} \mathbf{0} \\ - - \\ \mathbf{f}_i \end{Bmatrix} = \begin{bmatrix} U_1 | U_2 \\ - + - \\ U_3 | U_4 \end{bmatrix} \begin{Bmatrix} \mathbf{0} \\ - - \\ \mathbf{f}_{i-1} \end{Bmatrix} \tag{17}$$

To avoid the trivial solution of Eq. (17), $\mathbf{0}=[TM]\mathbf{0}$, then $U_2\,\mathbf{f}_{i-1}=\mathbf{0}$, without $\mathbf{f}_{i-1}=\mathbf{0}$. This can only be satisfied if U_2 is a singular matrix, that is, $\det(U_2)=0$. Since all the terms of U_2 for the beam studied here are functions of frequency, $\det(U_2)$ is a continuous function of frequency. Setting this function equal to zero gives the characteristic equation, $\det(U_2)=0$, the roots of this equation are the natural frequencies of the fixed-fixed beam problem. In other words, the terms of the dynamical stiffness matrix derived from a transfer matrix will become undefined whenever the circular frequencies selected are evaluated at the natural frequencies of the exact elements with fixed-fixed boundary conditions.

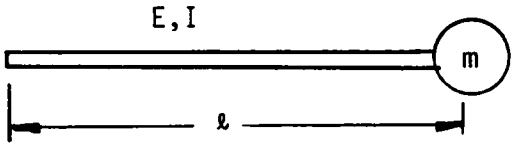
This result was expected based on the discussion of poles of a dynamical matrix, given in the previous section. In a frequency-search eigenvalue extraction, the determinant of a dynamical stiffness matrix will "blow up" as the trial frequency approaches a pole. Poles were shown to occur at the natural frequencies of each of the structure exact elements with fixed-fixed boundary conditions. At each pole, the U_2 matrix of the associated exact element becomes singular. This introduces terms into the structure dynamic matrix that tend to infinity. These terms that tend to infinity cause the

interminant of the dynamical stiffness matrix to become undefined. Obviously, it is necessary to avoid evaluating the determinant of the structure matrix at a pole.

It is simple to compute the poles before performing a frequency-search eigenvalue extraction. Blevins [40] has tabulated the natural frequencies of fixed-fixed beams. These published eigenvalues do not include the effects of shear deformation nor rotatory inertia. Nevertheless, these eigenvalues can be used as upper bounds for the eigenvalues of a beam element which includes both of these effects. Once a pole is located, it can be determined whether the determinant of the structure dynamic matrix will change signs as a pole is crossed. If there is an even number of like exact elements then the determinant will not change sign, whereas, if there is a odd number of identical exact elements then the determinant will change sign.

In a frequency-search, eigenvalue extraction of a lumped-parameter model, there are no poles. As an illustration of this a lumped-parameter transfer matrix for a beam is shown in Fig. 4. It is easily verified that U_2 for this transfer matrix does not contain frequency dependent terms. That is, $\det(U_2)=\text{constant}=-\ell^4/[12(EI)^2]$. U_2 can be singular only if $\ell=0$ or if the product EI is equal to infinity. Both of these conditions are outside the realm of practical structural analysis.

The dynamical stiffness matrix transformed from a transfer matrix was incorporated into a finite element code. A finite element code was written into which the above exact element could be incorporated. The FEM beam element used in this code follows, while the actual implementation of the code is presented in the next section.



$$[TM] = \left[\begin{array}{cc|cc} 1 & -\ell & -\frac{\ell^2}{2EI} & -\frac{\ell^3}{6EI} \\ 0 & 1 & \frac{\ell}{EI} & \frac{\ell^2}{2EI} \\ \hline 0 & 0 & 1 & \ell \\ -m\omega^2 & m\ell\omega^2 & \frac{m\ell^2\omega^2}{2EI} & \frac{m\ell^3\omega^2}{6EI} + 1 \end{array}\right]$$

Figure 4. Massless Elastic Beam With Lumped Mass Transfer Matrix.

B. The FEM Beam Element:

Of all the structural analysis elements in use, the beam is one of the simplest to develop [10, 11], because energy formulations are not required. The finite element presented here is assumed to have uncoupled bending and axial deformations. The development of the element matrices in local coordinates corresponding to the local displacement vector $\{q_1\ q_2\ q_3\ q_4\ q_5\ q_6\}^T$ (see Fig. 3) is found in Ref. [11]. The element stiffness matrix in local coordinates for a linear, plane beam element is,

$$[K_{element}] = \frac{EI}{\ell^3}\begin{bmatrix} A\ell^2/I & 0 & 0 & -A\ell^2/I & 0 & 0 \\ 0 & 12 & 6\ell & 0 & -12 & 6\ell \\ 0 & 6\ell & 4\ell^2 & 0 & -6\ell & 2\ell^2 \\ -A\ell^2/I & 0 & 0 & A\ell^2/I & 0 & 0 \\ 0 & -12 & -6\ell & 0 & 12 & -6\ell \\ 0 & 6\ell & 2\ell^2 & 0 & -6\ell & 4\ell^2 \end{bmatrix} \tag{18}$$

where, E is the modulus of elasticity,

ℓ is the length of the beam element, and

I is the area moment of inertia.

The mass matrix in local coordinates is,

$$[M_{element}] = (A\rho)\begin{bmatrix} \ell/3 & 0 & 0 & \ell/6 & 0 & 0 \\ 0 & 13/35\,\ell & 11/210\,\ell^2 & 0 & 9/70\,\ell & -13/420\,\ell^2 \\ 0 & 11/210\,\ell^2 & 1/105\,\ell^3 & 0 & 13/420\,\ell^2 & -1/140\,\ell^3 \\ \ell/6 & 0 & 0 & \ell/3 & 0 & 0 \\ 0 & 9/70\,\ell & 13/420\,\ell^2 & 0 & 13/35\,\ell & -11/210\,\ell^2 \\ 0 & -13/420\,\ell^2 & -1/140\,\ell^3 & 0 & -11/210\,\ell^2 & 1/105\,\ell^3 \end{bmatrix} \tag{19}$$

where, A is the cross section area and ρ is the mass density of the material.

The transformation from local into global coordinates is accomplished by the use of coordinate transformation matrix [g],

$$[g] = \begin{bmatrix} \cos(a) & \sin(a) & 0 & 0 & 0 & 0 \\ -\sin(a) & \cos(a) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos(a) & \sin(a) & 0 \\ 0 & 0 & 0 & -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

where, a is the orientation angle of the beam before any deformation has occurred. Then with Eq. (20) the element stiffness and mass matrices in global coordinates can be defined,

$$[K^G_{element}] = [g]^T[K_{element}][g], \tag{21}$$

$$[M^G_{element}] = [g]^T[M_{element}][g]. \tag{22}$$

The global mass and stiffness matrices described by Eqs. (21) and (22) were incorporated into a program along with the exact beam element formulation. The development of the program is presented in the next section. What follows in this section outlines the eigenvalue extraction technique implemented into the computer code.

C. The Incremental-Falsi-Bisection Method:

The Incremental, Falsi, and Bisection methods are basic root finding methods that can be found in most numerical analysis textbooks such, as [55]. All three are used in the eigenvalue extraction. First, the Incremental method is used to locate a frequency interval containing a root. This search used a frequency step supplied by the user of the program. Having found such an interval, the Falsi technique is used to home in on the root, to a desired tolerance. The Falsi technique is a first order polynomial interpolation of the determinant. Once the polynomial is found, the root of the polynomial is

used as an approximation to the zero of the determinant. In certain rare circumstances the Falsi technique has computed new approximations to the root which lie outside the interval containing the root [56]. When this occurs, the Falsi technique usually misses the root in the interval. To avoid this, the Falsi iteration implemented checks every time a new approximation is calculated to determine whether or not it lies inside the interval. If the new approximation lies outside the interval, the approximation is discarded and the extraction proceeds using the Bisection method.

A more thorough explanation of details related to the eigenvalues extraction is presented in the Description of Program DSTAP section. It is more appropriate to present the eigenvalue extraction technique in that section since many of the compromises and choices made were directly related to how the extraction was to be implemented.

DESCRIPTION OF PROGRAM DSTAP

A dynamic structural analysis program (DSTAP) was developed to study the two elements presented in the Theory section. This program was the result of extensive modifications and additions to the Structural Analysis Program (STAP). Program STAP is a small, instructional-type, finite element program that can be found in Ref. [9]. It was implemented on an IBM 370 at Virginia Polytechnic Institute and State University, by personnel of the Engineering Science and Mechanics Department [58]. This implementation was moved to a VAX cluster, running VMS 4.1, before the two elements were added. The VAX cluster consisted of two 11/780 processors with 8 megabytes of memory. Off-line storage was provided by RP07 and RA81 disks.

As presented in Ref. [9], program STAP is written in FORTRAN 66. It has a one element library, a truss element, and is set up to perform static analysis. The procedure for solving static problems may be seen in the flowchart presented in Fig. 5, taken from Ref. [9]. A detailed explanation of each function block in the flowchart may be found in Ref. [9], and will not repeated here.

In STAP, the set of equations arising from static problems is solved using a nodal front solver and Gauss elimination. The nodal front solver retains the same numbering scheme for the degrees of freedom as supplied by the user. That is, the numbering of the system of unknowns is not optimized for an in-core solution, as is done in some commercial codes. This same numbering scheme for the unknown degrees of freedom was retained in DSTAP. Program STAP also uses an efficient matrix storage scheme in which only the nonzero terms of the global matrices lying on or above the main diagonal are

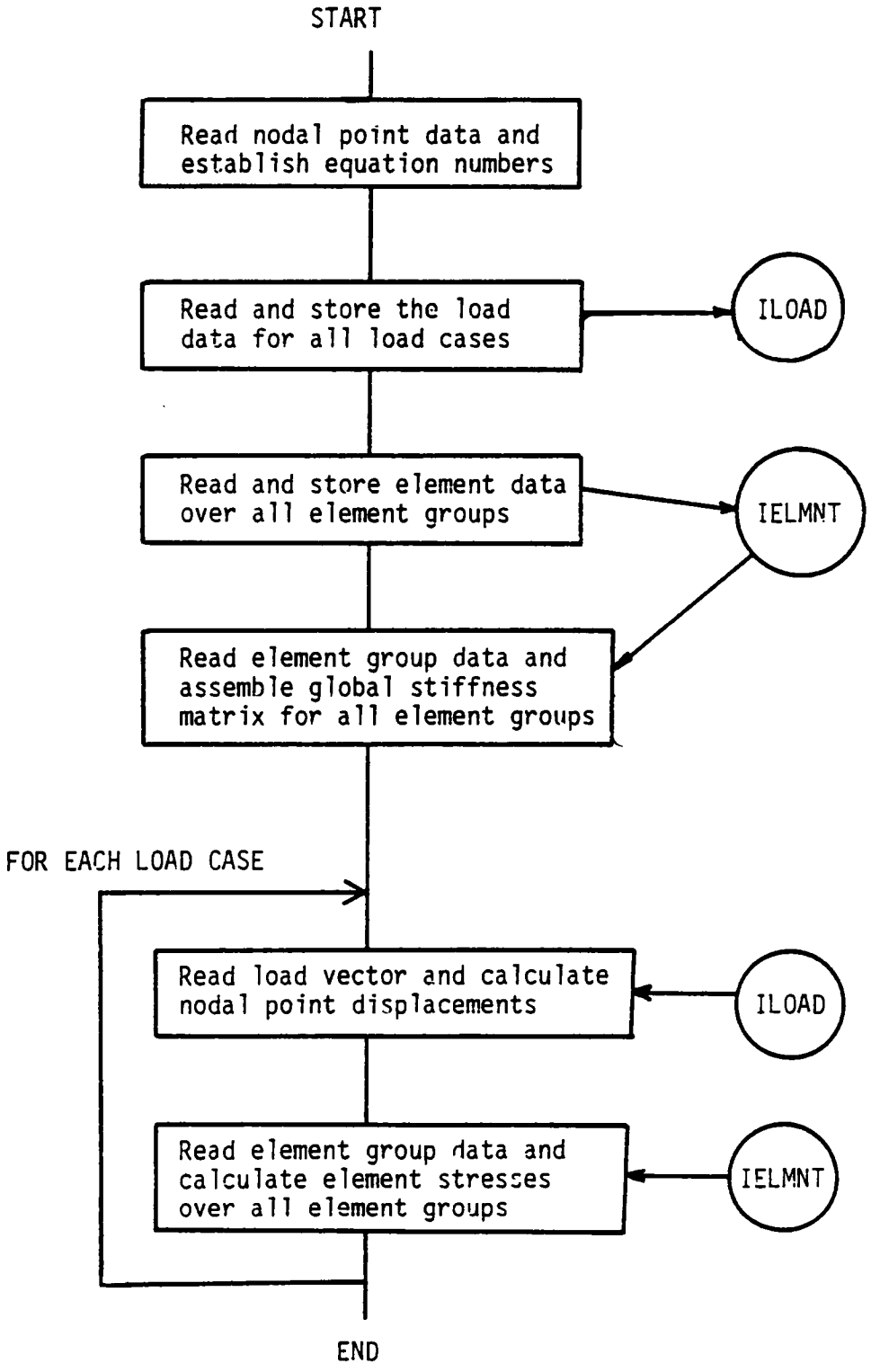


Figure 5. Flowchart of program STAP and Static Portion of Program DSTAP.

actually stored. This same storage scheme was adapted for program DSTAP. This technique for storing model matrices is oftentimes referred to as "Skyline" storage.

The last feature of STAP incorporated into DSTAP is, perhaps, the prime cause of frustration among novice programmers. This feature deals with the way in which model data is stored in-core. All array data needed in an analysis is stored in one large, one-dimensional array called the A-array. Since several arrays and matrices are stored in the A-array at once, pointers are set up to locate initial word of each array or matrix. This partitioning scheme is known as "dynamic storage". There are two advantages to using dynamic storage. First, the code can be easily modified for different size problems by changing the dimension of the A-array in the main program. Second, in-core memory can be efficiently reused as arrays are no longer needed by changing the pointers to the A-array. A set of tables explaining the allocation of pointers to the A-array during different phases of the program DSTAP are attached in Appendix A.

The flowchart of program DSTAP is presented in Fig. 6. Although it is not shown in Fig. 6, program DSTAP retains the static analysis capabilities of STAP. In the case of a static analysis program DSTAP functions in the same manner as program STAP (Fig. 5). In the flowchart of program DSTAP (Fig. 6) a subroutine name appears next to each function block. This indicates that a major portion of the operations described in said block are done in that subroutine. For the interested reader, the subroutines named in the flowchart are included in Appendix B (except subroutine INPUT which is in Ref. [9]), along with other supporting subroutines. The set of subroutines in Appendix B constitute the additions and modifications to program STAP which

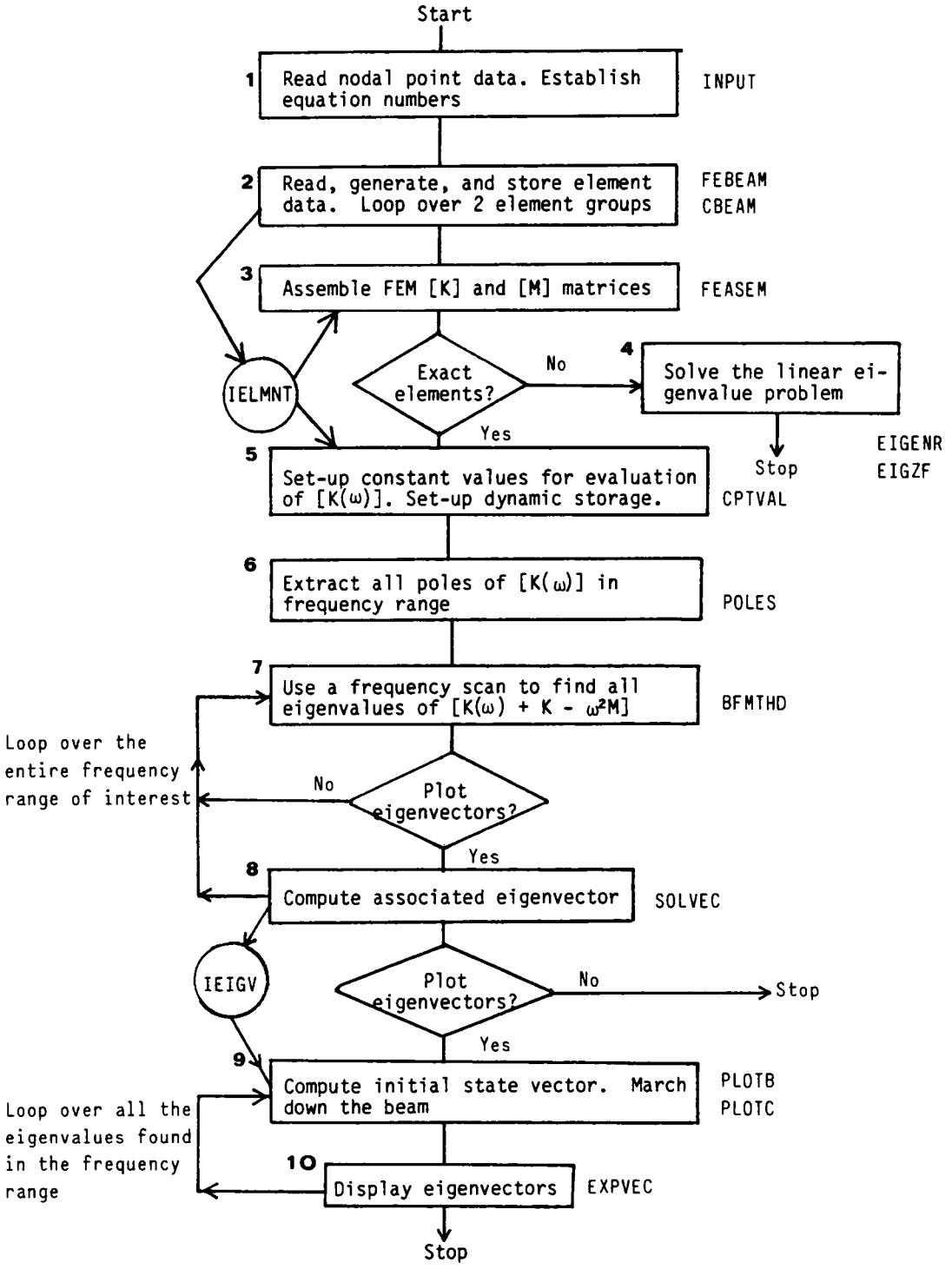


Figure 6. Flowchart of program DSTAP.

were necessary to implement the two elements described in the Theory section and the eigenvalue extraction outulined there. Program DSTAP consists of these subroutines and subroutines from program STAP [9], the PLOT10 library [59], and the IMSL library [60].

The remainder of this section will describe in detail the function blocks in the flowchart of program DSTAP, Fig. 6. These have been numbered to facilitate any references to a particular block. The number assigned to each block appears to the left of the block in bold numbers, while the names of the associated subroutines appear to the right of the block. The reader may wish to refer to the particular subroutine(s) in Appendix B, although this will not be necessary.

The first block in Fig. 6 describes the process of inputting the model nodal points. This task is performed in subroutine INPUT which is taken directly from program STAP; thus, it is not included in Appendix B. The nodal data consists of the coordinates and degrees of freedom of each node point. Provisions are included for generating nodes automatically along straight lines. This nodal generation technique is described in Ref. [9].

The second block in Fig. 6 summarizes the element generation process. The exact beam element information is read in through subroutine CBEAM. The pointers needed for the arrays in subroutine CBEAM are computed in subroutine TBEAM. Thus, the sole purpose of CBEAM is place the user supplied information into the storage locations allotted by TBEAM. Some automatic element generation capabilities are supplied in CBEAM, as described in Appendix C which contains the instruction for using program DSTAP. The FEM beam element formulation was placed in subroutine FEBEAM, and the computation of the pointers for dynamic storage were also placed in subroutine TBEAM.

Unlike CBEAM, FEBEAM is called twice, (1) to read in the element data, (2) to generate the FEM beam element stiffness and consistent mass matrices. These element matrices are assembled into the structure FEM matrices using the subroutine FEASEM, as shown in the third function block of Fig. 6.

A model made up of only finite elements gives a linear eigenvalue problem. There are many algorithms available for solving this problem. As shown in the fourth function block of Fig. 6, two algorithms are provided in DSTAP, one in subroutine EIGENR and another in subroutine EIGZF. The modified QR algorithm in EIGENR uses the global matrix properties of the finite element formulation to optimize the solution time; that is, it requires real, symmetric, positive definite matrices. This modified QR algorithm was developed by a graduate student of the Aerospace and Ocean Engineering Department at Virginia Polytechnic Institute and State University [61]. Unfortunately, no documentation is available for this routine.

In certain cases the FEM matrices become non-positive definite due to the finite arithmetic performed by the computer. In these cases it is convenient to have an extraction routine that does not require the matrices to be positive definite. This is why EIGZF was also added to STAP. EIGZF is a routine available in the IMSL library of subroutines [60]. EIGZF only requires that the matrices be real.

Whenever an exact beam element is incorporated into a structure model the resultant eigenvalue problem is a transcendental eigenvalue problem. In this case a frequency-scan is performed on the structure dynamic stiffness matrix as was outlined in the Theory section. This search is embodied in the first loop of Fig. 6. The kernel of this search resides in subroutine BFMTHD

which contains a rather crude algorithm that combines the Incremental Search, Regula-Falsi, and Bisection algorithms. The Incremental Search seeks a zero crossing of the determinant of the dynamic structure matrix,

$$\det([K(\omega)]+[K]-\omega^2[M])=0, \tag{23}$$

where, $[K(\omega)]$ is the dynamic stiffness matrix assembled from beam exact elements,
$[K]$ is the global stiffness matrix assembled from beam finite elements, and
$[M]$ is the assembled mass matrix obtained from the consistent beam finite element mass matrices.

Once a zero crossing has been detected, the Regula-Falsi algorithm, in subroutine FALSI, is invoked. Should the new approximation to a root in subroutine FALSI lie outside the interval containing the root, then FALSI calls subroutine HALF, which contains a Bisection algorithm. This rare occurrence was observed once during the development of program DSTAP. However, not enough information was saved about that model to be able to reproduce the results.

It should be evident from the discussion above that Eq. (23) will be evaluated at many different trial frequencies. If possible the trial frequencies should avoid the poles of $\det[K(\omega)]$ because this may cause an overflow condition. Thus, before subroutine BFMTHD is actually called, the poles of $\det[K(\omega)]$ are found and the dynamic memory is rearranged for an effective evaluation of $[K(\omega)]$. This is done in subroutines CPTVAL and POLES, as shown in the 5^{th} and 6^{th} function blocks of Fig. 6. CPTVAL first computes all the values independent of ω needed to evaluate $[K(\omega)]$. These constants are stored in dynamic memory, rather than off-line, to speed up the evaluation of $[K(\omega)]$. Prior to rearranging the A-array, CPTVAL calls subroutine POLES to compute the poles of $\det[K(\omega)]$. These poles are also stored in-core, as will be explained below.

In subroutine POLES, the 6th function block in Fig.6, the poles of det[K(ω)] are found by approximating them with the eigenvalues of an exact, fixed-fixed beam element, neglecting shear deformation and rotatory inertia. The approximating eigenvalues have been tabulated by Ref. [40]. Since the approximating eigenvalues correspond to a more rigid element, these eigenvalues are upper bounds to the poles of the exact beam element with both shear deformation and rotatory inertia. The lower bound for each pole can either be searched for, by backing off the upper bound, or by arbitrarily choosing a known lower bound, such as a previous pole. It was found more convenient to search for a pole by backing off the upper bound. Once the upper and lower bounds have been established, subroutine FALSI finds the pole with an accuracy of 0.1 rad/sec. The poles are extracted up to and including the first pole beyond the upper frequency specified by the user. This assures that the last pole in the frequency range of interest will not be missed, regardless of how close it is to the upper frequency.

Having computed and stored the poles in the frequency range of interest, subroutine CPTVAL rearranges the A-array storage in preparation for BFMTHD, the 7th block in Fig. 6. With the structure poles stored in-core, it is simple to determine whether a sign change of Eq. (23) is indicative of an eigenvalue or not. Let $f(\omega)=\det([K(\omega)]+[K]-\omega^2[M])$, so that an eigenvalue ω makes $f(\omega)=0$. Suppose that the last evaluation of $f(\omega)$ occurred at frequency ω_a, and that the current frequency being tested is ω_b. If there are an <u>odd</u> number of poles between ω_a and ω_b then the Wittrick-Williams theorem establishes that the $f(\omega)$ will change signs. This means that if $f(\omega_a)*f(\omega_b) > 0$ then an eigenvalue has been trapped, whereas if $f(\omega_a)*f(\omega_b) < 0$ then there is no eigenvalue in the interval $[\omega_a,\omega_b]$. This may seem contrary to the

typical criteria to trap roots of functions. However, it is easy to recast the bound criteria in a more traditional fashion by defining $g(\omega)=-1^{p}f(\omega)$, where p is the number of poles lying between ω_b and the low end of the frequency range specified by the user. For this reason, p is not necessarily equal to $J_o(\omega)$, since $J_o(\omega)$ is the number of all the poles below ω. This definition of $g(\omega)$ has the advantage that whenever $g(\omega_a)*g(\omega_b) < 0$, then an eigenvalue has been trapped. This definition of $g(\omega)$ was placed in function FCN, which is used by subroutines BFMTHD, FALSI, and HALF to extract the structure eigenvalues.

An additional feature added to the function FCN is a logarithmic compression of $g(\omega)$. Usually the terms along the main diagonal of structure stiffness matrices are quite large in magnitude. For the numerical examples to be presented in the next section, these terms were in the order of 10^5 (lb_f/in). The highest possible order of the determinant of a matrix may be estimated by raising the largest term in the matrix to the order of the matrix. Thus, for a 3 by 3 matrix with terms in the order of 10^5 the determinant could be in the order of 10^{15}. The double precision representation of real variables in a VAX 11/780 allows for number of up to 10^{36}. Therefore, if the the matrix in the example was an 8 by 8 matrix the determinant could be in the order of 10^{40}, exceeding the capacity of the VAX representation and causing an overflow error. For this reason, instead of returning $g(\omega)=-1^{p}f(\omega)$ function FCN actually returns,

$$g(\omega)= \begin{cases} -1^{p}f(\omega), & \text{if } |f(\omega)| < e \\ -1^{p}\text{sgn}(f(\omega))\ln(|f(\omega)|), & \text{if } |f(\omega)| > e \end{cases}$$

where the sgn function is 1 if $f(\omega)$ positive or -1 if $f(\omega)$ is negative, the ln function is the Napierian logarithm, and e is the base of the Napierian

logarithms. It is obvious that the function returned by FCN will have a slight discontinuity whenever $f(\omega) = \pm e$. Although it is unlikely, if this discontinuity should cause any difficulties for subroutine FALSI, then the Bisection algorithm in subroutine HALF would be called.

Once an eigenvalue has been found with the desired accuracy, the eigenvector is backed out using subroutine SOLVEC, the 8^{th} function block in Fig. 6. SOLVEC uses an full pivoting technique to find the eigenvector **x**. In full pivoting, the last diagonal term will be the term with the smallest magnitude. If there are n equations in the eigenvalue problem, then after upper triangularizing $\mathbf{A}=([K(\omega)]+[K] - \omega^2[M])$ at an eigenvalue ω with full pivoting, $a_{n,n}=0$. Therefore, to find the eigenvector **x**, the last entry in **x** is arbitrarily set to 1. The remaining (n-1) entries of **x** are found by backsolving the system in the upper-triangular matrix **A**. Once **x** is known, it is normalized so the largest entry in **x** is equal to 1, and the resulting vector is written to file IEIGV for post-processing.

The procedure described above for BFMTHD and SOLVEC is repeated for each eigenvalue found in the frequency range of interest, as shown in Fig. 6. Once the entire frequency range has been scanned, tape IEIGV will contain all the mode shapes and eigenvalues in the frequency range. These mode shapes will only contain the displacements at the juncture points of the elements. For example, if a cantilever beam is modeled with one exact beam element, and n mode shapes are extracted, it would be rather difficult to see any evident difference from mode shape to mode shape. The traditional vibration nodes would be missing from these mode shapes because the known displacements would be at the fixed-end (zero) and at the free end. As presented in the Literature Review section several studies in the exact stiffness field have

attempted to reconstruct the actual mode shapes by approximations of different types. These approximations are usually argued to be sufficient based on the premise that mode shapes are no more than a visual aid for the analyst [31]. However, recent work refutes this assumption and, furthermore, this work claims that it is not necessary to approximate the mode shapes at all, as explained below.

In the post-processing phase of program DSTAP is comprised of the last two function blocks of Fig. 6. The purpose of the part of the program is plot the structure eigenvectors on the terminal screen. This section of the program is optional and the user must turn on the plotting flag for this section to execute. The plotting flag is described in Appendix C. Prior to displaying the mode shape on the screen, DSTAP must compute the eigenvector of each continuous element, since only the end displacements are known. This is accomplished by resorting to Eqs. (11) through (14). In Eq. (11) the transfer matrix technique was outlined as,

$$\mathbf{z}_i = [TM]\mathbf{z}_{i-1} \tag{24}$$

where, $\mathbf{z}_{i-1} = \{\mathbf{d}_{i-1}\ \mathbf{f}_{i-1}\}^T$. If the initial state vector at one end of the continuous beam element is obtained, then it is possible to define a transfer matrix over a fraction of the beam length and use it to compute state vectors along the continuous element. It is also possible to define a transfer matrix over the entire length of the beam element such that,

$$\begin{Bmatrix} \mathbf{d}_J \\ \mathbf{f}_J \end{Bmatrix} = \begin{bmatrix} U_1 & U_2 \\ U_3 & U_4 \end{bmatrix} \begin{Bmatrix} \mathbf{d}_I \\ \mathbf{f}_I \end{Bmatrix} \tag{25}$$

where, the subscript I refers to the I^{th} end of the beam, and the subscript J refers to the J^{th} end of the beam. Both $\mathbf{d}_I$ and $\mathbf{d}_J$ are known from the eigenvector $\mathbf{x}$, which was stored on tape. With these displacements the

initial state vector $\mathbf{z}_I$ may be found with the use of Eq. (14),

$$\mathbf{f}_I = U_2^{-1}(\mathbf{d}_J - U_1\mathbf{d}_I) \tag{26}$$

Note, U_2 in Eq. (26) is nonsingular, because if it were singular then the frequency at which the eigenvector $\mathbf{x}$ was computed would be a pole of $\det[K(\omega)]$. This is not possible in program DSTAP; as frequencies that correspond to poles are ignored in the eigen-extraction process. Suppose that a structure eigenvalue is also pole. Although it is possible that $g(\omega)$ will be bounded at such an eigenvalue, it is unlikely that $g(\omega)=0$. Thus, $\mathbf{x}=\mathbf{O}$ at such at an eigenvalue, and $\mathbf{f}_I=\mathbf{O}$, which would make the eigenvector reconstruction impossible using Eq. 24. Therefore, the eigenvector reconstruction is performed for only those structure eigenvalues which are not poles of $\det[K(\omega)]$. With the initial state vector known, subroutine PLOTB loops over all continuous elements to obtain the state vectors along each one. This data is then scaled so the largest eigenvector entry is ten percent of the largest structure dimension. The scaled vector is finally plotted on the terminal screen for the user to see.

The evaluation of the continuous element eigenvectors at frequencies above 12,000 rad/sec incurred severe round-off error when double precision variables were used. At these relatively high structure frequencies the computed state vector displacements at the J^{th} end of the beam were not close to the $\mathbf{d}_J$ from eigenvector $\mathbf{x}$. In order to reduce truncation error, the computation described in Eq. (24) was performed in REAL*16, commonly known as quad-precision. The added precision in the REAL*16 operations permitted the extraction of all the eigenvectors attempted. This included the extraction of eigenvectors up to the 50,000 rad/sec region.

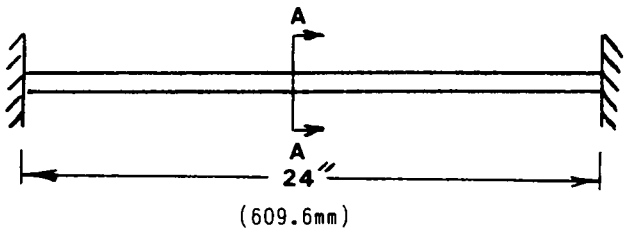
NUMERICAL EXAMPLES

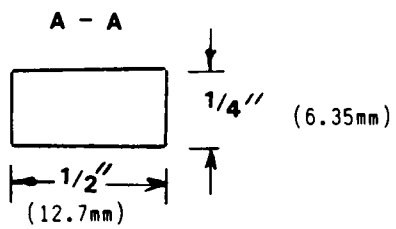
In this section, three different structures will be analyzed with the proposed enhancement to the finite element eigenvalue problem. The first example is a fixed-fixed beam, which will be used to benchmark both of the elements added to DSTAP against known solutions. After establishing the accuracy of these elements, the second example compares the FEM beam element against the exact beam element, by analyzing a Portal Arch in DSTAP. Finally, in the third example, another structure is examined to test the conclusions arrived at in the second example and to compare the results against a commercial FEM code.

A. Fixed-Fixed Beam:

The first example presented here is the analysis of a beam with fixed-fixed boundary conditions as shown in Fig. 7. This is a rather simple eigenvalue problem which should make the comparisons among the different models clearer. The first comparison to be presented is between the eigenvalues of a closed-form solution and those of a model using the so called exact elements. The second comparison is between the eigenvalues extracted from a finite element model in DSTAP and those from a finite element model in SUPERB [15].

The closed-form solution for a Timoshenko beam with no shear deformation is widely available in the literature [40, 41]. In this solution, the axial vibration is usually uncoupled from the transverse vibration. Hence, the axial and transverse eigenvalues are found independently. The exact displacement formulation, which led to the exact element, also uncouples





Material Properties:

E=30.0Mpsi (207GPa)

ρ=0.282 lb_f/in^3 (7.81 mgr/mm^3)

G= 11.6 Mpsi (79.9 GPa)

Figure 7. Schematic of the Beam for Example A.

these two types of vibration. To determine the accuracy of the exact beam element it is appropriate to compare the closed-form solution eigenvalues with those from an exact beam element model.

The closed-form solution to the transverse vibration of a beam with fixed-fixed end conditions may be found in [40, 41]. The associated characteristic equation is,

$$1 - \mathrm{Cosh}(\beta_i)\mathrm{Cos}(\beta_i) = 0 \tag{27}$$

where, $\beta_i^4 = \mu\,\omega_i^2 \ell^4/(EI)$, (28)

and, μ is the mass per unit length,

ℓ is the length of the beam,

E is the modulus of elasticity,

I is the moment of inertia, and

ω_i is the i^{th} eigenvalue.

The values of the above constants for this example appear in Fig. 7.

As the solution of Eq. (28) for the eigenvalues $_i$ is difficult to accomplish by hand calculations, the first twenty positive, nonzero roots of Eq. (27) were found numerically. A small program using a combination Incremental-Bisection method was written to extract these roots employing the quad-precision capability of VAX BASIC. This allows a mantissa of about 33 digits throughout all operations. The roots obtained with this program are presented in Table 2, along with the residuals of the computation. A residual is defined by letting $f(\beta)=1-\mathrm{Cosh}(\beta)\mathrm{Cos}(\beta)$, then at a computed root β_i the residual is $f(\beta_i)$, which usually is nonzero. The entries in this table have been included with all the digits available from the VAX BASIC program.

With the first twenty roots of Eq. (27) known, it is simple to compute the first twenty bending eigenvalues with Eq. (28). In addition to the bending

Table 2. The First 20 Roots of Eq. (27) Obtained in Quad-Precision With a VAX BASIC Program.

Root No.	Root	Residual
1	4.73004074486379977315664291381 83	-.6316491377664798064E-0010
2	7.85320462409583797125378623604 77	5.3339650106764214789E-0013
3	10.99560783800166774426543270237 74	9.4261355268224589479E-0011
4	14.13716549125746424486749219795 46	4.6735385022183045213E-0011
5	17.27875965739948144195281543034 07	-.6163440376432062314E-0010
6	20.42035224562606109078785610044 80	-.54866067749585744 72E-0010
7	23.56194490338636736773698454513 15	-.1150295493682681283E+0002
8	26.70353755697108200533582415559 49	4.8904674011102316517E-0011
9	29.84513021056595384466227124590 55	1.0000000000064232100E+0000
10	32.98672286415571531845598251808 06	-.6202721470963317770E-0010
11	36.12831551774551840726840770705 66	-.2682509567109112007E-0008
12	39.26990817133531122005884011563 28	-.6980363534164094482E-0007
13	42.41150082492510447691644793711 44	-.9954152765342742558E-0006
14	45.55309347851489771458417361397 00	4.6382038809404356563E-0005
15	48.69468613210469095308116770724 48	-.4886073873088245472E-0003
16	51.83627878569448419154232668202 26	-.1468249583513834485E-0001
17	54.97787143928427743000503413890 70	3.6833796870830283231E-0001
18	58.11946409287407066846767542134 25	-.2677648280154856920E+0001
19	61.26105674646386390693031835813 92	-.1573031435787337315E+0003
20	64.40264940005365714539296129493 59	1.5821998698373199700E+0003

eigenvalues, the axial vibration eigenvalues are also of interest. They will be the roots of the characteristic equation for the axial vibration of a fixed-fixed bar. From Ref. [41], that equation is

$$\sin(\ell\omega_i \sqrt{\rho/E}\,)=0 \tag{29}$$

where, ρ is the mass density of the material, and the remaining variables have been defined above. The positive, nonzero roots of this equation are, $i\pi$, i=1,2,3.... When the material and geometry values for this example are substituted into Eq. (29), the axial eigenvalues are given by the equation,

$$\omega_i = i\ 8{,}444.384175\,\pi\ (\text{rad/sec}),\ i=1,\ 2,\ 3,\ \ldots \tag{30}$$

The eigenvalues from Eqs. (28) and (30) were grouped and ordered in increasing magnitude. Once this list of eigenvalues was assembled, all the eigenvalues below 100,000 (rad/sec) were placed in Table 3. In this table, it is simple to determine which eigenvalues correspond to the axial vibrations of the beam because they have been left in the form of Eq. (30). That is, they are in the form of a constant times π. Also, in this table, the bending eigenvalues have been reported with the all the digits obtained from the VAX BASIC program used to evaluate them.

The two right columns of Table 3 contain the eigenvalues obtained with two exact-element models of the fixed-fixed beam in DSTAP. A two-element and a four-element models were prepared and analyzed with DSTAP. DSTAP was then requested to find all the eigenvalues in the frequency range from 1 rad/sec to 100,000 rad/sec. The step size for the incremental search part of the analysis was set at 400 rad/sec. The eigenvalues obtained by DSTAP have been reported reported with all the digits available from DSTAP. The input file supplied to DSTAP for the two-element model is presented in Appendix D, along with the corresponding DSTAP output. The Wittrick-Williams counter,

Table 3. Comparison Between Eigenvalues from Closed-Form Solution and Two Exact Element Models.

Eig. No.	Eigenvalue from Characteristic Equation, see Table 2 (rad/sec)	Eigenvalues from DSTAP 2 Element Model (rad/sec)	Exact Element Models 4 Element Model (rad/sec)
1	568.11452204081812654459542921931 9	568.11452200995041	568.11452200995051
2	1566.02955644747932437555625425702 0	1566.02955636311677	1566.02955636311660
3	3070.04408286797629209595426030849 0	3070.04408270259415	3070.04408270259376
4	5074.93881985546648180604298415582 0	5074.93881958207260	5074.93881958207805
5	7581.08305694470346010614373583736 0	7581.08305653635637	7581.08305653631123
6	10588.45476853051971313412656406550 0	10588.45476796008461	10588.45476796011121
7	14097.05517069221596797941223911030 0	14097.05516832095850	14097.05516832227977
8	18106.88419675103585034124229410050 0	18106.88419377764967	18106.88419379174047
9	22617.94185105266838456400117826150 0	22617.94184767444040	22617.94184761731458
10	8444.384175 π [a]	26528.81529027919487	26528.81529027919532
11	27630.22813357639845727881082606020 0	27630.22812977982403	27630.22812963752040
12	33143.74304432923280028466446569660 0	33143.74303937236709	33143.74303986000541
13	39158.48658331108784449666945634550 0	39158.48657111666944	39158.48657828463820
14	45674.45875052198176111253266516660 0	45674.45873240057063	45674.45874491329778
15	52691.65954596191369097277779312580 0	52691.65953725586132	52691.65953973247724
16	16888.76835 π [b]	53057.63058055839065	53057.63058055838974
17	60210.08896963088367439902363685540 0	60210.08919613237413	60210.08896276889118
18	68229.74702152889170951079640564310 0	68229.74224404596498	68229.74701399454716
19	76750.63370165593779639532113411920 0	76750.62564726539495	76750.63369342389524
20	25333.15253 π [c]	79586.44587083758597	79586.44587083758597
21	85772.74901001202193504857316206130 0	85772.84231593319964	85772.74900103338587
22	95296.09294659714412547073184129400 0	95296.25798389694683	95296.09293674484616

Notes: [a] 8444.384175 π = 26528.81528826990696439594387787...
[b] 16888.76835 π = 53057.63057653981392879188775574...
[c] 25333.15253 π = 79586.44586480972089318783163362...

presented in Eq. (10), is also demonstrated in this Appendix by performing a second analysis of the two-element model. In the reanalysis, DSTAP is requested to find the eigenvalues between 20,000 and 30,000 rad/sec. The sequential number of the eigenvalues in this frequency range is obtained by letting the counter determine how many structure eigenvalues lie below 20,000 rad/sec.

Comparing the columns in Table 3, it may be concluded that the exact displacement method does indeed find the exact eigenvalues. The values in the left column are for practical purposes the eigenvalues of the structure. They are based on the characteristic equation of the boundary value problem and they were computed in quad-precision, with a 33 digit mantissa. The middle column of Table 3 are the eigenvalues of the two-exact-element model in DSTAP. Comparing these eigenvalues with the exact eigenvalues, it can be verified that the DSTAP eigenvalues match to 9 significant digits with the correct eigenvalues for the eigenvalues below 60,000 rad/sec. Above 60,000 rad/sec, the DSTAP eigenvalues start deviating very slightly from the correct eigenvalues. By the 22^{nd} eigenvalue, the last eigenvalue in the range of interest, the DSTAP eigenvalue matches to 5 significant digits with the correct eigenvalues. The worst deviation from the correct eigenvalues occurs at this eigenvalue. This deviation is 0.165 rad/sec, which is a relative error of 0.000 173%. This is well within what is considered an acceptable result for practical applications. Thus, the two-exact-element (3 degrees of freedom) model is accurate enough to obtain the exact eigenvalues.

As a verification of the above conclusion, a four-exact-element (9 degrees of freedom) model was also run in DSTAP. The eigenvalues of this

model are presented in the third column of Table 3. Comparing these eigenvalues with the correct eigenvalues, it is evident that they match to at least 9 significant digits for the entire frequency range [1., 100,000.] rad/sec. Also, if the 10^{th} significant digit of the DSTAP eigenvalues is obtained by rounding, then the 10 significant digits match exactly. Thus, the conclusion that the exact-element models obtain the exact eigenvalues is more strongly reaffirmed by the DSTAP eigenvalues of the four-exact-element model.

In the following examples the frequency ranges of interest will lie below 50,000 rad/sec. The eigenvalues reported in these examples will be given with at most 9 significant digits. Thus, the eigenvalues obtained from the exact displacement method will be considered the correct eigenvalues. Eigenvalues from other models will be compared to these eigenvalues to determine whether or not they are acceptable. The definition of an acceptable eigenvalue will be presented in the Portal Arch example.

Having established the applicability of the exact beam element, the second comparison in this example seeks to determine how well the finite element added to DSTAP approximates the results obtained by a commercial code. It was not possible to add the exact beam element formulation to a commercial finite element code. Therefore, a standard finite element beam formulation was added to DSTAP. Before using this element in case studies, it is appropriate to compare its results with the results of a commercial code. The code chosen for this comparison was General Electric's SUPERB finite element program [15].

The three geometries presented in Fig. 8 were modeled with FEM beam elements in DSTAP and in SUPERB. The results from these analyses are

(a) Two element model

(b) Four element model

(c) Eight element model

Figure 8. Models Used in Example A.

presented in Table 4. The table is divided into three wide columns, one for each geometry of Fig. 8. Each one of these wide columns is divided into three smaller columns. From left to right these columns are: (1) the set of results obtained from a DSTAP run, (2) the set of eigenvalues from a SUPERB run, and (3) the relative difference between the analyses. For example, the central wide column contains the results that correspond to the geometry in Fig. 8-b. This model had 4 elements and 9 degrees of freedom in both DSTAP and SUPERB. It took 0.92 of a CPU second to solve this problem in DSTAP, while it took 15.76 CPU seconds to solve the same model in SUPERB. The largest relative difference in this model between the eigenvalues computed with the two codes occurred in the sixth eigenvalue, where DSTAP computed it as 15,808.2841 rad/sec and SUPERB computed it as 15,772.68 rad/sec, a relative difference of 0.225%.

The relative difference column in Table 4 is not the same as the relative error of the computed eigenvalues. The relative error may be obtained by using the closed-form solution eigenvalues which have been reproduced in Table 4, truncated to 4 decimal places. For example, sixth eigenvalue from the closed-form solution column is 10,588.4547 rad/sec. Thus, the relative error that both DSTAP and SUPERB had for the sixth eigenvalue from the 4-FEM-element model was approximately 49%.

The main reason for computing the relative difference between the two codes is to demonstrate that the two codes give comparable results for the FEM beam element formulations. The largest relative difference between the two codes occurred in the third model, sixteenth eigenvalue, where the relative difference was approximately 1%. The available SUPERB documentation was rather ambiguous concerning both the plane-beam element formulation and the

Table 4. Comparison Between FEM models in DSTAP and SUPERB Eigenvalues (rad/sec).

Program:		CLOSED-	DSTAP	SUPERB		DSTAP	SUPERB		DSTAP	SUPERB	
Routine:		FORM SOL.[1]	EIGENR	JACOBI		EIGENR	JACOBI		EIGENR	JACOBI	
# Elements[2]:			2	2	Rel.	4	4	Rel.	8	8	Rel.
# dof[3]:			3	3	Diff.	9	9	Diff.	21	21	Diff.
CPU secs.[4]:			0.7600	15.1300	(%)	0.9200	15.7600	(%)	2.0700	18.5700	(%)
E	1	568.1145	577.3233	577.2927	0.0052	568.8685	568.8356	0.005	568.1624	568.1319	0.005
i	2	1566.0295	2081.5687	2081.0538	0.0247	1580.5149	1580.1582	0.022	1567.0189	1566.712	0.019
g	3	3070.0440	29252.2048	29251.9975	0.0007	3135.6118	3134.1785	0.045	3077.3190	3075.933	0.045
e	4	5074.9388				5932.3102	5927.3056	0.084	5106.5973	5102.637	0.077
n	5	7581.0830				9811.1566	9797.3708	0.140	7680.4152	7671.140	0.120
v	6	10588.4547				15808.2841	15772.6800	0.225	10827.0682	10808.33	0.173
a	7	14097.0551				27214.8261	27214.9888	0.000	14403.8973	14369.64	0.237
l	8	18106.8841				58504.4097	58504.6233	0.000	20331.8919	20268.29	0.312
u	9	22617.9418				95071.9966	95070.8769	0.001	25956.9975	25854.67	0.394
e	10	26528.8152							26699.5893	26699.76	0.000
s	11	27630.2281							33148.3963	32984.83	0.493
	12	33143.7430							42068.6960	41810.20	0.614
	13	39158.4865							52925.8963	52522.40	0.762
	14	45674.4587							54429.6523	54429.97	0.000
	15	52691.6595							65279.4975	64666.54	0.938
	16	53057.6305							76624.1348	75768.93	1.116
	17	60210.0889							84227.5928	84226.09	0.001
	18	68229.7470							117008.8194	117011.7	-0.002
	19	76750.6336							153002.1231	153001.8	0.000
	20	79586.4458							190143.9932	190141.7	0.001
	21	85772.7490							221254.4869	221256.0	0.000

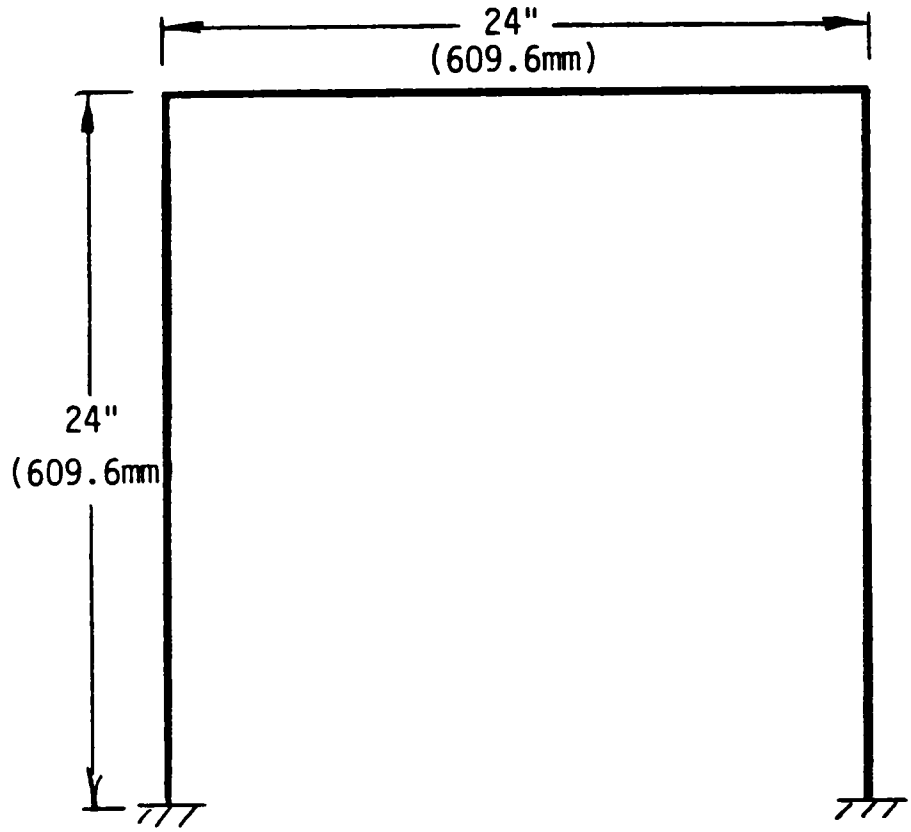
Notes: [1] This column contains the exact eigenvalues presented in Table 3
[2] This row contains the number of elements in the model
[3] This row contains the number of degrees of freedom
[4] This row contains the number of CPU seconds the model took to run

SUPERB version of the Modified Jacobi technique for extracting eigenvalues. Therefore, there was no easy way to implement an element in DSTAP that would give the same results as SUPERB.

Based on this example, it can be concluded that the exact displacement beam element arrives at almost the same eigenvalues as if the characteristic equation had been solved. For practical purposes, the exact-element models gave the correct eigenvalues. The fixed-fixed beam was modeled with only two elements, three degrees of freedom. These three degrees of freedom were enough to compute all the eigenvalues below 60,000 rad/sec correctly to the ninth or tenth digits. Between 60,000 and 100,000 rad/sec, these three degrees of freedom were enough to compute all the eigenvalues to the fifth significant digit. The FEM models tested here were not able to match this feat. The best FEM result obtained was for the first eigenvalue of the 21-degrees-of-freedom model which matched the correct eigenvalues to 4 digits. The second conclusion arrived at here was that the beam finite element along with the EIGENR routine incorporated into DSTAP gave approximately the same results as SUPERB. SUPERB took about an order of a magnitude longer to carry out the computation. This is probably because the overhead in SUPERB is necessarily much greater than in DSTAP.

B. Portal Arch:

The second example presented here is the Portal Arch, which is shown in Fig. 9. It is made of three identical beams, which have the same properties as the fixed-fixed beam in the previous example. The legs of the Portal Arch are built into the ground, so that the translational and rotational degrees of freedom are eliminated.



Material Properties:

E = 30.0 Mpsi (207. GPa)
G = 11.6 Mpsi (79.9 GPa)
ρ = 0.282 lb_f/in^3 (7.81 mgr/mm^3)

Figure 7. Schematic of the Beam for Example A.

The Portal Arch was first modeled with 3 exact beam elements to determine the first 35 or so eigenvalues. In this model, one element was assigned to each side of the arch, giving a total of 6 active degrees of freedom. This model was then input to DSTAP with the shear deformation and rotatory inertia options on. The first 35 eigenvalues obtained this way are presented in Table 5. Next, the same model was analyzed without shear deformation and rotatory inertia effects. These eigenvalues also appear in Table 5, along with the absolute and relative difference between the computed eigenvalues. The relative difference column in Table 5 shows that the shear deformation and rotatory inertia effects are not significant in the first 35 eigenvalues. This was expected since the arch beams meet the classical definition of slender beams; that is, the length to largest cross section dimension is greater than 8. Therefore, the shear deformation effects may be neglected for lower circular frequencies.

The eigenvectors corresponding to the eigenvalues with shear and rotatory inertia are presented in Fig. 10. These eigenvectors were reconstructed in the post-processing option of DSTAP, using the technique described in the Program Implementation section. This catalogue of mode shapes for the Portal Arch is believed to be unique. A sample output for the mode shapre reconstruction in DSTAP is presented in Appendix D. This output corresponds to the 10^{th} eigenvalue. That is, DSTAP was requested to reanalyze the Portal Arch so that the only eigenvalue it could find was the 10^{th} eigenvalue. Then, DSTAP was requested to reconstruct the mode shape using the TMM. The output shown in Appendix D includes most of the optional output, such as: the initial state vector for each reconstructed element, the transfer matrix used for each exact-element, the full mode shape (deflection,

Table 5. Portal Arch Exact-Element Model Eigenvalues, With and Without Shear Deformation and Rotatory Inertia.

G (psi)=	1.150E07	infinite		
i_y (in)=+	0.0722	zero	Absolute	Relative
dof:	6	6	Difference	Difference
CPU sec.:	117.1900	111.4100	(rad/sec)	(%)
1	81.3588	81.3702	0.01136	0.01396
2	321.0140	321.1035	0.08943	0.02785
3	523.5551	523.8114	0.25626	0.04894
4	567.5300	567.8924	0.36233	0.06384
5	1145.9509	1146.9407	0.98971	0.08636
6	1399.3720	1401.0730	1.70092	0.12154
7	1618.2454	1620.6311	2.38564	0.14742
8	2454.8651	2459.1925	4.32734	0.17627
9	2898.4833	2905.0732	6.58981	0.22735
10	3055.0334	3063.0854	8.05194	0.26356
11	4265.4817	4278.1797	12.69799	0.29769
12	4751.2740	4768.5267	17.25264	0.36311
13	5100.3959	5121.2102	20.81425	0.40809
14	6543.7767	6573.0268	29.25008	0.44699
15	7241.8762	7280.4695	38.59325	0.53291
16	7484.3794	7527.8916	43.51216	0.58137
17	9271.7240	9328.1874	56.46331	0.60898
18	10046.0964	10119.1890	73.09255	0.72757
19	10446.7596	10525.9006	79.14098	0.75756
20	11979.3648	12032.0228	52.65791	0.43957
21	12898.8176	12917.9393	19.12165	0.14824
22	13171.2416	13260.7219	89.48030	0.67936
23	13526.9674	13655.9250	128.95754	0.95333
24	14301.7357	14416.1023	114.36653	0.79966
25	16569.3844	16746.9062	177.52179	1.07138
26	17188.0366	17392.3481	204.31141	1.18868
27	18017.0837	18247.5027	230.41900	1.27889
28	20660.9769	20939.7906	278.81365	1.34946
29	21680.0371	22007.5501	327.51297	1.51066
30	21932.2404	22239.2230	306.98254	1.39968
31	24791.8520	24999.2945	207.44250	0.83673
32	25332.9633	25754.6906	421.72724	1.66473
33	27091.5871	27608.7668*	517.17967	1.90900
34	27366.0790	27746.7669	380.68782	1.39109
35	30399.8477	31009.4317	609.58397	2.00522

+i_y is the radius of Gyration

*This eigenvalue was obtained with a 6-exact-element model. There were many poles around this eigenvalue in the 3-exact-element model, which made the extraction difficult.

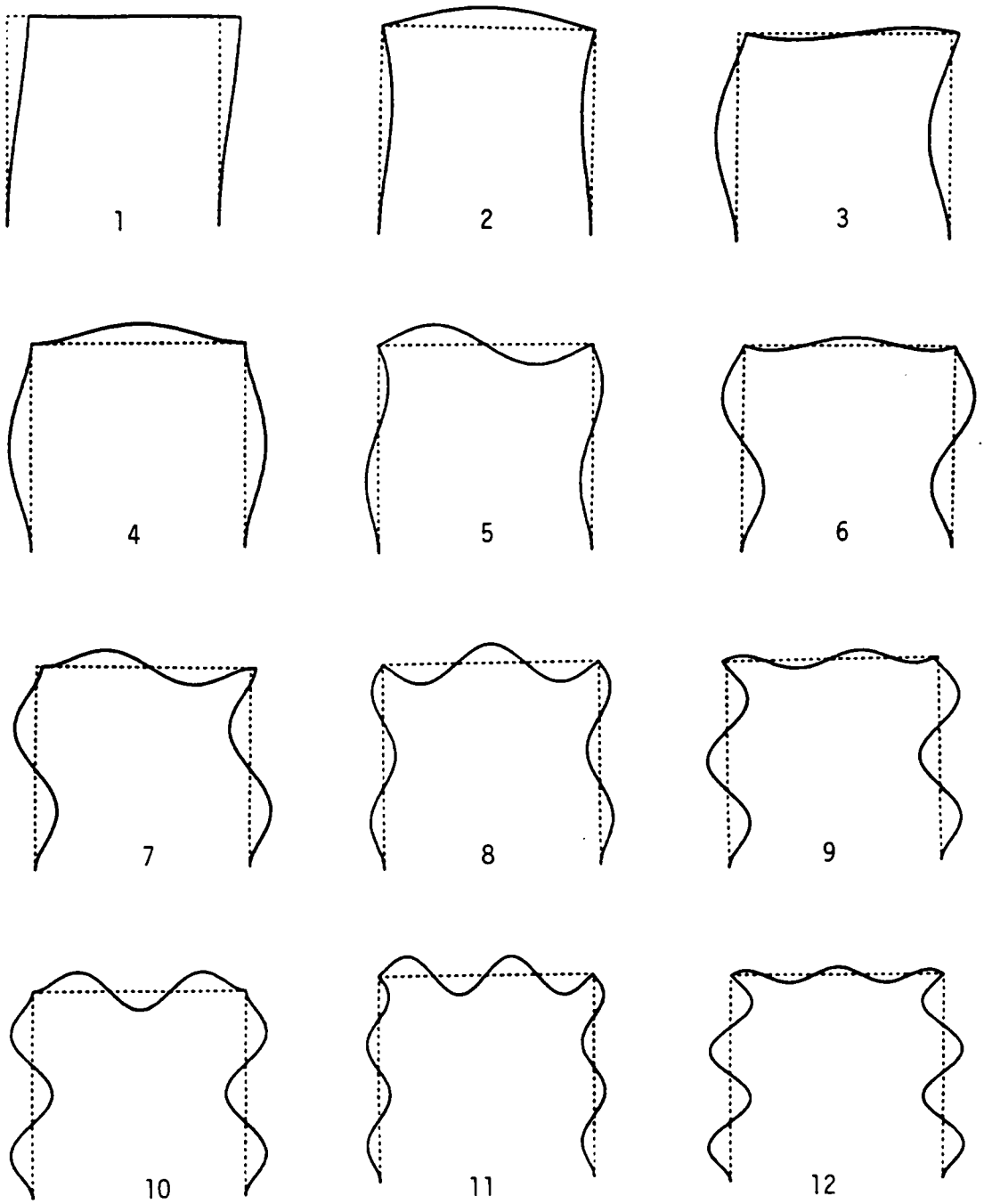


Figure 10. Modes Shapes for the Portal Arch, See Table 11.

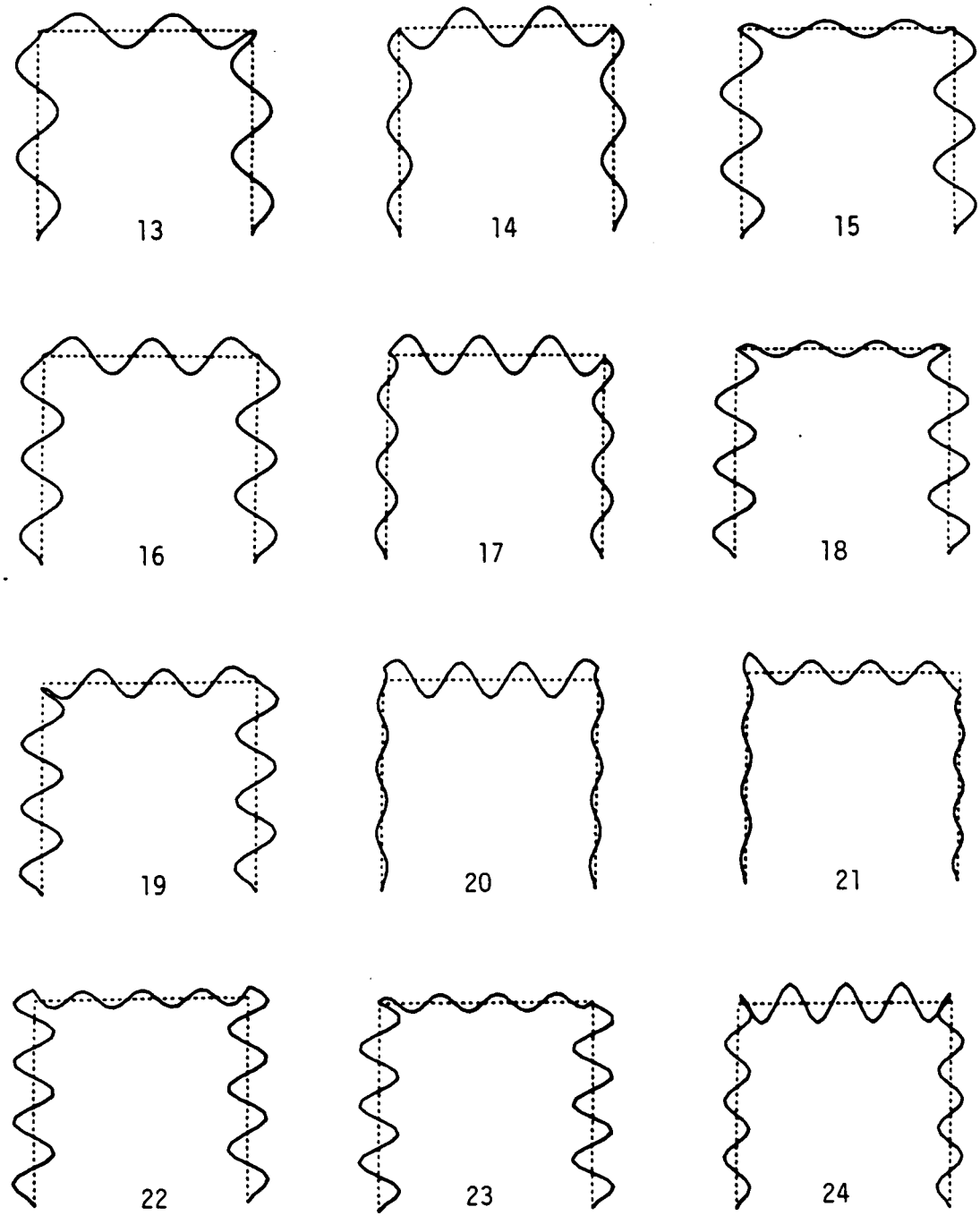


Figure 10. Modes Shapes for the Portal Arch, See Table 11. (continued)

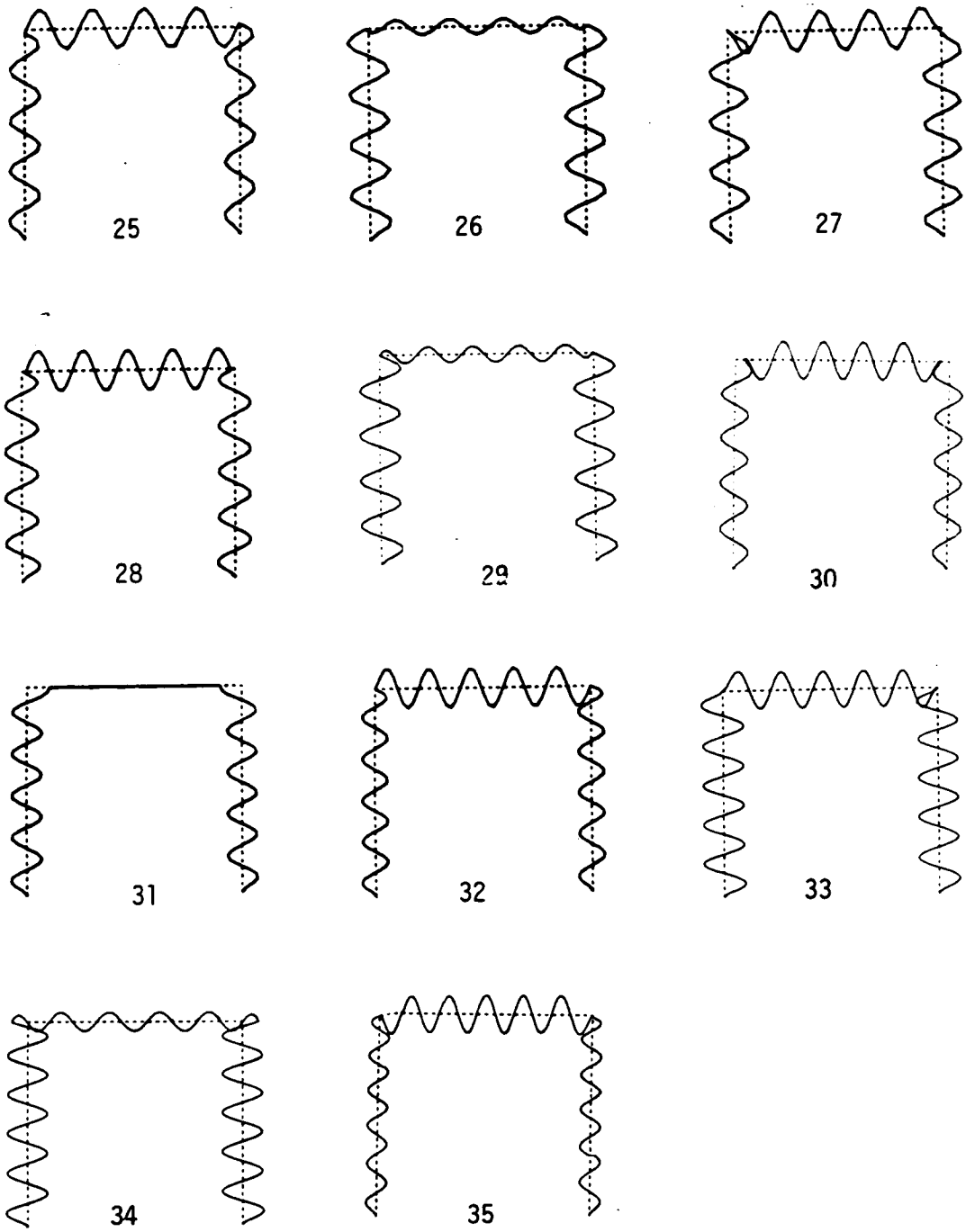


Figure 10. Modes Shapes for the Portal Arch, See Table 11. (continued)

slope, moment, and shear), and the scaled mode shape (deflection only) for plotting.

The arch in Fig. 10 was also modeled with finite elements to determine whether the first 30 or so eigenvalues could be extracted satisfactorily. The first 30 eigenvalues were chosen because the finite element in DSTAP does not have shear deformation and rotatory inertia capabilities. It was seen in Table 5 that these effects do not become significant factors until the 34^{th} eigenvalue. Having established a desired number of eigenvalues, the definition of a satisfactory eigenvalue had to be established. The set of acceptable eigenvalues for this study was defined as all lower eigenvalues up to the first occurrence of a relative difference of more than 5 percent. The relative difference used was between the finite element model eigenvalues and the exact eigenvalues in the third column of Table 5.

In every finite element model developed for the Portal Arch, each segment was divided into the same number of equal-size elements. This element distribution was chosen to keep the comparisons independent of the modeling technique used in each model. Hence, the first model used one element to model each side of the arch. The next model used two elements to model each side, and so on. The results obtained from these models are tabulated in Table 6.

The left side of Table 6 contains thirty correct eigenvalues from Table 5, for comparison purposes. The right side of Table 6 contains the results from the different finite element models in DSTAP. Moving from left to right, the number of degrees of freedom per model increases. The results of each model occupy two columns in the table. The top of the first column contains the model parameters, such as number of elements, eigenvalue

Table 6. Portal Arch Finite Element Model Eigenvalues.

Program Routine	DSTAP BFMTHD	DSTAP EIGENR		DSTAP EIGENR		DSTAP EIGENR		DSTAP EIGENR	
# of FE	0	3		6		9		12	
# of CE	3	0	Rel.	0	Rel.	0	Rel.	0	Rel.
dof	6	6	Diff.	15	Diff.	24	Diff.	33	Diff.
CPU sec		0.9300	(%)	1.3400	(%)	2.5700	(%)	5.1000	(%)
1	81.3702	80.4074	-1.18	81.9956	0.76	81.7642	0.48	81.6163	0.30
2	321.1035	384.3153	19.68	336.6700	4.84	330.5853	2.95	327.0545	1.85
3	523.8114	737.3769	40.77	575.6669	9.89	561.8787	7.26	549.3870	4.88
4	567.8924	9242.3121	1527.	600.9932	5.82	593.2593	4.46	585.8758	3.16
5	1146.9407	11565.8657	908.4	1259.4344	9.80	1209.6473	5.46	1197.2259	4.38
6	1401.0730	16889.2266	1105.	1667.1899	18.99	1670.1834	19.20	1603.1965	14.42
7	1620.6311			2080.7392	28.39	1849.2565	14.10	1807.8869	11.55
8	2459.1925			3242.5577	31.85	2698.8713	9.74	2611.5018	6.19
9	2905.0732			4176.0193	43.74	3182.3908	9.54	3499.1263	20.44
10	3063.0854			11901.2435		3564.0379	16.35	3599.9348	17.52
11	4278.1797			12745.0689		4739.0393	10.77	4730.1071	10.56
12	4768.5267			21370.4003		5492.1177	15.17	5257.0530	10.24
13	5121.2102			37112.7394		6837.0365	33.50	5680.8305	10.92
14	6573.0268			37224.5293		8965.6496	36.40	6990.6763	6.35
15	7280.4695			45174.2353		9921.7519	36.27	7731.9373	6.20
16	7527.8916					13095.8040	73.96	8939.3861	18.75
17	9328.1874					13869.7972	48.68	10166.1388	8.98
18	10119.1890					22821.8220	125.5	10546.4189	4.22
19	10525.9006					40682.5523	286.4	13038.1753	23.86
20	12032.0228					40790.6590	239.0	14935.1844	24.12
21	12917.9393					48380.5309	274.5	15450.5719	19.60
22	13260.7219					64168.0204	383.8	18077.6504	36.32
23	13655.9250					64193.9858	370.0	20193.4744	47.87
24	14416.1023					74811.8918	418.9	23643.3254	64.00
25	16746.9062							42082.7056	151.2
26	17392.3481							42147.7442	142.3
27	18247.5027							49269.4254	170.0
28	20939.7906							70524.0782	236.7
29	22007.5501							70545.9382	220.5
30	22239.2230							77879.8456	250.1

Table 6. Portal Arch Finite Element Model Eigenvalues (continued)

Program Routine	DSTAP BFMTHD	DSTAP EIGENR		DSTAP EIGENR		DSTAP EIGENR		DSTAP EIGENR	
# of FE	0	15		21		27		33	
# of CE	3	0	Rel.	0	Rel.	0	Rel.	0	Rel.
dof	6	42	Diff.	60	Diff.	78	Diff.	96	Diff.
CPU sec		9.3900	(%)	25.8500	(%)	55.6600	(%)	106.8700	(%)
1	81.3702	81.5352	0.20	81.4577	0.10	81.4240	0.06	81.4065	0.04
2	321.1035	325.1102	1.24	323.2398	0.66	322.4195	0.40	321.9925	0.27
3	523.8114	541.4888	3.37	533.4182	1.83	529.7689	1.13	527.8487	0.77
4	567.8924	580.6399	2.24	574.9707	1.24	572.3200	0.77	570.9060	0.53
5	1146.9407	1184.9243	3.31	1169.5037	1.96	1161.5115	1.27	1157.0293	0.87
6	1401.0730	1550.4349	10.66	1486.5476	6.10	1455.0120	3.84	1437.9099	2.62
7	1620.6311	1764.9179	8.90	1705.6553	5.24	1674.7679	3.34	1657.7385	2.28
8	2459.1925	2597.6407	5.62	2557.0343	3.97	2528.2678	2.80	2509.4814	2.04
9	2905.0732	3423.1899	17.83	3249.7084	11.86	3134.1679	7.88	3065.1001	5.50
10	3063.0854	3520.4894	14.93	3357.6890	9.61	3254.2915	6.24	3194.5238	4.29
11	4278.1797	4555.3870	6.47	4499.5876	5.17	4442.5548	3.84	4402.5345	2.90
12	4768.5267	5849.7068	22.67	5632.8730	18.12	5383.0306	12.88	5209.7872	9.25
13	5121.2102	6082.5899	18.77	5928.6924	15.76	5693.3284	11.17	5527.3460	7.93
14	6573.0268	7309.2210	11.20	6970.9751	6.05	6886.7126	4.77	6822.4032	3.79
15	7280.4695	7908.6567	8.62	8821.7980	21.17	8535.7836	17.24	8237.6945	13.14
16	7527.8916	8330.6246	10.66	9019.8128	19.81	8716.7190	15.79	8405.6031	11.65
17	9328.1874	9883.9861	5.95	9979.8495	6.98	9855.8027	5.65	9733.1997	4.34
18	10119.1890	10436.2087	3.13	12205.0045	20.61	12191.0940	20.47	11843.5854	17.04
19	10525.9006	11538.9652	9.62	12222.8783	16.12	12384.8776	17.66	12179.6822	15.71
20	12032.0228	12171.9492	1.16	12618.3705	4.87	12431.2135	3.31	12275.5567	2.02
21	12917.9393	12214.4087	-5.44	13065.7560	1.14	13072.4185	1.19	12966.9298	0.37
22	13260.7219	13971.9695	5.36	14230.0287	7.30	13999.0244	5.56	14004.8451	5.61
23	13655.9250	15113.4268	10.67	14970.4706	9.62	16542.0768	21.13	16321.3862	19.51
24	14416.1023	17223.9989	19.47	15514.1616	7.61	17013.3369	18.01	16723.6579	16.00
25	16746.9062	21238.6104	26.82	17567.8200	4.90	17914.5877	6.97	17913.2748	6.96
26	17392.3481	23407.8678	34.58	18405.1997	5.82	21204.9640	21.92	21123.4245	21.45
27	18247.5027	24165.7571	32.43	19454.2764	6.61	21738.8886	19.13	22067.7959	20.93
28	20939.7906	25434.7970	21.46	21493.2864	2.64	23246.8911	11.01	22285.1198	6.42
29	22007.5501	29184.7022	32.61	22542.8885	2.43	24220.8148	10.05	25425.4957	15.53
30	22239.2230	31388.3407	41.13	24228.8848	8.94	24289.5525	9.21	26326.2515	18.37

Table 6. Portal Arch Finite Element Model Eigenvalues (continued)

Program Routine	DSTAP BFMTHD	DSTAP EIGENR		DSTAP EIGENR		DSTAP EIGENR		DSTAP EIGENR	
# of FE	0	39		45		57		69	
# of CE	3	0	Rel.	0	Rel.	0	Rel.	0	Rel.
dof	6	114	Diff.	132	Diff.	168	Diff.	204	Diff.
CPU sec		176.1500	(%)	279.9700	(%)	584.4800	(%)	1032.0000	(%)
1	81.3702	81.3963	0.03	81.3898	0.02	81.3825	0.01	81.3786	0.01
2	321.1035	321.7434	0.19	321.5857	0.15	321.4051	0.09	321.3099	0.06
3	523.8114	526.7223	0.55	526.0073	0.41	525.1868	0.26	524.7530	0.17
4	567.8924	570.0706	0.38	569.5381	0.28	568.9249	0.18	568.5999	0.12
5	1146.9407	1154.3070	0.64	1152.5424	0.48	1150.4818	0.30	1149.3769	0.21
6	1401.0730	1427.7402	1.90	1421.2381	1.43	1413.7358	0.90	1409.7538	0.61
7	1620.6311	1647.5429	1.66	1641.0021	1.25	1633.4367	0.79	1629.4144	0.54
8	2459.1925	2497.0450	1.53	2488.5477	1.19	2478.1763	0.77	2472.4130	0.53
9	2905.0732	3022.2692	4.03	2994.2887	3.07	2961.4920	1.94	2943.8861	1.33
10	3063.0854	3158.3254	3.10	3135.0625	2.34	3108.2033	1.47	3093.9735	1.00
11	4278.1797	4374.4641	2.25	4354.3973	1.78	4328.7781	1.18	4313.9525	0.83
12	4768.5267	5095.9704	6.86	5019.5644	5.26	4928.3697	3.35	4878.8323	2.31
13	5121.2102	5419.3396	5.82	5347.7773	4.42	5263.7323	2.78	5218.8405	1.90
14	6573.0268	6774.4755	3.06	6738.0148	2.51	6687.8552	1.74	6656.5812	1.27
15	7280.4695	8012.5320	10.05	7851.0672	7.83	7649.6756	5.07	7537.1018	3.52
16	7527.8916	8180.5893	8.67	8025.3242	6.60	7839.6993	4.14	7740.3395	2.82
17	9328.1874	9649.2191	3.44	9589.5782	2.80	9511.1118	1.96	9462.7783	1.44
18	10119.1890	11494.8017	13.59	11214.0516	10.81	10840.6480	7.12	10624.2982	4.99
19	10525.9006	11841.2802	12.49	11561.9707	9.84	11194.6203	6.35	10986.7158	4.37
20	12032.0228	12207.0936	1.45	12166.5590	1.11	12121.1363	0.74	12096.3381	0.53
21	12917.9393	12937.3082	0.14	12929.1925	0.08	12925.3954	0.05	12924.4872	0.05
22	13260.7219	13964.4636	5.30	13916.5094	4.94	13813.1663	4.16	13712.8479	3.40
23	13655.9250	15911.9671	16.52	15515.3112	13.61	14921.0303	9.26	14552.2050	6.56
24	14416.1023	16222.5769	12.53	15788.9241	9.52	15218.6317	5.56	14917.5085	3.47
25	16746.9062	17714.6553	5.77	17559.0009	4.84	17358.2196	3.65	17229.6925	2.88
26	17392.3481	20739.1000	19.24	20247.4633	16.41	19400.2704	11.54	18835.3323	8.29
27	18247.5027	21620.9948	18.48	21036.0720	15.28	20099.6591	10.15	19517.2543	6.95
28	20939.7906	22081.5786	5.45	21909.9113	4.63	21714.9020	3.70	21595.5639	3.13
29	22007.5501	25258.3320	14.77	24916.5848	13.21	24204.2912	9.98	23645.8159	7.44
30	22239.2230	26197.5452	17.79	25844.4662	16.21	24996.6103	12.39	24249.2741	9.03

Table 6. Portal Arch Finite Element Model Eigenvalues (continued)

Program Routine	DSTAP BFMTHD	DSTAP EIGENR		DSTAP EIGENR	
# of FE	0	81		93	
# of CE	3	0	Rel.	0	Rel.
dof	6	240	Diff.	276	Diff.
CPU sec		1712.4700	(%)	3219.3600	(%)
1	81.3702	81.3763	0.00	81.3748	0.00
2	321.1035	321.2534	0.04	321.2172	0.03
3	523.8114	524.4956	0.13	524.3307	0.09
4	567.8924	568.4068	0.09	568.2829	0.06
5	1146.9407	1148.7160	0.15	1148.2906	0.11
6	1401.0730	1407.3852	0.45	1405.8657	0.34
7	1620.6311	1627.0196	0.39	1625.4826	0.29
8	2459.1925	2468.8961	0.39	2466.6046	0.30
9	2905.0732	2933.3501	0.97	2926.5687	0.73
10	3063.0854	3085.5221	0.73	3080.1083	0.55
11	4278.1797	4304.6812	0.61	4298.5464	0.47
12	4768.5267	4849.0175	1.68	4829.7706	1.28
13	5121.2102	5192.1148	1.38	5174.9888	1.05
14	6573.0268	6636.0668	0.95	6622.0646	0.74
15	7280.4695	7468.4420	2.58	7423.8097	1.96
16	7527.8916	7681.4723	2.04	7643.9638	1.54
17	9328.1874	9430.7726	1.09	9408.6387	0.86
18	10119.1890	10490.4252	3.66	10402.8015	2.80
19	10525.9006	10860.6685	3.18	10779.4943	2.40
20	12032.0228	12080.7604	0.40	12070.2249	0.31
21	12917.9393	12923.8883	0.04	12923.3097	0.04
22	13260.7219	13626.2904	2.75	13557.0183	2.23
23	13655.9250	14317.8028	4.84	14162.5584	3.70
24	14416.1023	14752.8315	2.33	14656.8390	1.66
25	16746.9062	17136.4133	2.32	17065.7314	1.90
26	17392.3481	18466.0236	6.17	18217.9854	4.74
27	18247.5027	19156.5597	4.98	18925.4086	3.71
28	20939.7906	21501.6349	2.68	21422.0001	2.30
29	22007.5501	23255.7946	5.67	22992.4234	4.47
30	22239.2230	23706.5373	6.59	23326.1119	4.88

extraction routine, and execution time. For example, the first model in Table 6 has 3 finite elements, and it took 0.93 of a CPU second to find the eigenvalues, using EIGENR. The 6 degrees of freedom allowed the extraction of 6 eigenvalues. These eigenvalues appear below the model data, in the same column. The second column allotted to each model contains the relative difference between model eigenvalues and the exact eigenvalues, expressed as a percentage. For example, the 6 degree of freedom model only has one eigenvalue with a relative difference less than 5 per cent, the first one. The remaining eigenvalues for this model have relative differences greater than 5 per cent.

Note, the acceptable eigenvalue obtained for the 6-degrees-of-freedom model is questionable. Meirovitch [6] has shown that the FEM eigenvalue problem is a form of Rayleigh-Ritz approximation. This guarantees that the eigenvalues obtained with a FEM discretization will be upper bounds on the true eigenvalues; in other words, they should be greater than the exact eigenvalues. This is true for all the eigenvalues in Table 6, except for the first eigenvalue obtained with the 6-degree-of-freedom model. This eigenvalue is less than the exact eigenvalue, indicating that this is an unacceptable result. However, in practice, this eigenvalue would be considered acceptable; thus, this eigenvalue will be considered acceptable here.

Table 6 shows that indeed the finite element models in DSTAP are capable of finding the first 30 or so eigenvalues within the set tolerance. In Table 6, a heavy line separates the acceptable eigenvalues, above the line, from the rest. It is possible that some eigenvalues lying below the line may meet the specified tolerance of 5 per cent. This is because the line was drawn below

the first eigenvalue which had a relative difference greater than 5 per cent. Moving from left to right, this line moves downward, indicating that more acceptable eigenvalues are extracted by the models with more degrees of freedom. Thus, as the degrees of freedom increase, so do the number of acceptable eigenvalues.

The number of acceptable eigenvalues per model size was entered into a statistical package [62], which automatically tested the following 8 regression models,

1. $y = Ax$,
2. $y = A + Bx$,
3. $y = A\ \mathrm{Exp}(x)$,
4. $y = 1/(A + Bx)$,
5. $y = A + B/x$,
6. $y = A + B\ \mathrm{Log}(x)$,
7. $y = A\ x^B$, and
8. $y = x/(A + Bx)$,

where x is the number of degrees of freedom, y is the number of acceptable eigenvalues obtained with a model of x degrees of freedom, A and B are regression constants. From these models the best fit for the finite element models was provided by the equation for a straight line (model 2),

$$y=-0.302 + (0.10298)\ x \tag{31}$$

A plot of this line may be seen in Fig. 11. This regression line may be used to approximate the number of correct eigenvalues knowing the total number of active degrees of freedom in a model. Since one of the question to be answered was "how many active nodes are needed per acceptable eigenvalue?", Eq. (31) is used in reverse form below. Being that Eq. (31) is a regression of x on y, it is improper to use it for a y on x calculation. Nonetheless, this is done here because the answer sought is an approximate answer and the answer is not expected to vary with a proper regression. Thus, (y + 0.302)/0.103

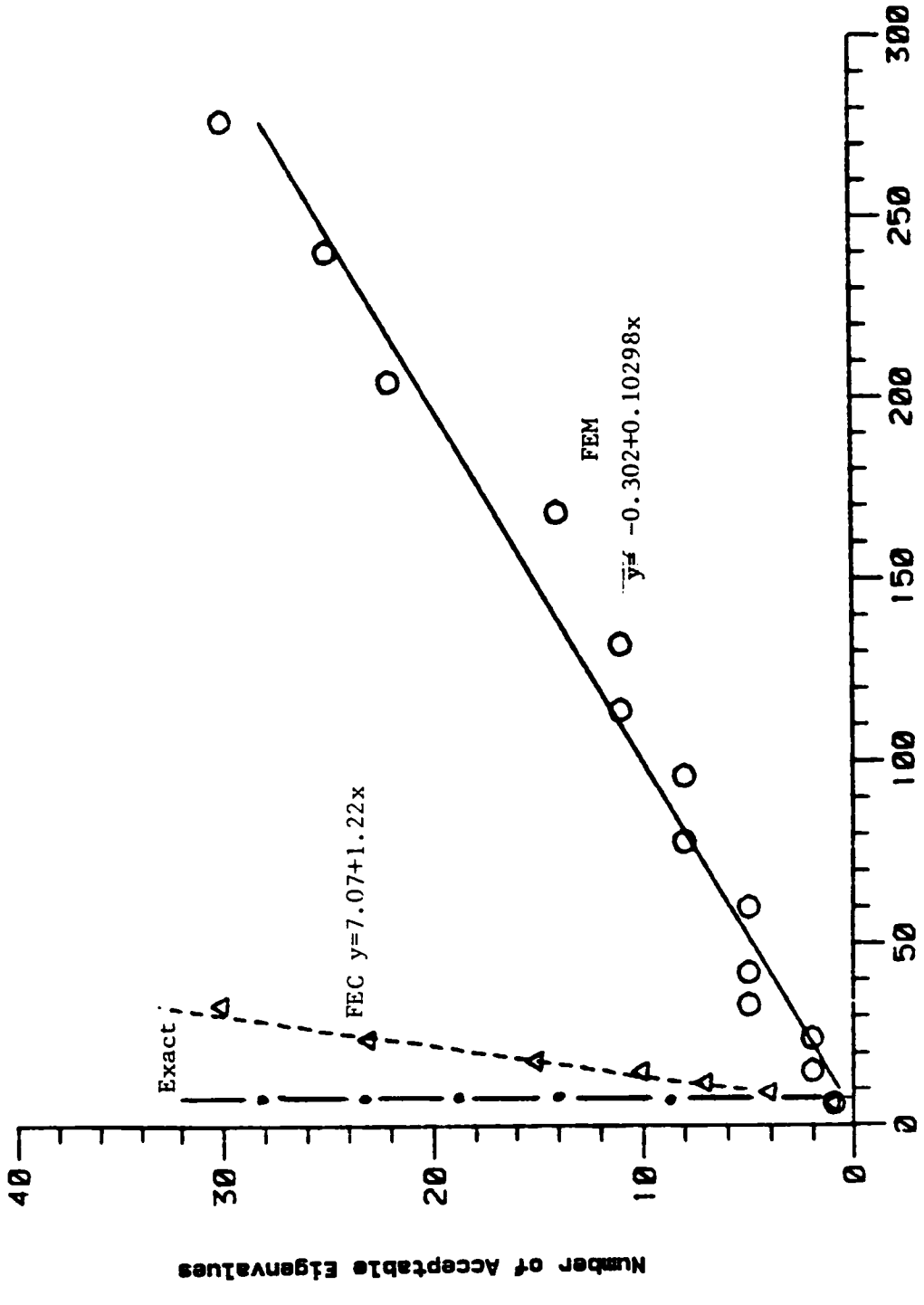


Figure 11. Number of Acceptable Eigenvalues versus Degrees of Freedom for Portal Arch.

active degrees are freedom are needed for the first y eigenvalues to be acceptable, which is approximately equal to 10y + 3. Since there are 3 degrees of freedom per node, then approximately between 3 and 4 active nodes are needed per acceptable eigenvalue.

The regression line in Eq. (31) is also a verification of the proof, as presented by Strang and Fix [13], that a computed eigenvalue can be made sufficiently close to a true eigenvalue by increasing the number of degrees of freedom. Consequently, regardless of how many acceptable eigenvalues are desired, it should be possible to extract them with a sufficiently large model, provided the computer is able to handle the numerical difficulties associated with larger matrices. As presented earlier, it is generally not feasible to increase the size of the model arbitrarily.

In Fig. 11, there is a second curve that corresponds to the finite element - continuous element (FEC) models. The slope of this curve is much steeper than the one for finite elements alone, indicating that fewer finite elements are required to compute the same number of acceptable eigenvalues. The model used to generate this curve had both of the arch legs modeled with exact beam elements, while the top was modeled with finite elements. The number of degrees of freedom in these elements were increased by adding more finite elements to the top of the arch. The data for the FEC curve are presented in Table 7, which is organized similarly to Table 6. The final line in Fig. 11 is a vertical line at 6 dof. This line corresponds to the model made of purely exact beam elements, which was presented in Table 5. Note that no data points are presented for this last line because theoretically any number of eigenvalues can be obtained with the 6-degree-of-freedom model!

The results for the FEM models analyzed here indicate that at least for

Table 7. Portal Arch Combined Model (FEC) Eigenvalues.

Program	DSTAP	DSTAP		DSTAP		DSTAP		DSTAP	
Routine	BFMTHD	BFMTHD		BFMTHD		BFMTHD		BFMTHD	
# of FE	0	1		2		3		4	
# of CE	3	2	Rel.	2	Rel.	2	Rel.	2	Rel.
dof	6	6	Diff.	9	Diff.	12	Diff.	15	Diff.
CPU sec	111.4100	84.5700	(%)	117.8800	(%)	151.3800	(%)	223.1700	(%)
1	81.3702	81.3704	0.00	81.3703	0.00	81.3702	0.00	81.3702	0.00
2	321.1035	351.9085	9.59	322.4091	0.40	321.3797	0.08	321.1924	0.02
3	523.8114	524.9729	0.22	524.4229	0.11	523.9031	0.01	523.8403	0.00
4	567.8924	739.5138	30.22	570.9586	0.53	568.6643	0.13	568.1431	0.04
5	1146.9407	1303.5349	13.65	1236.4063	7.80	1157.8771	0.95	1150.8222	0.33
6	1401.0730	1619.3473	15.57	1402.2725	0.08	1403.3923	0.16	1401.7007	0.04
7	1620.6311	2153.2601	32.86	1777.4224	9.67	1629.5647	0.55	1624.8925	0.26
8	2459.1925	3091.3132	25.70	2668.2318	8.50	2601.1204	5.77	2488.1906	1.17
9	2905.0732	3376.2028	16.21	2979.9512	2.57	2908.2686	0.10	2912.9787	0.27
10	3063.0854	5078.1218	65.78	4035.4435	31.74	3318.7031	8.34	3086.0232	0.74
11	4278.1797	5271.8786	23.22	4668.4929	9.12	4585.3893	7.18	4527.5265	5.82
12	4768.5267	7563.6879	58.61	5554.7787	16.48	4835.8577	1.41	4772.5707	0.08
13	5121.2102	7727.9307	50.90	6356.0937	24.11	5937.0383	15.93	5452.8584	6.47
14	6573.0268	10531.9099	60.22	7781.2465	18.38	7050.3207	7.26	6921.5856	5.30
15	7280.4695	10682.2848	46.72	8207.2945	12.73	7588.5255	4.23	7359.2410	1.08
16	7527.8916	12277.5529	63.09	10608.7764	40.92	9437.9533	25.37	8414.0948	11.77
17	9328.1874	12704.1274	36.19	10825.9317	16.05	10107.2717	8.35	10005.4207	7.26
18	10119.1890	14021.3739	38.56	12878.3789	27.26	11311.2420	11.78	10348.0909	2.26
19	10525.9006	14296.0815	35.81	12995.2982	23.46	11913.8193	13.18	11868.0257	12.75
20	12032.0228	17905.8595	48.81	14283.5372	18.71	13151.5505	9.30	12752.7386	5.98
21	12917.9393	18248.0941	41.26	14625.7616	13.22	13294.3528	2.91	13266.7576	2.70
22	13260.7219	22089.1691	66.57	18018.2383	35.87	15414.4364	16.24	13459.8030	1.50
23	13655.9250	22742.8013	66.54	18384.0679	34.62	15901.6255	16.44	15009.6078	9.91
24	14416.1023	25817.2154	79.08	22151.9892	53.66	18596.6968	28.99	16995.1017	17.88
25	16746.9062	27745.4060	65.67	22801.9465	36.15	19078.5891	13.92	17664.8516	5.48
26	17392.3481	29038.8182	66.96	25823.9772	48.47	22375.0407	28.64	20712.8523	19.09
27	18247.5027	33784.7939	85.14	27736.4626	52.00	23176.2321	27.01	21284.4572	16.64
28	20939.7906			29056.0230	38.75	25436.6690	21.47	24166.7261	15.41
29	22007.5501			33791.1110	53.54	27952.4897	27.01	24914.5367	13.20
30	22239.2230					28780.6913	29.41	25271.9573	13.63

Table 7. Portal Arch Combined Model (FEC) Eigenvalues (continued).

Program	DSTAP	DSTAP		DSTAP		DSTAP	
Routine	BFMTHD	BFMTHD		BFMTHD		BFMTHD	
# of FE	0	5		7		10	
# of CE	3	2	Rel.	2	Rel.	2	Rel.
dof	6	18	Diff.	24	Diff.	33	Diff.
CPU sec	111.4100	267.8800	(%)	459.2200	(%)	888.9300	(%)
1	81.3702	81.3702	0.00	81.3702	0.00	81.3702	0.00
2	321.1035	321.1401	0.01	321.1131	0.00	321.1058	0.00
3	523.8114	523.8235	0.00	523.8147	0.00	523.8123	0.00
4	567.8924	567.9960	0.01	567.9196	0.00	567.8989	0.00
5	1146.9407	1148.5901	0.14	1147.3818	0.03	1147.0480	0.00
6	1401.0730	1401.3262	0.01	1401.1396	0.00	1401.0892	0.00
7	1620.6311	1622.4714	0.11	1621.1259	0.03	1620.7522	0.00
8	2459.1925	2472.8170	0.55	2463.0269	0.15	2460.1436	0.03
9	2905.0732	2907.9400	0.09	2905.8250	0.02	2905.2610	0.00
10	3063.0854	3077.1794	0.46	3067.0895	0.13	3064.0750	0.03
11	4278.1797	4341.5724	1.48	4298.3474	0.47	4283.3473	0.12
12	4768.5267	4780.8821	0.25	4771.4837	0.06	4769.2575	0.01
13	5121.2102	5155.6498	0.67	5136.5694	0.29	5125.1104	0.07
14	6573.0268	6899.4338	4.96	6637.5677	0.98	6590.6318	0.26
15	7280.4695	7288.1155	0.10	7292.0351	0.15	7283.3535	0.03
16	7527.8916	8020.8646	6.54	7576.7773	0.64	7540.9813	0.17
17	9328.1874	9834.1331	5.42	9503.3707	1.87	9380.6951	0.56
18	10119.1890	10199.2542	0.79	10145.7880	0.26	10125.5586	0.06
19	10525.9006	11215.6370	6.55	10589.1073	0.60	10550.1468	0.23
20	12032.0228	12594.0183	4.67	12456.7844	3.53	12117.5350	0.71
21	12917.9393	13141.2437	1.72	12942.7339	0.19	12949.1544	0.24
22	13260.7219	13365.8063	0.79	13339.8122	0.59	13272.3267	0.08
23	13655.9250	14063.0276	2.98	13675.3277	0.14	13679.9656	0.17
24	14416.1023	16479.6733	14.31	15388.0054	6.74	14531.1908	0.79
25	16746.9062	17477.1565	4.36	17221.0686	2.83	16915.6607	1.00
26	17392.3481	18379.4713	5.67	17531.2343	0.79	17425.9243	0.19
27	18247.5027	21378.3324	17.15	19708.9779	8.00	18436.7089	1.03
28	20939.7906	21660.1579	3.44	21416.0289	2.27	21179.2150	1.14
29	22007.5501	24576.8910	11.67	22334.3166	1.48	22082.7241	0.34
30	22239.2230	25162.4174	13.14	25084.0135	12.79	22543.3575	1.36

beams the FEM is able to extract midrange eigenvalues successfully. These models do require a larger number of degrees of freedom than the exact-element models. Nevertheless, there does not seem to be a need for an enhancement. But, the structure presented here is a small, simple structure. In practice, the structures analyzed are large and complex. This makes the addition of degrees of freedom almost impossible. In fact most large structures need to be solved through some form of dynamic substructuring. Sung and Nefske [3] have reported difficulties extracting acceptable eigenvalues when substructuring large models, and Thomas [29] has proven that the errors due to dynamic substructuring increase for larger number of degrees of freedom or for higher frequencies. The small test models presented here may have actually helped the FEM eigenvalue extraction. But even for these small models, if the solution times for the different models presented here are compared, then the advantage of the proposed enhancement is quickly seen.

It is more inexpensive to extract higher eigenvalues with exact-beam-element models. The solution time, for each model, reported in Tables 6 and 7 were divided by the number of acceptable eigenvalues obtained with that model. The result is shown in Fig. 12, which clearly shows that the cost of extracting eigenvalues within the set tolerance increases for larger finite element models. The data was input into the statistical package [62], which tested the 8 regression models presented earlier. From these models the better fits were provided by the models: (1) -12.915 + 0.348x and (2) $y=(0.00309)x^{(1.81562)}$. The residual norms for the linear model was less than one percent lower than the residual norm for the second model. Since the data in Fig. 12 seems to be ascending in a parabolic fashion, the second regression model is chosen over the first regression model. Based on this

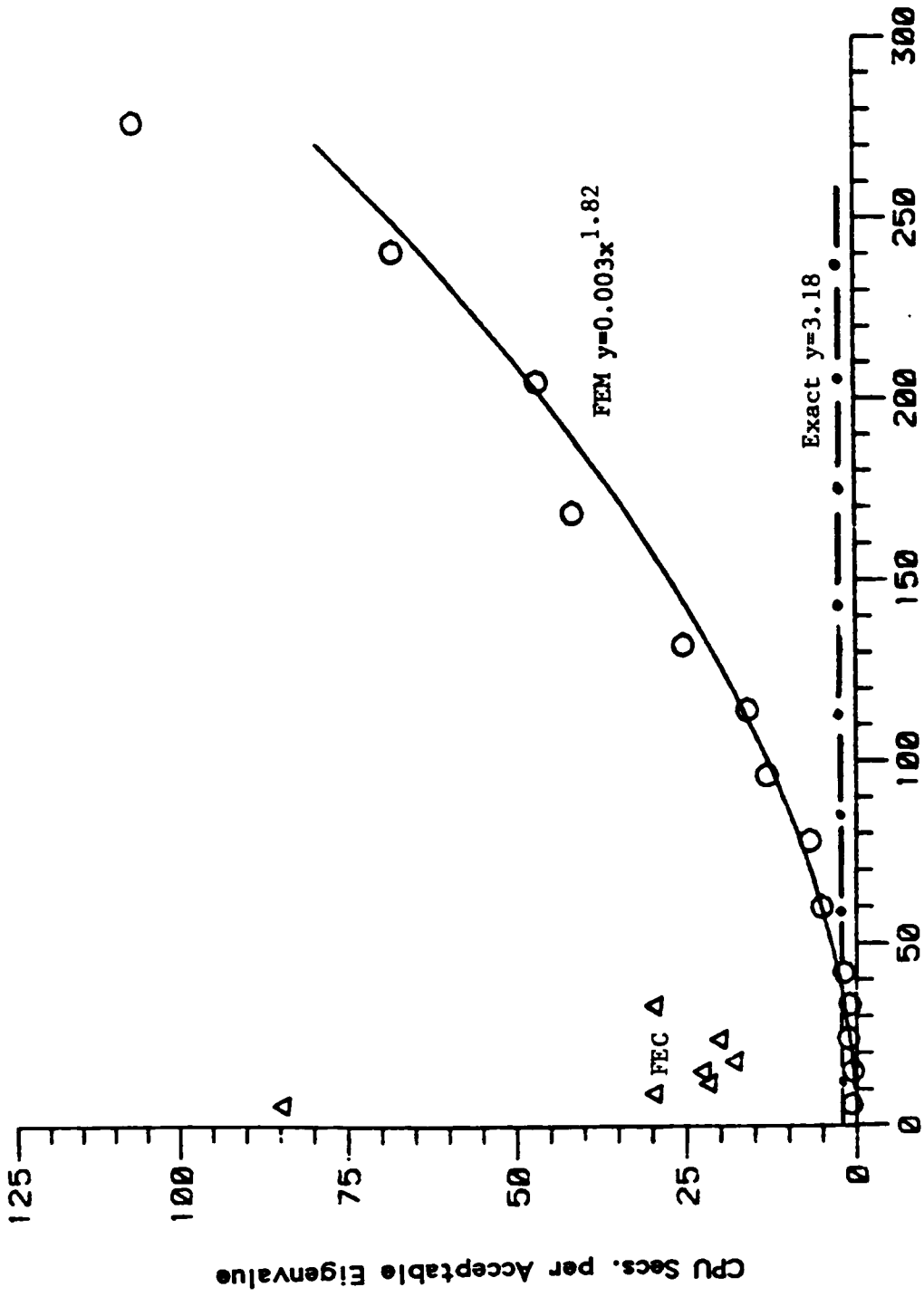


Figure 12. CPU seconds per Acceptable Eigenvalue versus Model Size for Portal Arch.

regression, the finite element models used should have few degrees of freedom to reduce the cost of the eigenvalue extraction. However, the only way that the first 30 acceptable eigenvalues of the Portal Arch can be obtained is by using at least 276 degrees of freedom. The cost of this extraction is 100.6 CPU sec per acceptable eigenvalue. On the other hand, the CPU cost per correct eigenvalue with a purely exact element model was only 3.18 CPU secs. per correct eigenvalue. This leads to the obvious conclusion that, for the Portal Arch, whenever the finite element model has more than 46 degrees of freedom then it less expensive to use an exact model. This break-even point corresponds to roughly 4 acceptable eigenvalues. In other words, when 4 or less acceptable eigenvalues are needed then it is more economical to use a FEM model, but when more that 4 acceptable eigenvalues are needed then it is more economical to use an exact model.

It is also more economical to extract the higher eigenvalues with combined models. Figure 12 has a set of points toward the left of the curve, labeled FEC. These points correspond to the combined FEM-exact models. These points are too scattered to perform a reliable regression analysis. But, neglecting the FEC model with only 1 FEM beam element, the remaining FEC models seem to be scattered about the 25 CPU sec per acceptable eigenvalue line. Using this last value for a break-even analysis, whenever more than 142 degrees of freedom are required in a finite element model, then it is more economical to use the combined model presented here. This corresponds to 15 acceptable eigenvalues. Therefore, the break-even point for the FEC models used here is 15 acceptable eigenvalues.

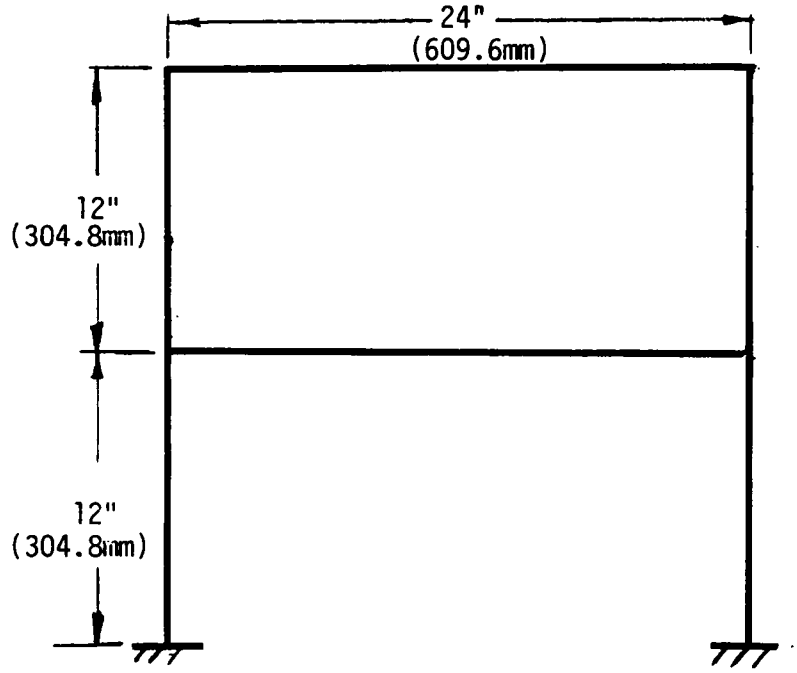
The results presented above for the FEC models are only for the FEC models in which the legs of the Portal Arch are modeled with exact-beam

elements. Obviously, the results of this analysis would change if the Portal Arch is modeled with a different combination of FEM-elements and exact-elements. Thus, it is impossible to escape modeling considerations when studying the FEC models. This implies that depending on an analyst's choice of location and quantity of exact-elements the cost versus accuracy comparison may vary. It seems natural that since the eigenvector of the exact elements can be reconstructed with almost any degree of precision desired, the analyst should tend to place these exact elements in areas where much detail is required. That is, replace as many finite elements as possible with a single exact displacement element.

C. Verdeel Truss

The final example is presented in Fig. 13. In the literature, this structure is sometimes called a Verdeel Truss. This name is really a misnomer because the structure is made up of beams rather than trusses. The beams in this example have the same cross-section dimensions and material properties as the beams in the previous two examples. The first objective of this example is to verify the conclusions arrived at in the Portal Arch example. This leads to the following tasks, (1) determine whether finite element models can obtain a certain number of eigenvalues within the specified tolerance (2) ascertain whether the eigenvalue extraction can be improved by using a combination of FEM elements and exact elements. The second objective in this example is to compare solution efficiency with a commercial code.

The first model created for this example was the exact-beam model. This model was made up of 6 exact beam elements, with a total of 12 degrees of



Material Properties:

E = 30.0 Mpsi (207. GPa)
G = 11.6 Mpsi (79.9 GPa)
ρ = 0.282 lb_f/in^3 (7.81 mgr/mm^3)

Figure 13. Schematic of the Beam for Example C.

freedom. As in the previous example, this model was analyzed with DSTAP, both with and without the shear deformation and rotatory inertia effects. These results are presented in Table 8, along with the relative difference between the eigenvalues. According to the results in Table 8, the shear deformation and rotatory effects do not have a noticeable impact on the computed eigenvalues in the frequency range specified to DSTAP.

The reconstructed eigenvectors from the analysis with shear deformation and rotatory inertia are presented in Fig. 14. As was the case with the Portal Arch, this catalog of the first 48 mode shapes for the Verdeel Truss is believed to be unique. This catalogue further shows that the incorporation of the TMM into the eigenvector extraction is more desirable than the approximate methods for mode shape determination. For example, Hopper and Williams [31] tested their approximate mode extraction technique on a Verdeel Truss similar to the one for this example. With their approximation they obtained a possible 6^{th} and 7^{th} modes, as shown in Fig. 15. Comparing their approximated mode shapes with the catalog of mode shapes in Fig. 16, it is evident that their mode shapes resemble the 7^{th} and 9^{th} mode shapes more closely than the 6^{th} and 7^{th}. However, even the 7^{th} and 9^{th} modes do not exhibit the relatively stationary substructure as depicted in Fig. 15.

Having determined from Table 8 that shear deformation and rotatory inertia are minor factors in the first 48 eigenvalues, the number of desired acceptable eigenvalues was set at 35. This is an increase from the 30 acceptable eigenvalues sought in the Portal Arch example. The criteria for acceptable eigenvalues remains the same for this example; that is, all lower eigenvalues up to the first occurrence of a 5 per cent relative difference were are acceptable. The relative difference was computed for the FEM and

Table 8. Verdeel Truss Exact-Element Model Eigenvalues, With and Without Shear Deformation and Rotatory Inertia.

Program Routine	DSTAP BFMTHD	DSTAP BFMTHD		
G(psi)=	1.1500E7	infinite		
iy(in)=	0.0721	zero	Absolute	Relative
dof	12	12	Difference	Difference
CPU sec	260.4600	258.1900	(rad/sec)	(%)
1	107.1658	107.1966	0.03080	0.02874
2	377.2853	377.4589	0.17360	0.04601
3	397.0859	397.2549	0.16900	0.04256
4	475.4492	475.7334	0.28420	0.05977
5	1098.4206	1099.2899	0.86930	0.07914
6	1314.7449	1316.2433	1.49840	0.11396
7	1502.0990	1504.0376	1.93860	0.12905
8	1908.0520	1911.6293	3.57730	0.18748
9	2057.2227	2061.4500	4.22730	0.20548
10	2441.9228	2447.5039	5.58110	0.22855
11	2688.9983	2695.0238	6.02550	0.22407
12	2896.4220	2903.7459	7.32390	0.25286
13	4159.4145	4171.0937	11.67920	0.28078
14	4602.2060	4618.2581	16.05210	0.34879
15	4924.9877	4943.6005	18.61280	0.37792
16	5585.9694	5612.5382	26.56880	0.47563
17	5855.2977	5885.1186	29.82090	0.50929
18	6370.0693	6405.0077	34.93840	0.54847
19	6913.5078	6949.5716	36.06380	0.52164
20	7187.3031	7227.1974	39.89430	0.55506
21	9172.4259	9227.6363	55.21040	0.60191
22	9588.1739	9648.5681	60.39420	0.62988
23	10272.7812	10349.2749	76.49370	0.74462
24	11242.8782	11343.0263	100.14810	0.89076
25	11458.0602	11550.2063	92.14610	0.80420
26	11873.7397	11931.5463	57.80660	0.48684
27	12139.6204	12249.1092	109.48880	0.90191
28	12828.1377	12862.2577	34.12000	0.26597
29	13515.4265	13650.0872	134.66070	0.99634
30	14101.0285	14190.6086	89.58010	0.63527
31	16419.5340	16589.2026	169.66860	1.03333
32	16950.8632	17151.7868	200.92360	1.18532
33	17306.4635	17505.5446	199.08110	1.15032
34	18554.3350	18789.5466	235.21160	1.26769
35	18960.2803	19224.3217	264.04140	1.39260
36	19861.8067	20167.4736	305.66690	1.53896
37	20975.5233	21260.2766	284.75330	1.35755
38	21985.1230	22315.8535	330.73050	1.50433
39	23303.4247	23478.5106	175.08590	0.75133

Table 8. Verdeel Truss Exact-Element Model Eigenvalues, With and Without Shear Deformation and Rotatory Inertia (continued).

Program Routine	DSTAP BFMTHD	DSTAP BFMTHD		
G(psi)=	1.1500E7	infinite		
iy(in)=	0.0721	zero	Absolute	Relative
dof	12	12	Difference	Difference
CPU sec	260.4600	258.1900	(rad/sec)	(%)
40	24899.5450	25300.6025	401.05750	1.61070
41	25205.4059	25374.8936	169.48770	0.67242
42	26284.5061	26753.6289	469.12280	1.78478
43	27717.9006	28142.0960	424.19540	1.53040
44	27992.8458	28562.4642	569.61840	2.03487
45	29192.3830	29826.5806	634.19760	2.17247
46	30082.2357	30637.8194	555.58370	1.84688
47	31532.3081	32204.5491	672.24100	2.13191
48	31750.5550	32432.3189	681.76390	2.14725

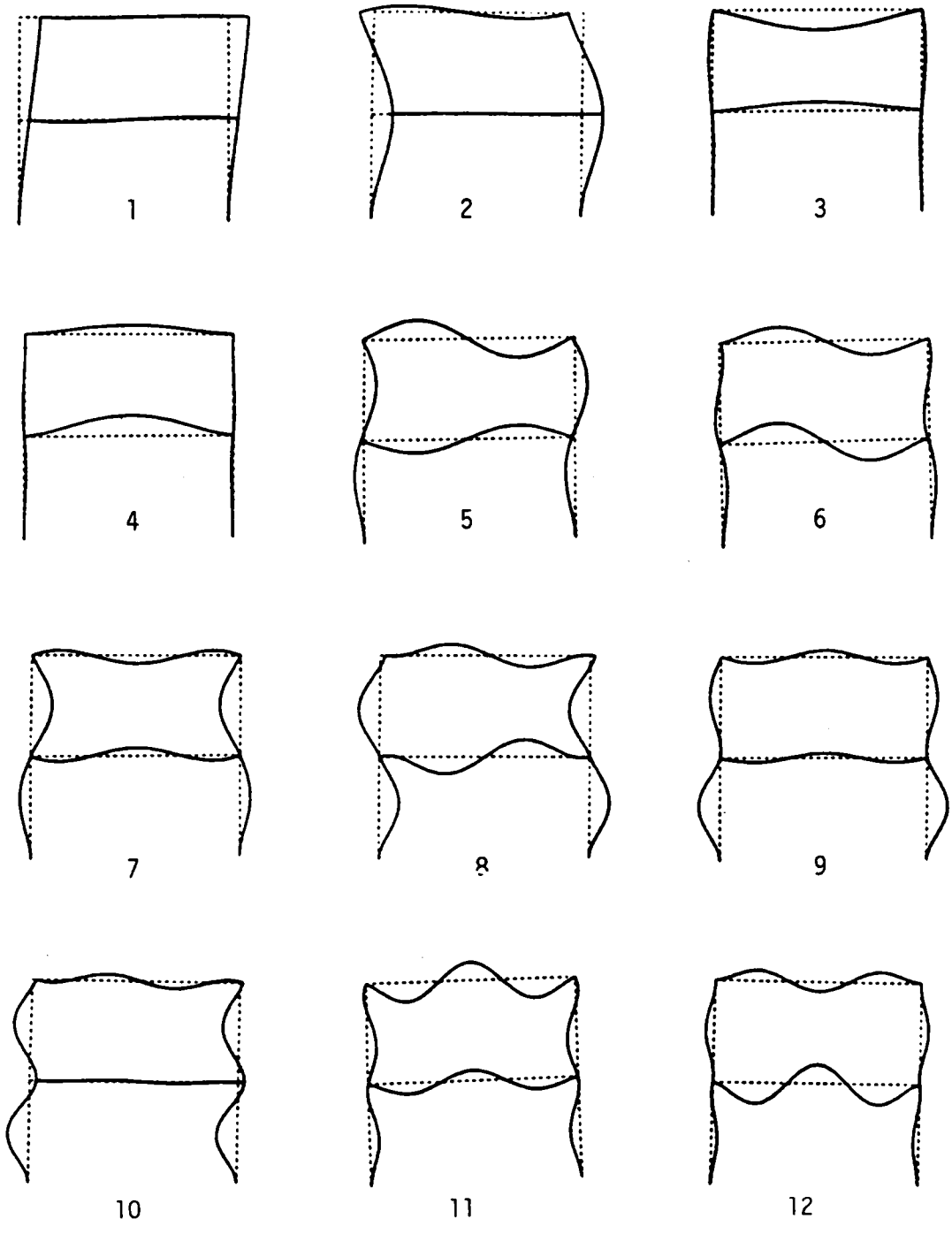


Figure 14. Mode Shapes for the Verdeel Truss Model with Shear Deformation and Rotatory Inertia.

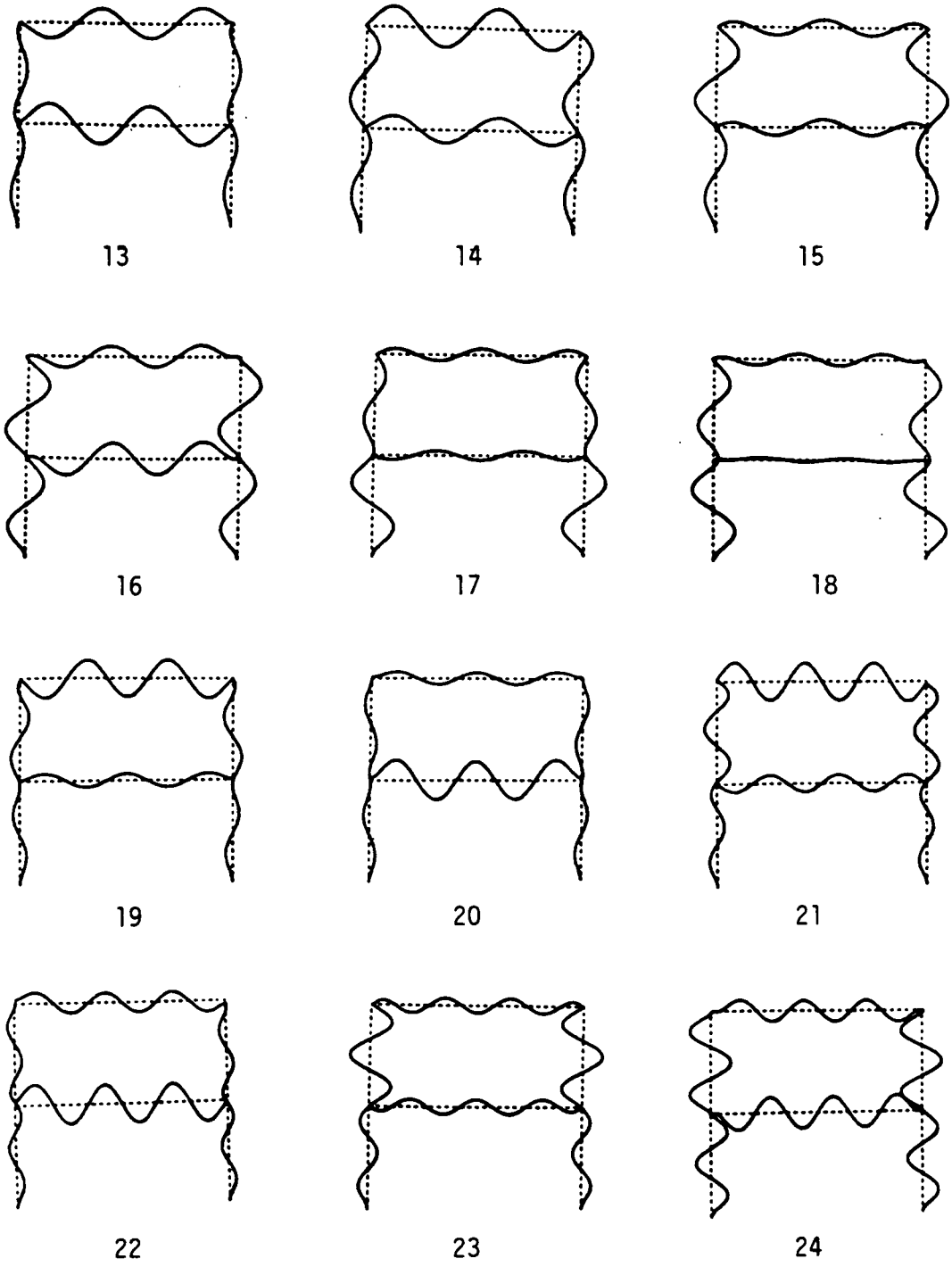


Figure 14. Mode Shapes for the Verdeel Truss Model with Shear Deformation and Rotatory Inertia (continued).

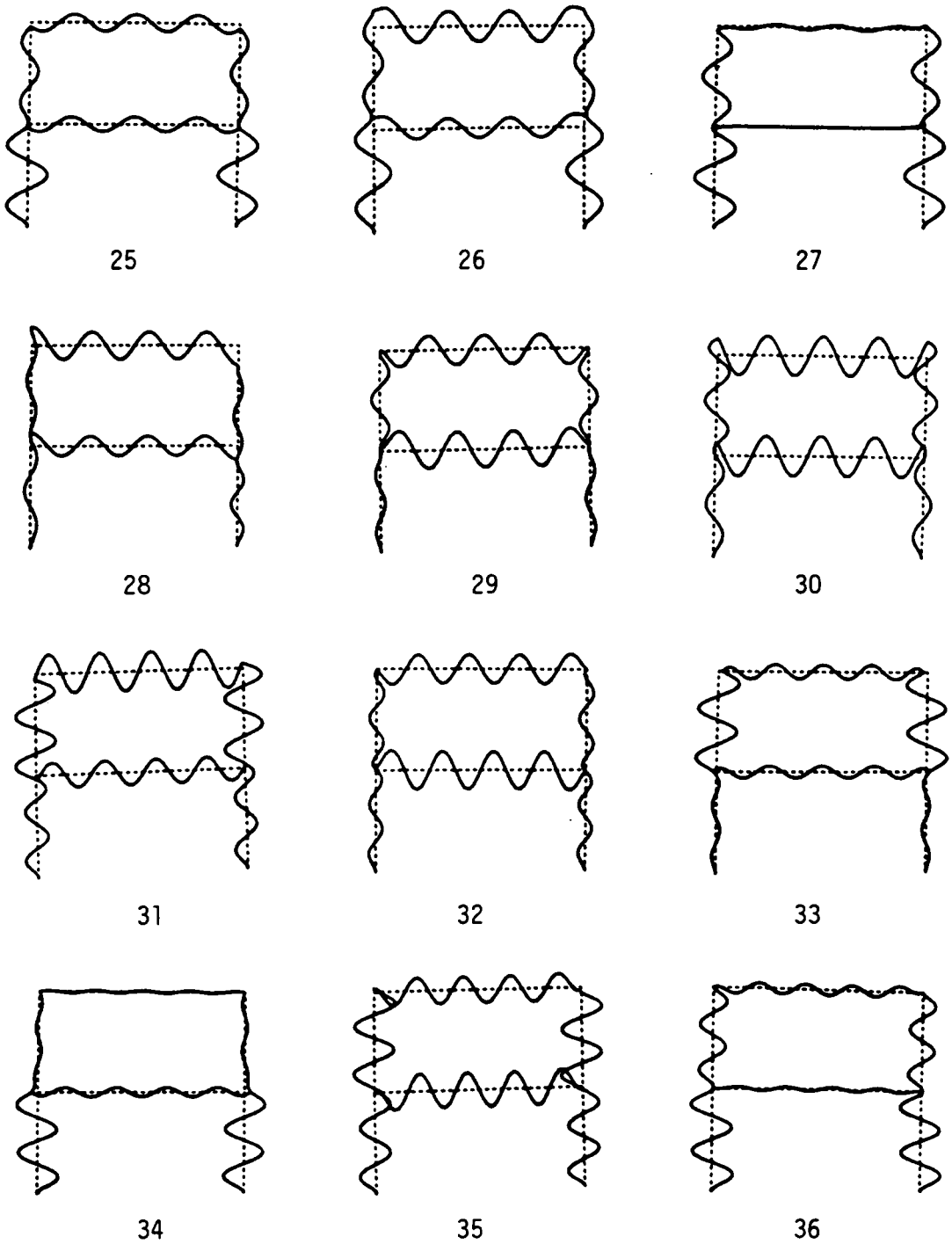


Figure 14. Mode Shapes for the Verdeel Truss Model with Shear Deformation and Rotatory Inertia (continued).

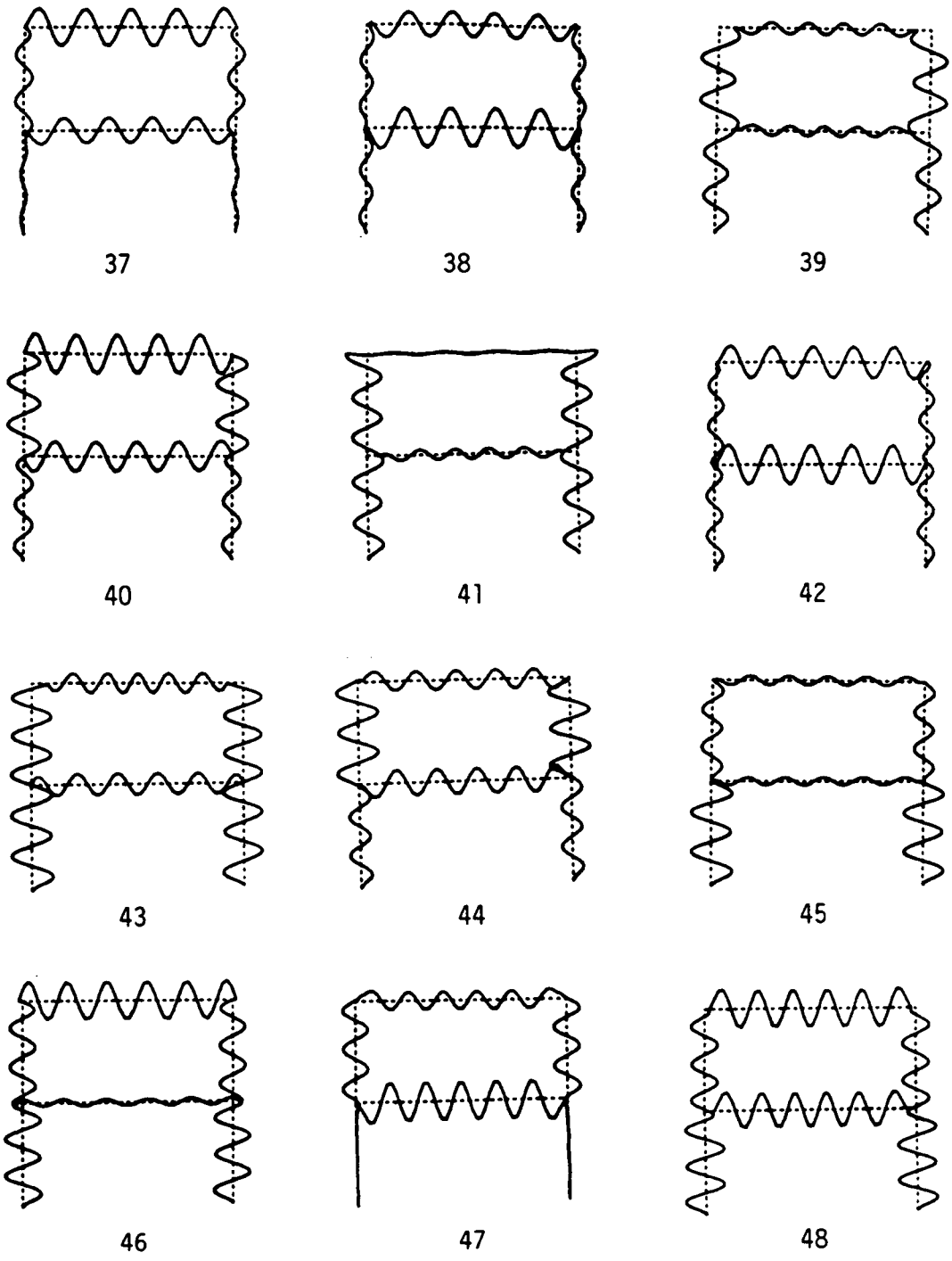


Figure 14. Mode Shapes for the Verdeel Truss Model with Shear Deformation and Rotatory Inertia (continued).

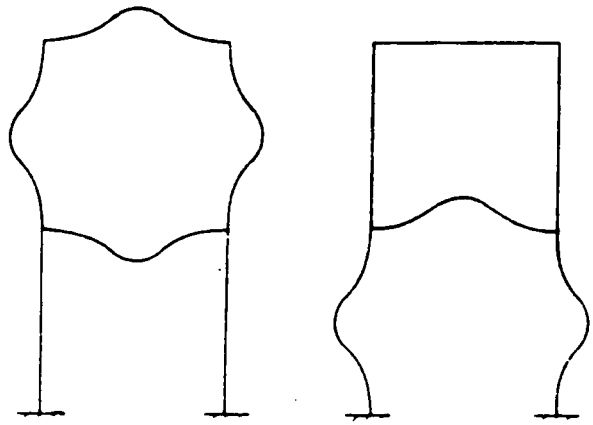


Figure 15. Proposed 6^{th} and 7^{th} Modes (After Ref. [31]).

FEC models by comparing the approximate eigenvalues to the exact eigenvalues.

Several finite element models were created for DSTAP. In each successive model, more degrees of freedom were included until approximately 35 acceptable eigenvalues were extracted. The results from these models are included in Tables 9 and 10. In both of these tables the eigenvalues from Table 8 are included for comparison. Table 9 contains the eigenvalues computed with subroutine EIGENR. This routine uses the positive definite property of the finite element matrices to optimize the eigenvalue extraction. If numerical round off makes the matrices nonpositive definite then EIGENR extracts wrong eigenvalues. For example, the first model in Table 9 had 18 degrees of freedom and gave a first eigenvalue of 107.8450 rad/sec. As the number of degrees of freedom in the succesive models increased, the estimate of the first eigenvalue approached the exact eigenvalue, 107.1966 rad/sec, from above. When a model of 114 degrees of freedom was used the estimate of the first eigenvalue dropped to 75.01 rad/sec. Along with this unexpected result, EIGENR sent a "non-positive definite matrix" message to the terminal screen. Thus, the model with 85 degrees of freedom was the largest model solved satisfactorily with EIGENR.

Table 10 presents the finite element model eigenvalues computed using subroutine EIGZF, from the IMSL library [60]. All the models prepared for routine EIGENR were rerun using routine EIGZF. The numerical difficulty encountered in the solution with routine EIGENR manifested itself in EIGZF as complex conjugate pair of eigenvalues. The real component of this complex pair of eigenvalues was zero. This pair of eigenvalues was easily discarded because the matrices were known to be positive-definite at the start of the eigenvalue extraction, therefore, all eigenvalues are guaranteed to be real,

Table 9. DSTAP FEM Eigenvalues using Subroutine EIGENR.

Program Routine	DSTAP BFMTHD	DSTAP EIGENR		DSTAP EIGENR		DSTAP EIGENR		DSTAP EIGENR	
# of FE	0	16	Rel.	24	Rel.	32	Rel.	40	Rel.
# of CE	6	0	Diff	0	Diff	0	Diff	0	Diff
dof	12	42	(%)	66	(%)	90	(%)	114	(%)
CPU sec	258.1900	9.6000		32.8000		87.5500		181.1900	
1	107.1966	107.4856	0.26	107.3378	0.13	107.2787	0.07	75.0090	-30.0
2	377.4589	384.9831	1.99	381.4011	1.04	379.7971	0.61	126.3616	-66.5
3	397.2549	398.9907	0.43	398.1451	0.22	397.7827	0.13	241.3093	-39.2
4	475.7334	476.5404	0.16	476.1578	0.08	475.9911	0.05	368.1057	-22.6
5	1099.2899	1133.0545	3.07	1118.4496	1.74	1111.0391	1.06	544.1080	-50.5
6	1316.2433	1330.7447	1.10	1325.2077	0.68	1321.8825	0.42	719.4389	-45.3
7	1504.0376	1616.6789	7.48	1574.8122	4.70	1548.4618	2.95	1029.3087	-31.5
8	1911.6293	2005.2588	4.89	1985.2447	3.85	1960.5662	2.55	1512.7147	-20.8
9	2061.4500	2120.0610	2.84	2144.6072	4.03	2123.2459	2.99	1775.1528	-13.8
10	2447.5039	2511.1180	2.59	2581.4980	5.47	2550.3930	4.20	1995.6209	-18.4
11	2695.0238	2743.6629	1.80	2730.1718	1.30	2721.7413	0.99	2178.8817	-19.1
12	2903.7459	2935.6575	1.09	2928.6360	0.85	2921.3386	0.60	2287.0080	-21.2
13	4171.0937	4631.6685	11.04	4331.7611	3.85	4294.4801	2.95	2595.0512	-37.7
14	4618.2581	5096.1212	10.34	4693.2228	1.62	4676.9491	1.27	3435.0934	-25.6
15	4943.6005	5784.8380	17.01	5409.5455	9.42	5363.6059	8.49	3623.2252	-26.7
16	5612.5382	6729.2361	19.89	5991.6522	6.75	6015.3827	7.17	4572.7918	-18.5
17	5885.1186	6915.3330	17.50	6148.9476	4.48	6317.9385	7.35	5386.8732	-8.46
18	6405.0077	7789.5899	21.61	6688.0172	4.41	6938.3559	8.32	5738.4236	-10.4
19	6949.5716	8800.3154	26.63	7132.5681	2.63	7120.1827	2.45	6155.0409	-11.4
20	7227.1974	9260.6072	28.13	7403.3654	2.43	7365.4795	1.91	6466.8575	-10.5
21	9227.6363	11073.9428	20.00	10249.4232	11.07	9563.6520	3.64	6560.2610	-28.9
22	9648.5681	11843.7890	22.75	10628.5455	10.15	9888.7795	2.48	7035.5089	-27.0
23	10349.2749	12616.8334	21.91	12107.8912	16.99	11204.5037	8.26	9067.5834	-12.3
24	11343.0263	13346.3152	17.66	12260.3361	8.08	11869.7849	4.64	10182.5401	-10.2
25	11550.2063	14956.1695	29.48	12982.2507	12.39	11975.8197	3.68	10673.3853	-7.59
26	11931.5463	17256.4841	44.62	13152.4545	10.23	12180.5677	2.08	11672.7721	-2.16
27	12249.1092	18874.2703	54.08	13575.9913	10.83	12770.5564	4.25	12291.0977	0.34
28	12862.2577	20029.7633	55.72	14309.3630	11.25	12996.4916	1.04	12389.7607	-3.67
29	13650.0872	23051.6905	68.87	15450.8103	13.19	13894.0462	1.78	12636.9612	-7.42
30	14190.6086	29760.7079	109.7	15948.4922	12.38	14487.2679	2.09	12858.0201	-9.39
31	16589.2026	37986.9133	128.9	17590.0374	6.03	18531.0488	11.70	13291.4433	-19.8
32	17151.7868	38698.5476	125.6	19778.6478	15.31	18781.6279	9.50	13512.4330	-21.2
33	17505.5446	44214.9083	152.5	20037.5294	14.46	20938.3109	19.60	15428.5609	-11.8
34	18789.5466	59487.6167	216.5	22342.6451	18.90	21154.9665	12.58	16002.2458	-14.8
35	19224.3217	68626.6892	256.9	22891.7210	19.07	21914.9138	13.99	16673.8788	-13.2
36	20167.4736	70492.6914	249.5	24469.6868	21.33	23554.1929	16.79	17307.3786	-14.1

Table 10. DSTAP FEM Eigenvalues using Subroutine EIGZF.

Program	DSTAP	DSTAP		DSTAP		DSTAP		DSTAP	
Routine	BFMTHD	IMSL-EIGZF		IMSL-EIGZF		EIGENR		IMSL-EIGZF	
# of FE	0	8	Rel.	16	Rel.	24	Rel.	32	Rel.
# of CE	6	0	Diff	0	Diff	0	Diff	0	Diff
dof	12	18	(%)	42	(%)	66	(%)	90	(%)
CPU sec	258.1900	2.1100		14.4700		50.1300		123.1400	
1	107.1966	107.8450	0.60	107.4856	0.26	107.3378	0.13	107.2787	0.07
2	377.4589	383.5772	1.62	384.9831	1.99	381.4011	1.04	379.7971	0.61
3	397.2549	403.9738	1.69	398.9907	0.43	398.1451	0.22	397.7827	0.13
4	475.7334	481.8348	1.28	476.5404	0.16	476.1578	0.08	475.9911	0.05
5	1099.2899	1216.4625	10.65	1133.0545	3.07	1118.4496	1.74	1111.0391	1.06
6	1316.2433	1572.5912	19.47	1330.7447	1.10	1325.2077	0.68	1321.8825	0.42
7	1504.0376	1881.1082	25.07	1616.6789	7.48	1574.8122	4.70	1548.4618	2.95
8	1911.6293	2874.2904	50.35	2005.2588	4.89	1985.2447	3.85	1960.5662	2.55
9	2061.4500	3206.8254	55.56	2120.0610	2.84	2144.6072	4.03	2123.2459	2.99
10	2447.5039	4568.6860	86.66	2511.1180	2.59	2581.4980	5.47	2550.3930	4.20
11	2695.0238	11825.9291	338.8	2743.6629	1.80	2730.1718	1.30	2721.7413	0.99
12	2903.7459	11959.7983	311.8	2935.6575	1.09	2928.6360	0.85	2921.3386	0.60
13	4171.0937	17215.8794	312.7	4631.6685	11.04	4331.7611	3.85	4294.4801	2.95
14	4618.2581	25982.8561	462.6	5096.1212	10.34	4693.2228	1.62	4676.9491	1.27
15	4943.6005	35374.1146	615.5	5784.8380	17.01	5409.5455	9.42	5363.6059	8.49
16	5612.5382	35809.8707	538.0	6729.2361	19.89	5991.6522	6.75	6015.3827	7.17
17	5885.1186	41031.4551	597.2	6915.3330	17.50	6148.9476	4.48	6317.9385	7.35
18	6405.0077	59373.5434	826.9	7789.5899	21.61	6688.0172	4.41	6938.3559	8.32
19	6949.5716			8800.3154	26.63	7132.5681	2.63	7120.1827	2.45
20	7227.1974			9260.6072	28.13	7403.3654	2.43	7365.4795	1.91
21	9227.6363			11073.9428	20.00	10249.4232	11.07	9563.6520	3.64
22	9648.5681			11843.7890	22.75	10628.5455	10.15	9888.7795	2.48
23	10349.2749			12616.8334	21.91	12107.8912	16.99	11204.5037	8.26
24	11343.0263			13346.3152	17.66	12260.3361	8.08	11869.7849	4.64
25	11550.2063			14956.1695	29.48	12982.2507	12.39	11975.8197	3.68
26	11931.5463			17256.4841	44.62	13152.4545	10.23	12180.5677	2.08
27	12249.1092			18874.2703	54.08	13575.9913	10.83	12770.5564	4.25
28	12862.2577			20029.7633	55.72	14309.3630	11.25	12996.4916	1.04
29	13650.0872			23051.6905	68.87	15450.8103	13.19	13894.0462	1.78
30	14190.6086			29760.7079	109.7	15948.4922	12.38	14487.2679	2.09
31	16589.2026			37986.9133	128.9	17590.0374	6.03	18531.0488	11.70
32	17151.7868			38698.5476	125.6	19778.6478	15.31	18781.6279	9.50
33	17505.5446			44214.9083	152.5	20037.5294	14.46	20938.3109	19.60
34	18789.5466			59487.6167	216.5	22342.6451	18.90	21154.9665	12.58
35	19224.3217			68626.6892	256.9	22891.7210	19.07	21914.9138	13.99

Table 10. DSTAP FEM Eigenvalues using Subroutine EIGZF (continued).

Program	DSTAP	DSTAP		DSTAP		DSTAP		DSTAP	
Routine	BFMTHD	IMSL-EIGZF		IMSL-EIGZF		IMSL-EIGZF		IMSL-EIGZF	
# of FE	0	40	Rel.	48	Rel.	56	Rel.	64	Rel.
# of CE	6	0	Diff	0	Diff	0	Diff	0	Diff
dof	12	114	(%)	138	(%)	162	(%)	186	(%)
CPU sec	258.1900	246.6900		417.1400		673.5600		991.1400	
		1546990.4i		1627377.5i		1916267.3i		2305931.7i	
1	107.1966	107.2499	0.04	107.2339	0.03	107.2241	0.02	107.2177	0.01
2	377.4589	378.9914	0.40	378.5368	0.28	378.2569	0.21	378.0728	0.16
3	397.2549	397.6011	0.08	397.4986	0.06	397.4354	0.04	397.3938	0.03
4	475.7334	475.9046	0.03	475.8549	0.02	475.8238	0.01	475.8032	0.01
5	1099.2899	1107.1220	0.71	1104.8520	0.50	1103.4322	0.37	1102.4896	0.29
6	1316.2433	1320.0532	0.28	1318.9703	0.20	1318.2843	0.15	1317.8251	0.12
7	1504.0376	1533.8831	1.98	1525.3036	1.41	1519.9021	1.05	1516.3036	0.81
8	1911.6293	1945.2593	1.75	1935.8621	1.26	1929.8231	0.95	1925.7523	0.73
9	2061.4500	2105.8834	2.15	2094.2203	1.58	2086.3904	1.20	2080.9779	0.94
10	2447.5039	2522.0126	3.04	2502.5530	2.24	2489.4150	1.71	2480.3184	1.34
11	2695.0238	2714.5940	0.72	2709.6173	0.54	2706.2075	0.41	2703.8210	0.32
12	2903.7459	2916.3195	0.43	2913.0348	0.31	2910.8352	0.24	2909.3110	0.19
13	4171.0937	4262.8783	2.20	4240.7399	1.66	4225.2360	1.29	4214.1522	1.03
14	4618.2581	4660.9758	0.92	4650.3282	0.69	4643.0660	0.53	4637.9403	0.42
15	4943.6005	5260.3633	6.40	5182.2843	4.82	5127.3678	3.71	5088.4923	2.93
16	5612.5382	5923.5402	5.54	5847.7285	4.19	5793.4184	3.22	5754.8511	2.53
17	5885.1186	6254.5292	6.27	6180.7491	5.02	6120.7806	4.00	6075.0129	3.22
18	6405.0077	6879.5840	7.40	6788.3823	5.98	6710.9041	4.77	6651.2993	3.84
19	6949.5716	7085.7846	1.96	7054.4850	1.50	7031.1003	1.17	7014.1641	0.92
20	7227.1974	7327.7611	1.39	7301.7314	1.03	7284.1264	0.78	7271.9081	0.61
21	9227.6363	9511.6938	3.07	9460.2752	2.52	9420.0138	2.08	9388.3639	1.74
22	9648.5681	9845.0794	2.03	9798.5572	1.55	9765.5803	1.21	9741.9966	0.96
23	10349.2749	11375.2562	9.91	11229.2676	8.50	11072.2645	6.98	10941.6468	5.72
24	11343.0263	11884.9691	4.77	11840.3856	4.38	11811.6767	4.13	11790.6518	3.94
25	11550.2063	12325.8188	6.71	12205.2446	5.67	12048.4152	4.31	11915.4551	3.16
26	11931.5463	12744.4040	6.81	12734.2100	6.72	12606.9449	5.66	12475.6058	4.55
27	12249.1092	12887.2144	5.20	12835.7679	4.78	12806.9739	4.55	12782.5319	4.35
28	12862.2577	13508.9993	5.02	13499.7098	4.95	13361.4756	3.88	13219.6200	2.77
29	13650.0872	13991.7059	2.50	13296.7789	-2.58	13865.9530	1.58	13820.2671	1.24
30	14190.6086	14476.3084	2.01	14425.3127	1.65	14382.0484	1.34	14348.7028	1.11
31	16589.2026	17299.0867	4.27	17314.2871	4.37	17236.5947	3.90	17162.2227	3.45
32	17151.7868	17557.0282	2.36	17554.8012	2.34	17475.5471	1.88	17411.0941	1.51
33	17505.5446	18775.8067	7.25	19230.1172	9.85	19087.8445	9.03	18875.5749	7.82
34	18789.5466	19535.7594	3.97	20483.0307	9.01	20460.6644	8.89	20286.0626	7.96
35	19224.3217	19829.5485	3.14	20810.5726	8.25	20792.2777	8.15	20593.1358	7.12

Table 10. DSTAP FEM Eigenvalues using Subroutine EIGZF (continued).

Program	DSTAP	DSTAP		DSTAP		DSTAP	
Routine	BFMTHD	IMSL-EIGZF		IMSL-EIGZF		IMSL-EIGZF	
# of FE	0	72	Rel.	80	Rel.	120	Rel.
# of CE	6	0	Diff	0	Diff	0	Diff
dof	12	210	(%)	234	(%)	354	(%)
CPU sec	258.1900	1404.4500		1921.0700		7150.2700	
		2772981.0i		3308629.2i		6912257.7i	
1	107.1966	107.2133	0.01	107.2102	0.01	107.2026	0.00
2	377.4589	377.9456	0.12	377.8541	0.10	377.6355	0.04
3	397.2549	397.3651	0.02	397.3444	0.02	397.2949	0.01
4	475.7334	475.7889	0.01	475.7785	0.00	475.7537	0.00
5	1099.2899	1101.8335	0.23	1101.3592	0.18	1100.2192	0.08
6	1316.2433	1317.5037	0.09	1317.2704	0.07	1316.7064	0.03
7	1504.0376	1513.7943	0.64	1511.9781	0.52	1507.6064	0.23
8	1911.6293	1922.8933	0.58	1920.8137	0.48	1915.7751	0.21
9	2061.4500	2077.1161	0.75	2074.2766	0.62	2067.2961	0.28
10	2447.5039	2473.8248	1.07	2469.0504	0.88	2457.3182	0.40
11	2695.0238	2702.1039	0.26	2700.8342	0.21	2697.6869	0.09
12	2903.7459	2908.2198	0.15	2907.4151	0.12	2905.4259	0.05
13	4171.0937	4206.0367	0.83	4199.9529	0.69	4184.5396	0.32
14	4618.2581	4634.2120	0.34	4631.4265	0.28	4624.3907	0.13
15	4943.6005	5060.3578	2.36	5039.4893	1.93	4987.5977	0.88
16	5612.5382	5726.9767	2.03	5706.3481	1.67	5655.3435	0.76
17	5885.1186	6040.3816	2.63	6013.9269	2.18	5945.6013	1.02
18	6405.0077	6605.9939	3.13	6571.5050	2.59	6482.8325	1.21
19	6949.5716	7001.7803	0.75	6992.5403	0.61	6969.3959	0.28
20	7227.1974	7263.1613	0.49	7256.7089	0.40	7240.7410	0.18
21	9227.6363	9363.2314	1.46	9343.0929	1.25	9286.0289	0.63
22	9648.5681	9724.6783	0.78	9711.6359	0.65	9678.2511	0.30
23	10349.2749	10838.5634	4.72	10757.9269	3.94	10543.7887	1.87
24	11343.0263	11773.0047	3.79	11730.4435	3.41	11522.5282	1.58
25	11550.2063	11811.2718	2.26	11757.2097	1.79	11687.8872	1.19
26	11931.5463	12365.4852	3.63	12278.2165	2.90	12062.4855	1.09
27	12249.1092	12753.2764	4.11	12716.4597	3.81	12517.6500	2.19
28	12862.2577	13106.2469	1.89	13026.3630	1.27	12900.4189	0.29
29	13650.0872	13786.7987	1.00	13761.9863	0.81	13700.9249	0.37
30	14190.6086	14323.0158	0.93	14302.9126	0.79	14247.4502	0.40
31	16589.2026	17095.9577	3.05	17037.5253	2.70	16841.1833	1.51
32	17151.7868	17362.7603	1.23	17326.4639	1.01	17234.8508	0.48
33	17505.5446	18676.9942	6.69	18507.3169	5.72	18007.5892	2.86
34	18789.5466	20097.1826	6.95	19925.0038	6.04	19380.0982	3.14
35	19224.3217	20382.4027	6.02	20199.1740	5.07	19681.3101	2.37

positive, and nonzero. Eliminating these eigenvalues from the solution set, the eigenvalues computed with EIGZF exhibit the expected behavior. The higher eigenvalues can be extracted satisfactorily, provided the model has a sufficient number of degrees of freedom.

Figure 16 presents graphically the results from Table 10. The data points in this figure were used in a regression analysis. The analysis gave the curve fit,

$$y=(0.13095)x, \tag{32}$$

where x is again the number of degrees of freedom and y is the number of acceptable eigenvalues. The number of necessary active nodes per acceptable eigenvalue may be estimated by using Eq. (32) in reverse fashion. Approximately 8 degrees of freedom are required for each acceptable eigenvalue, which corresponds to approximately 3 active nodes per acceptable eigenvalue. Figure 16 also contains two more curves, one for the results obtained with FEM models in SUPERB and another one for combined models in DSTAP. These curves are discussed below.

Some of the smaller models used with EIGENR and EIGZF were analyzed with the SUPERB finite element program [15]. The results from SUPERB are presented in Table 11 and shown graphically in Fig. 16. These results show that the FEM-beam element in SUPERB is better suited for extracting higher eigenvalues than the FEM-beam element added to DSTAP. The regression line for the SUPERB models is of the form,

$$y = -3.293 + (0.37024)x$$

The higher slope of this regression line indicates that SUPERB requires fewer degrees of freedom to extract the desired number of acceptable eigenvalues. This result is quite surprising in light of the fixed-fixed example, where the

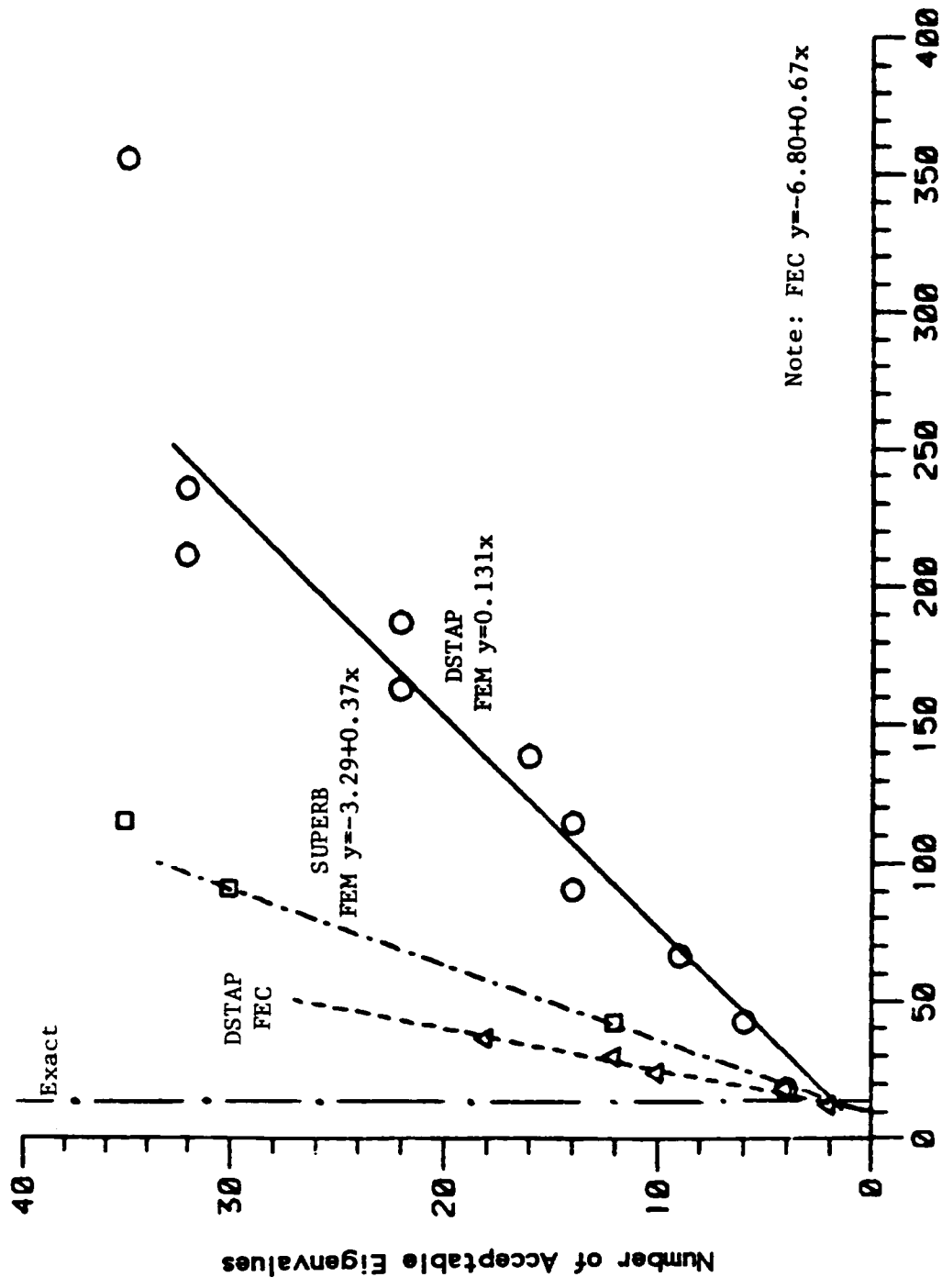


Figure 16. Number of Acceptable Eigenvalues versus Degrees of Freedom for the Verdeel Truss.

Table 11. SUPERB FEM Eigenvalues.

Program	DSTAP	SUPERB		SUPERB		SUPERB		SUPERB	
Routine	BFMTHD	JACOBI		JACOBI		JACOBI		JACOBI	
# of FE	0	16	Rel.	32	Rel.	40	Rel.	48	Rel.
# of CE	6	0	Diff	0	Diff	0	Diff	0	Diff
dof	12	42	(%)	90	(%)	114	(%)	138	(%)
CPU sec	258.1900	48.2700		351.7500		700.7000		1347.9000	
1	107.1966	107.1974	0.00	107.1974	0.00	107.1974	0.00	107.1974	0.00
2	377.4589	377.5189	0.01	377.4497	-0.00	377.4497	-0.00	377.4435	-0.00
3	397.2549	397.4931	0.05	397.2543	-0.00	397.2418	-0.00	397.2732	0.00
4	475.7334	476.1523	0.08	475.7376	0.00	475.7188	-0.00	475.7125	-0.00
5	1099.2899	1104.0813	0.43	1099.4317	0.01	1099.2432	-0.00	1099.1804	-0.00
6	1316.2433	1324.6839	0.64	1316.5786	0.02	1316.2016	-0.00	1316.0759	-0.01
7	1504.0376	1516.6352	0.83	1504.5715	0.03	1504.0689	0.00	1503.8804	-0.01
8	1911.6293	1934.6555	1.20	1912.9157	0.06	1911.8476	0.01	1911.4706	-0.00
9	2061.4500	2091.2954	1.44	2063.2095	0.08	2061.8900	0.02	2061.3874	-0.00
10	2447.5039	2480.1617	1.33	2449.8767	0.09	2448.0546	0.02	2447.4263	-0.00
11	2695.0238	2752.9776	2.15	2698.8794	0.14	2695.9891	0.03	2694.9838	-0.00
12	2903.7459	2965.6634	2.13	2908.7378	0.17	2904.9050	0.03	2903.7112	-0.00
13	4171.0937	4642.5827	11.30	4185.6695	0.34	4175.7421	0.11	4172.2235	0.02
14	4618.2581	5247.2765	13.62	4638.7500	0.44	4625.1155	0.14	4620.7173	0.05
15	4943.6005	5627.7234	13.83	4968.9942	0.51	4952.2810	0.17	4946.1234	0.05
16	5612.5382	6788.3534	20.94	5649.2119	0.65	5624.7074	0.21	5616.0995	0.06
17	5885.1186	7168.4861	21.80	5928.7508	0.74	5901.2304	0.27	5890.8632	0.09
18	6405.0077	8145.5214	27.17	6455.3445	0.78	6423.3003	0.28	6411.3622	0.09
19	6949.5716	8753.1054	25.95	7019.5746	1.00	6976.2206	0.38	6959.2560	0.13
20	7227.1974	9206.7514	27.39	7305.4595	1.08	7260.8489	0.46	7246.3976	0.26
21	9227.6363	11538.4415	25.04	9385.1938	1.70	9291.5744	0.69	9255.1319	0.29
22	9648.5681	12161.1051	26.04	9819.9903	1.77	9740.8221	0.95	9710.6628	0.64
23	10349.2749	12612.2378	21.86	10562.6628	2.06	10438.8840	0.86	10386.7336	0.36
24	11343.0263	14437.5032	27.28	11584.3087	2.12	11454.8751	0.98	11393.2999	0.44
25	11550.2063	14716.4766	27.41	11801.0786	2.17	11674.7866	1.07	11611.9547	0.53
26	11931.5463	17780.1577	49.01	12110.8396	1.50	12028.5299	0.81	11984.5476	0.44
27	12249.1092	18023.9453	47.14	12452.0166	1.65	12371.5918	0.99	12304.3617	0.45
28	12862.2577	19940.3168	55.02	13011.8484	1.16	12930.1670	0.52	12894.9812	0.25
29	13650.0872	21357.8035	56.46	13949.2997	2.19	13894.6359	1.79	13807.2997	1.15
30	14190.6086	25166.6704	77.34	14384.7244	1.36	14344.5120	1.08	14276.0253	0.60
31	16589.2026	38228.1560	130.4	18291.6090	10.26	16931.9277	2.06	16763.5384	1.05
32	17151.7868	38571.8462	124.8	19043.0780	11.02	17691.5648	3.14	17527.5737	2.19
33	17505.5446	44082.8281	151.8	19231.5735	9.85	17886.3436	2.17	17691.5648	1.06
34	18789.5466	52500.4114	179.4	20843.8389	10.93	19202.6709	2.19	19024.2284	1.24
35	19224.3217	68926.5428	258.5	21649.3433	12.61	19666.3700	2.29	19510.5470	1.48

greatest deviation between DSTAP and SUPERB model eigenvalues was about one percent. Without appropriate documentation, it is difficult to determine why the results from DSTAP and SUPERB agreed closely in the small models of the fixed-fixed example but deviate in the larger models of the Verdeel Truss.

The third curve in Fig. 16 corresponds to the combined models run in DSTAP. These models used exact-beam elements for the legs of the Verdeel Truss and FEM-beam elements for the horizontal members. The number of degrees of freedom in these models was increased by adding more FEM-beam elements. The regression line for these results is,

$$y = -6.8 + (0.66667)x$$

This line is steeper than either of the other two regression lines. Thus, it takes even fewer degrees of freedom to extract acceptable higher eigenvalues with combined models than with DSTAP or SUPERB finite element models. This result is promising since it was found above that the SUPERB FEM-beam formulation gave better results than the DSTAP FEM-beam formulation which was used in these combined models. Thus, even with the poorer FEM-beam element formulation employed in the FEC models, the exact-beam elements were able to improve the models sufficiently to give better model results than the SUPERB models. The tabulated results of the FEC models are presented in Table 12.

The solution times for the different techniques are compared in the same manner as in the previous example. That is, the total solution time for a model is divided by the number of acceptable eigenvalues obtained with that model. These ratios are presented in graphically in Fig. 17. The curves in this figure reflect the same trends as were seen in the Portal Arch example. The higher eigenvalues are extracted more economically with either a combined model or an exact model. For example, when there are more than 65 degrees of

Table 12. DSTAP Combined Model (FEC) Eigenvalues.

Program Routine	DSTAP BFMTHD	DSTAP BFMTHD		DSTAP BFMTHD		DSTAP BFMTHD		DSTAP BFMTHD	
# of FE	0	2	Rel.	4	Rel.	6	Rel.	8	Rel.
# of CE	6	4	Diff	4	Diff	4	Diff	4	Diff
dof	12	12	(%)	18	(%)	24	(%)	30	(%)
CPU sec	258.1900	227.5500		369.1000		664.6500		1058.4700	
1	107.1966	107.1978	0.00	107.1970	0.00	107.1966	0.00	107.1966	0.00
2	377.4589	377.6729	0.05	377.5641	0.02	377.4760	0.00	377.4647	0.00
3	397.2549	534.3450	34.50	400.8031	0.89	398.0289	0.19	397.5047	0.06
4	475.7334	830.6440	74.60	481.6326	1.24	477.0909	0.28	476.1728	0.09
5	1099.2899	1273.7065	15.86	1189.7511	8.22	1109.9622	0.97	1102.9998	0.33
6	1316.2433	1931.4248	46.73	2081.3655	58.12	1337.3768	1.60	1324.3020	0.61
7	1504.0376	2291.7081	52.37	2193.4467	45.83	1512.5566	0.56	1506.2520	0.14
8	1911.6293	2451.3042	28.23	2520.3462	31.84	1923.2429	0.60	1920.0321	0.43
9	2061.4500	2848.6489	38.18	3678.1027	78.42	2078.0802	0.80	2065.6214	0.20
10	2447.5039	3597.7735	46.99	4675.5190	91.03	2449.0649	0.06	2450.8987	0.13
11	2695.0238	6197.7142	129.9	4893.6162	81.57	3044.0843	12.95	2742.2607	1.75
12	2903.7459	6293.0632	116.7	5897.1256	103.0	3382.4969	16.48	2959.7318	1.92
13	4171.0937	6610.4310	58.48	6510.8447	56.09	4527.8708	8.55	4450.9114	6.70
14	4618.2581	6872.8248	48.81	8758.7492	89.65	5117.7473	10.81	4958.7765	7.37
15	4943.6005	11762.6313	137.9	11856.2889	139.8	5634.7126	13.97	5186.5908	4.91
16	5612.5382	12045.9483	114.6	12286.7726	118.9	5958.1750	6.15	5917.2160	5.42
17	5885.1186	12223.3724	107.6	12775.3813	117.0	6775.9252	15.13	6044.6658	2.71
18	6405.0077	12278.3550	91.69	13213.5893	106.3	6971.0718	8.83	6492.8655	1.37
19	6949.5716	12679.8749	82.45	13577.7597	95.37	9088.7753	30.78	8063.1715	16.02
20	7227.1974	12976.6223	79.55	13821.7643	91.24	10247.7793	41.79	8714.5409	20.57
21	9227.6363	18813.2399	103.8	18925.1919	105.0	10491.3559	13.69	10058.0795	8.99
22	9648.5681	20064.4725	107.9	20523.6193	112.7	11425.8471	18.42	10996.3008	13.96
23	10349.2749	20516.7454	98.24	20606.5506	99.11	11893.5116	14.92	11372.3592	9.88
24	11343.0263	20870.9895	83.99	21413.8087	88.78	12296.1247	8.40	11749.1811	3.58
25	11550.2063	24381.8745	111.0	24404.2744	111.2	14533.9294	25.83	12289.7745	6.40
26	11931.5463	26871.8127	125.2	26917.1903	125.5	15332.4979	28.50	12564.9921	5.30
27	12249.1092	30396.4043	148.1	30334.7198	147.6	15868.9029	29.55	14570.2375	18.94
28	12862.2577	30799.6458	139.4	30520.9834	137.2	16929.4891	31.62	14859.3327	15.52
29	13650.0872	30968.3991	126.8	31055.3661	127.5	19215.0941	40.76	16610.9193	21.69
30	14190.6086	33170.1899	133.7	33173.3234	133.7	20941.3572	47.57	18057.0110	27.24
31	16589.2026	35150.4064	111.8	37840.3711	128.1	22441.4776	35.27	18524.7306	11.66
32	17151.7868	37044.4389	115.9	38191.9977	122.6	23725.4898	38.32	19809.6326	15.49
33	17505.5446	42499.7480	142.7	41270.1573	135.7	24128.9750	37.83	21737.1593	24.17
34	18789.5466	42686.6842	127.1	42477.5194	126.0	26479.1886	40.92	23527.0430	25.21
35	19224.3217	42881.0598	123.0	42854.0589	122.9	30443.4340	58.35	24484.3340	27.36

Table 12. DSTAP Combined Model (FEC) Eigenvalues (continued).

Program Routine	DSTAP BFMTHD	DSTAP BFMTHD	
# of FE	0	10	Rel.
# of CE	6	4	Diff
dof	12	36	(%)
CPU sec	258.1900	1729.6800	
1	107.1966	107.1966	0.00
2	377.4589	377.1646	-0.07
3	397.2549	397.3580	0.02
4	475.7334	475.9149	0.03
5	1099.2899	1100.8606	0.14
6	1316.2433	1319.6876	0.26
7	1504.0376	1504.9309	0.05
8	1911.6293	1915.2443	0.18
9	2061.4500	2063.2616	0.08
10	2447.5039	2448.8780	0.05
11	2695.0238	2718.8640	0.88
12	2903.7459	2935.6818	1.09
13	4171.0937	4238.3727	1.61
14	4618.2581	4716.7942	2.13
15	4943.6005	4983.1073	0.79
16	5612.5382	5648.0927	0.63
17	5885.1186	5924.3911	0.66
18	6405.0077	6409.2304	0.06
19	6949.5716	7703.0385	10.84
20	7227.1974	8112.0575	12.24
21	9227.6363	9778.0739	5.96
22	9648.5681	10576.6049	9.61
23	10349.2749	10716.8253	3.55
24	11343.0263	11687.6693	3.03
25	11550.2063	12253.8659	6.09
26	11931.5463	12367.4574	3.65
27	12249.1092	12764.6829	4.20
28	12862.2577	13552.1932	5.36
29	13650.0872	15985.6317	17.11
30	14190.6086	17183.8486	21.09
31	16589.2026	17709.1930	6.75
32	17151.7868	18900.0709	10.19
33	17505.5446	19667.0212	12.34
34	18789.5466	19812.3991	5.44
35	19224.3217	23436.8548	21.91

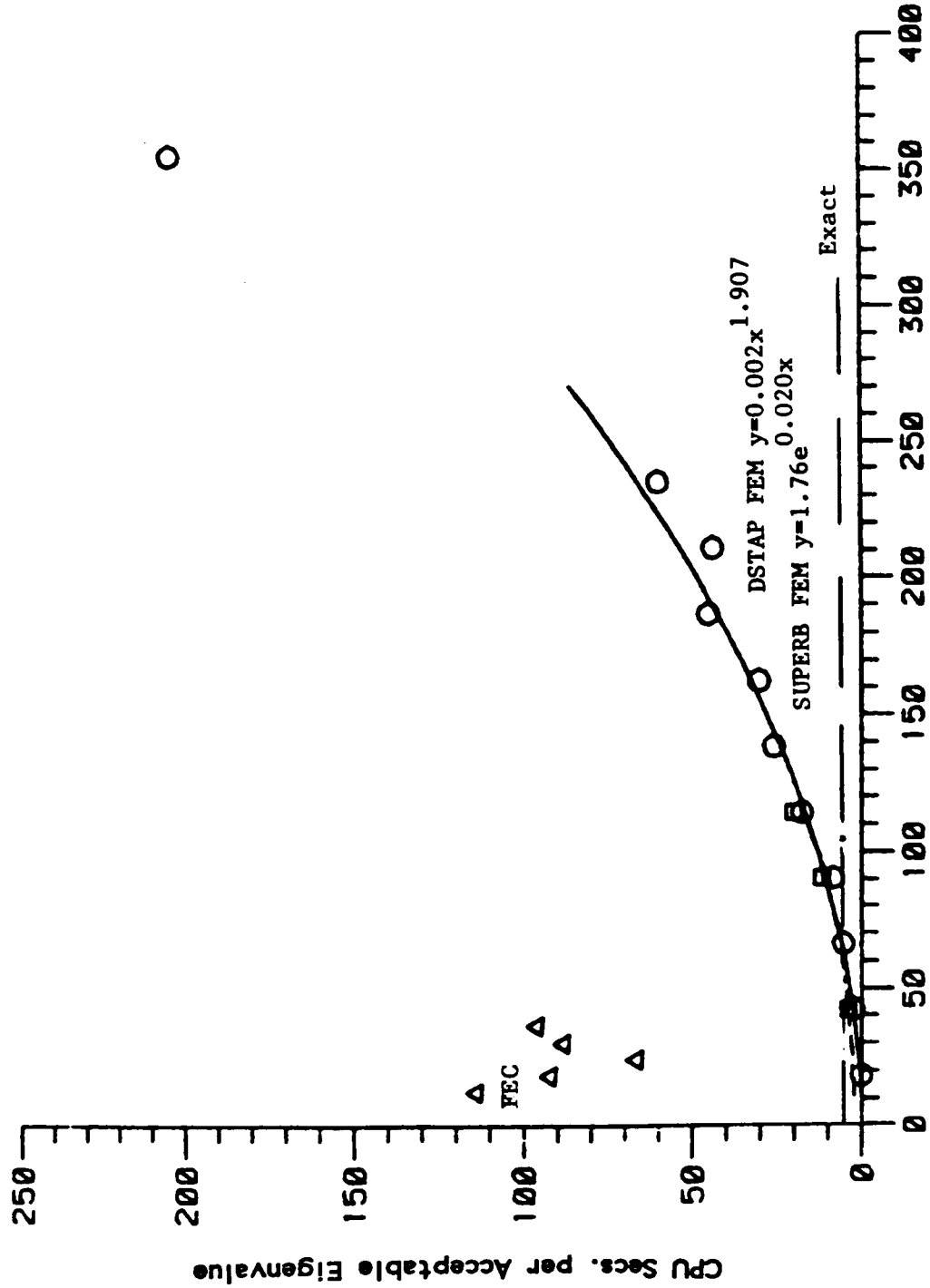


Figure 17. CPU seconds per Acceptable Eigenvalue versus Model Size for the Verdeel Truss.

freedom in a FEM model in DSTAP then it is more economical to use an exact-beam model in DSTAP. As this corresponds to 8 acceptable eigenvalues, the break-even point for the exact is 8 acceptable eigenvalues. Again, the combined model data shows too much scatter for a meaningful regression. The data for these models seem to be scattered mostly around 90 CPU sec per eigenvalue. Using this value, the break-even point for the combined models is about 36 eigenvalues.

The solution times for the SUPERB FEM models are lower than those from the DSTAP FEM and combined models. In fact, according to Fig. 17, the combined models could not give more economical solutions than SUPERB, at least for the range of eigenvalues sought. It is important to realize that the combined models were using the DSTAP FEM-beam element which was seen to be less effective than the SUPERB FEM-beam element for higher eigenvalues. Thus, this is an unfair comparison for the combined models. Furthermore, while SUPERB uses a sophisticated eigenvalue extraction technique, the DSTAP eigenvalue extraction for the FEC models is rather crude. The last reason why this is an unfair comparison for the FEC models is that DSTAP automatically finds the eigenvectors for the model degrees of freedom, and SUPERB was not asked to find eigenvectors.

The DSTAP exact-beam model solution was by far the least expensive for higher eigenvalues. It cost an average of 5.38 CPU seconds to extract an acceptable eigenvalue with an exact model in DSTAP, regardless of whether it was a high or low eigenvalue. As pointed out above, this model is more economical than the DSTAP FEM-beam element model if 8 or more acceptable eigenvalues are sought. The break-even point with the SUPERB solutions is 17 acceptable eigenvalues. This is quite remarkable considering that the same

rather crude eigenvalue extraction algorithm used for the FEC models is used for exact models in DSTAP. Moreover, DSTAP also finds the mode shapes automatically for the exact models, whereas SUPERB was not asked to find eigenvectors. However, in spite of these handicaps, the exact model in DSTAP gave the most economical way to compute the higher eigenvalues.

CONCLUSIONS

An exact beam-element formulation, based on a converted continuum transfer matrix was presented. This exact-element formulation was incorporated into a typical FEM code. In this setting, the exact-element formulation was shown to be equivalent to a dynamic condensation of an infinity of FEM degrees of freedom. Models using this exact element generate a transcendental, nonlinear eigenvalue problem, unsolvable with traditional linear eigenvalue problem extraction algorithms. A simple extraction technique was implemented into the hybrid finite element program DSTAP to study the effects of the exact-beam element on the approximation of higher structure eigenvalues. This extraction algorithm had to take into account the poles of the determinant of the dynamic stiffness matrix of each exact element. If the poles are not handled properly, there is the possibility of either predicting nonexistant eigenvalues or exceeding the computer representation of real numbers.

In the Numerical Examples section it was seen that an exact-element model is better suited for the extraction of higher structure eigenvalues/vectors than a standard FEM model. The first numerical example showed that an exact-element model is able to approximate the correct eigenvalues with an accuracy unmatched by an FEM model of the same size, or even models seven times larger. A fixed-fixed beam was modeled with only two equal-size-exact-beam elements, giving a total of 3 degrees of freedom. These 3 degrees of freedom lead to a 3 by 3 dynamic stiffness matrix, which was sufficient to approximate the 22 eigenvalues lying in the 1 to 100,000 rad/sec range. The computed eigenvalues matched to 9 significant digits with the

correct, closed-form-solution eigenvalues up to 60,000 rad/sec. For the extremely high structure frequencies between 60,000 and 100,000 rad/sec the accuracy of the exact-element model dropped slightly. By the 22^{nd} eigenvalue, the last eigenvalue in the range, the computed eigenvalues matched to 5 significant digits with the correct eigenvalues, giving a relative error of less than two ten-thousandths of one percent. An FEM model of the same size was only able to approximate 3 eigenvalues and only matched the first eigenvalue of the closed-form solution to 1 digit.

The addition of this exact-beam-element into an FEM model improved the capabilities of the FEM models. The resulting models of combined FEM and exact-beam elements needed fewer degrees of freedom than an FEM model to calculate acceptable higher eigenvalue approximations. The reduction of model size is an advantage when higher eigenvalues in large structures are needed with precision. Also, in the upper frequency region, the models using exact-elements had lower execution times per acceptable eigenvalue. The lower execution time per acceptable eigenvalue was an unexpected benefit of the exact-element incorporation which resulted from the rather simple transcendental-eigenvalue-extraction algorithm used. Furthermore, since the addition of an exact-element was shown to be analogous to the dynamic condensation of an infinite number of degrees of freedom, the resulting eigenvectors are actually improved. That is, smaller models give better eigenvectors because the reconstruction of the condensed degrees of freedom can be accomplished with almost any precision required. The only limitating factor is the truncation error incurred by the finite arithmetic on the computer. For the frequencies above 12,000 rad/sec, the truncation error was overcome by using a larger word length.

In the second example, a Portal Arch was modeled with purely exact-beam-element models, purely FEM-beam-element models, and with combined FEM-exact-beam-element models. Of the three types of models, the exact-beam-element models were the best suited for the extraction of higher eigenvalues and had the lowest execution time per acceptable eigenvalue. A three-element model with 6 degrees of freedom was able to extract the 35 eigen-pairs lying in the frequency of interest, 1 to 33,000 rad/sec. The only difficulty encountered with this model occurred in the 33^{rd} eigenvalue of the model with neither shear deformation nor rotatory inertia. This eigenpair was extracted by resorting to a 6-element model, thus, pushing the poles to higher frequencies. In other words, when it becomes difficult to find an eigenvalue due to the presence of poles, then by reducing the size of the beam elements, it is possible to change the location of the poles.

Of the two remaining types of models used to model the Portal Arch, the combined FEM-exact-beam-element models were better suited than the FEM-beam-element models for eigenvalue extraction. That is, for the same size model, the combined models gave better approximations for the higher eigenvalues. The combined models also gave lower solution times per acceptable eigenvalue than the purely FEM-element models. Naturally, the number of exact-elements as well as their location regulated the results for the combined models. These choices depend on the modeling techniques of the analyst. Thus, the combined-model results presented here are dependent on the author's choice of location and quantity of exact-elements. These choices were arbitrary because the optimum selection was beyond the scope of this study. Until such a study is conducted, the analyst using combined models must develop an engineering "feel" for the number of exact-elements and their location. The

rule of thumb used here was to replace as many FEM-elements as possible with a single exact-element.

In the third example, a Verdeel Truss structure was modeled with the same type of models as used in the previous example in program DSTAP and with standard FEM models in the commercial program SUPERB. The observations presented above were also applicable to the results from this example. The exact-element models were the best suited for extracting higher eigenvalues and had the lowest execution time per acceptable eigenvalue. The combined models were better suited than the purely FEM-element models in the upper frequency region. Finally, the hybrid research code presented here, DSTAP, was compared with a commercial code, SUPERB. These comparisons showed that the SUPERB-FEM models were better suited for higher eigenvalue extraction than the DSTAP-FEM models. Although the DSTAP-FEM-beam element was not as well formulated for higher frequencies as the SUPERB-FEM-beam element, the DSTAP combined models were better suited for higher eigenvalue/vector extractions than either the SUPERB-FEM or DSTAP-FEM models.

In summary, the addition of exact-beam elements to plane structure FEM-element models improves the approximations of higher structure eigenvalues. It will be up to the analyst to determine the choice of location and number of exact-elements to be added. From the results observed here, it is evident that if the structure to be modeled entirely of beams, then an exact-element model will be the better model. Moreover, if the structure to be modeled has non-beam components which can be found in transfer matrix catalogs, then it would be advantageous to convert these matrices to stiffness formulation and generate a new exact-element to be used with the exact-beam-element to model the structure.

Up to this point the advantages of using the exact element in DSTAP have been compared to the standard finite element method; however, DSTAP also has advantages over the standard transfer matrix method. By recasting the continuum transfer matrix into a general stiffness formulation, branching in the TMM can be easily accomplished. The user need not keep track of which path or branch is being followed nor in which direction they are being followed. As pointed out by Pestel and Leckie [42], this bookkeeping of paths, branches, and directions is rather cumbersome, and makes the TMM undesirable for complex geometries. Furthermore, by recasting a transfer matrix as a simple transformation into a stiffness matrix, the versatility of creating new elements with the TMM is preserved. That is, an analyst may create exact elements by combining several different continuum transfer matrices to describe a physical substructure. Then, the total transfer matrix may be entered into the code presented here to generate a new exact-element. Thus, although there are libraries of continuum transfer matrices for many applications, an analyst is not limited to these libraries, since he or she may easily create new exact-elements.

RECOMMENDATIONS FOR FUTURE WORK

Few investigations, if any, are ever completely finished. This investigation has shown that exact-displacement-beam elements are a feasible alternative to FEM-beam elements for the extraction of higher structure eigenvalues. The results presented here raise several unanswered questions. Some of these questions appear here as recommendations for future work.

A. Eigenvalue Sensitivity Study:

The dynamical stiffness of the beam exact displacement element contains the hyperbolic functions Sinh and Cosh, and the trigonometric functions Sin and Cos. The hyperbolic functions are known to grow very rapidly in magnitude as the frequency increases, whereas the magnitude of the trigonometric functions never exceeds unity. Several of the terms associated with the transverse deflection of an exact beam are made up of sums of trigonometric and hyperbolic functions. Therefore, there is the possibility that the contribution of the trigonometric functions will be lost in these terms at higher frequencies, due to machine truncation. A sensitivity study may lead to a better arrangement of the functions needed to evaluate the dynamic stiffness matrix. Ideally, the structure dynamic matrix might be decomposed into a product of two matrices, one matrix containing all the hyperbolic terms and another one containing the trigonometric terms. It may not be possible to obtain the ideal decomposition; however, determining which terms are more sensitive to frequency, geometry, and material parameters may lead to a better arrangement of the terms to reduce the truncation errors.

B. Eigenvector Sensitivity Study:

The acceptability of a solution to the eigenvalue problem was measured strictly on the basis of the computed eigenvalue in this study. This criteria does not guarantee that the corresponding eigenvectors will be likewise acceptable. Therefore, it should prove advantageous to study the sensitivity of both the purely FEM models and the combined models for eigenvector extractions. Qualitatively, the combined models should provide better eigenvector approximations. This study could quantify the error in eigenvector approximations and establish an acceptability criteria based on this error.

C. Eigenvalue Extraction Algorithm:

The eigenvalue extraction used here is a simple algorithm. Thus, there is much room for improvement. First, if the computer used is a sequential processor, the extraction algorithm can be easily changed such that the function $g(x)$ is curve fitted with an n^{th} order polynomial. The zeroes of the polynomial can be used to determine intervals that contain a zero of $g(x)$. This investigation should determine a suitable size polynomial. Preferably, the selection of the correct size polynomial could be automated with an empirical formula so that it would depend on the problem size. Second, if the computer used is a parallel processor, then the evaluation of $g(x)$ can be made simultaneously at several frequencies. These values of $g(x)$ can then be used to establish intervals containing roots or the roots themselves, depending on the number of evaluations. Once the intervals are found the search can continue in several intervals at the same time.

D. Linearization of the Non-Linear Eigenvalue Problem:

For the same size matrices, the FEM matrices gave more economical solutions than the exact displacement method, but the exact displacement method

gave more accurate eigenvalues. Should the exact displacement method eigenvalue problem be approximated by a linear eigenvalue problem, then the solution would take advantage of both methods. As presented earlier, linearization attempts have been able to provide bounds to the true eigenvalue(s) provided both the linearization and eigenvalue(s) were below the first pole. Once a pole is crossed the theorems developed for the linearization technique were no longer valid.

The current linearization techniques have been developed for frequencies below the first pole. This study has shown that it is at high frequencies when there is a definite advantage to using exact elements. The high frequencies of interest may lie above the first pole, so current linearization techniques are not really useful. These linearization schemes have been developed for models made up entirely of exact elements. In the combined models, the contribution of the FEM-elements does not need to be linearized. Thus, this fact can be used as a starting point for the linearization scheme. The task then is to find the contribution from the exact elements to this linear eigenvalue problem.

E. Extension to Other Exact-Displacement Models:

As presented in the discussion of the Transfer Matrix Method, there are many other continuum elements in TMM catalogs. These transfer matrices can be recast as exact-displacement element matrices by the transformation presented here and added to DSTAP. Thus, further studies with other exact-displacement elements can be carried out to verify the conclusions presented here. In particular, since plates and shells are popular FEM elements, it would be advantageous to develop exact-displacement shell and plate elements. The results of such a study should provide some interesting results

since the FEM plate and shell elements are based on energy formulations, which may make these elements less stable than the the FEM-beam element used in this study. The FEM-beam element may be formulated without resorting to energy formulations.

REFERENCES

1. Courant, R., "Variational Methods for the Solution of Equilibrium and Vibrations", Bulletin of American Mathematical Society, Vol 49, 1-23, 1943.

2. Turner, M. J., R. W. Clough, H.C. Martin, and L.J. Topp, "Stiffness and Deflection Analysis of Complex Structures", Journal of Aeronautical Sciences, Vol 23, 805-824, 1956.

3. Sung, S. H., and Nefske, D. J., "A Coupled Structural-Acoustic Finite Element Model for Vehicle Interior Noise Analysis", Presented at the Design and Production Engineering Technical Conference (Paper 83-DET-85), Dearborn Michigan. Sept. 11-14, 1983.

4. Zienkiewicz, O. C., and Newton, R. E., "Coupled Vibrations of a Structure Submerged in a Compressible Fluid", Proceedings of the Symposium on Finite Element Techniques, University of Stuttgart, pp. 360-379, 1969.

5. Schroeder, B. A., and Marcus, M.S., "Finite Element Solution of Fluid-Structural Interaction Problems", David Taylor Naval Ship Research and Development Center, Report 76-0145, Bethesda, Maryland, 1976.

6. Meirovitch, Leonard, Computational Methods in Structural Dynamics, Sijthoff & Noordhoff, Alphen aan den Rijn, The Netherlands, 1980.

7. Cook, R. D., Concepts and Applications of Finite Element Analysis, John Wiley and Sons, New York, 1974.

8. Zienkiewicz, O. C., The Finite Element Method, McGraw-Hill Book Co. (UK) London, 1979.

9. Bathe, K. J., and Wilson, E. L., Numerical Methods in Finite Element Analysis, Prentice-Hall Incorporated, Englewood Cliffs, New Jersey, 1976.

10. Bathe, K.J., Finite Element Procedures in Engineering Analysis, Prentice-Hall Incorporated, Englewood Cliffs, New Jersey, 1980

11. Rao, S.S., The Finite Element Method in Engineering, Pergamon Press, Oxford, England, 1982.

12. McCormick, C.W., Editor, MSC/NASTRAN User's Manual, MSR-39, The MacNeal-Schwendler Corp., Los Angeles, 1981.

13. Strang, G., and Fix, F.J., An Analysis of the Finite Element Method, Prentice-Hall, Englewood Cliffs, N.J., 1973.

14. DeSalvo, J., and Swanson, J.A., ANSYS Engineering Analysis System

User's Manual, Second Edition, Third printing, Swanson Analysis Systems, Inc., Houston Pennsylvania, 1983.

15. SDRC, User Manual for SUPERB 6.0, General Electric CAE International, Milford Ohio, 1983.

16. Kron, G., "Diakoptics - The Piecewise Solution of Large-Scale Systems", Electrical Journal, 20 articles starting June 1957 through February 1959.

17. Kron, G., Diakoptics, Macdonald, London England, 1963.

18. Przemieniecki, J.S., "Matrix Structural Analysis of Substructures", American Institute of Aeronautics and Astronautics Journal, vol. 1, pp. 138-147, 1963.

19. Spillers, W.R., "Network Analogy for Linear Structures", Journal of the Engineering Mechanics Division, ASCE, vol. 89, No. EM 4.

20. Spillers, W. R., "Network Analogy for Linear Structures", Journal of the Engineering Quarterly, vol. 15, pp. 31-41, 1964

21. Fenves, S. J., and Branin, F. H., "Network-Topological Formulation of Structural Analysis", Journal of the Structural Division, ASCE, vol. 89, ST 4, Proceedings Paper 3611, 1963.

22. Wiberg, N. E., System Analysis in Structural Mechanics by Use of Force and Displacement Variables, Ph.D. Thesis, Chalmers University, Gothenburg, Sweden, 1970.

23. Taig, I. C., "Automated Stress Analysis Using Substructures", Technical Report AFFDL-TR-66-80, U.S. Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio, 1966.

24. Guyan, R. "Reduction of Stiffness and Mass Matrices", AIAA Journal, vol. 3, pp. 380, 1965.

25. Irons, B. M., "Structural Eigenvalue Problem: Elimination of Unwanted Variables", AIAA Journal, vol. 3, pp. 961-962, 1965.

26. Petersmann, N., "Calculation of Eigenvalues using Substructures and Dynamic Condensation", paper 22 presented at the Conference of the Society of Experimental Stress Analysis, London, 1983.

27. Wright, E.C., and Miles, G.A., " An Economical Method for Determining the Smallest Eigenvalues of Large Linear Systems", International Journal for Numerical Methods in Engineering, Vol. 3, pp., 25-33, 1971.

28. Henshell, R.D. , and Ong, J.H.,"Automatic Masters for Eigenvalue Economization", Earthquake Engineering and Structural Dynamics,

vol. 3, pp. 375-383, 1975

29. Thomas, D.L., "Errors in Natural Frequency Calculations using Eigenvalue Economization", International Journal for Numerical Methods in Engineering, vol. 8, pp. 1521-1527, 1982

30. Williams, F. W., and Wittrick, W. H., "Exact Buckling and Frequency Calculations Surveyed", Journal of Structural Engineering, vol. 109, pp.169-187. Errata to 16 mistakes in this article can be found in vol. 109, p. 2759 of the same journal.

31. Hopper, C.T., and Williams, F. W., "Mode Finding in Nonlinear Structural Eigenvalue Calculations", Journal of Structural Mechanics, vol 5, pp. 255-278, 1977.

32. Hopper, C.T., Simpson, A., and Williams, F. W., " A Study of the Bounds on Eigenvalues of a Transcendental Dynamic Stiffness Matrix Provided by a Simply Derived Linear Matrix Pencil", Journal of Structural Mechanics, vol. 8, pp. 365-422, 1980.

33. Williams, F.W., "Incorporation of Lagrangian Multipliers into an Algorithm for Finding Exact Natural Frequencies or Critical Buckling Loads", International Journal of Mechanical Science, vol. 25, pp.579-584, 1983.

34. Richards, T.H., and Leung, A.Y.T., " An Accurate Method in Structural Vibration Analysis", Journal of Sound and Vibrations, vol. 55, pp. 363-376, 1977.

35. Williams, F. W., and Wittrick, W. H., "An Automatic Computational Procedure for Calculating Natural Frequencies of Skeletal Structures", International Journal of Mechanics and Science, vol. 12, pp. 781-791, 1970.

36. Wittrick, W. H., and Williams, F. W., "A General Algorithm for Computing Natural Frequencies of Elastic Structures", Quarterly Journal of Mechanics and Applied Mathematics, vol 24, pp 263-284, 1971.

37. Akesson, B. A., "PFVIBAT - A Computer Program for Plane-Frame Vibration Analysis by an Exact Method", International Journal for Numerical Methods in Engineering, vol 10, pp. 1221-1231, 1976.

38. Williams, F.W., and Howson, W.P., "Compact Computation of Natural Frequencies and Buckling Loads for Plane Frames", International Journal for Numerical Methods in Engineering, vol. 11, pp. 1067-1081, 1977.

39. Simpson, A., "A Newtonian Procedure for the Solutionn of [E]x= [A]x", Journal of Sound and Vibrations, vol. 82, pp. 161-170, 1982.

40. Blevins, R.D., Formulas for Natural Frequencies and Mode Shapes, Van Nostrand and Reinhotz and Co., New York, 1979.

41. Bishop, R.E.D., and Johnson, D.C., The Mechanics of Vibrations, Cambridge University Press, Cambridge, Great Britain, 1964.

42. Pestel, E.C., and Leckie, F.A., Matrix Methods in Elasto Mechanics, McGraw-Hill Book Company, Inc, New York, 1963.

43. Pilkey, W. D., Manual for the Response of Structural Members, Vols. 1 and 2, ITT Research Institute, Contract No. N00014-66-00343, Project No. 064-494, distributed by N.T.I.S..

44. Dokainish, M.A. "A New Approach for Plate Vibrations: Combination of Transfer Matrix and Finite-Element Technique", Transactions of the ASME, pp. 526-530, May 1972.

45. McDaniel, T.J. and Eversole, K.B., "A Combined Finite Element-Transfer Matrix Structural Analysis Method", Journal of Sound and Vibrations, Vol. 51, pp. 157-169, 1977.

46. Basci, M.I., Toridis, T.G., and Khozeimeh, K., "Improved Method of Free Vibration Analysis of Frame Structures", Computers and Structures, Vol. 10, pp. 255-265, 1979.

47. Chiatti, G. and Sestieri, A., "Analysis of Static and dynamic Structural Problems by a Combined Finite Element - Transfer Matrix Method", Journal of Sound and Vibrations, Vol. 67, pp. 35-42, 1979.

48. Sankar, S. and Hoa,S.V., "An extended Transfer Matrix-Finite Element Method for Free Vibration of Plates", Journal of Sound and Vibrations, Vol. 70, pp. 205-211, 1980.

49. McDaniel, T.J., and Chang, K.J., "Dynamics of Rotationally Periodic Large Space Structures", Journal of Sound and Vibrations, Vol. 68, pp. 351-368, 1980.

50. Mucino, V.H., and Pavelic, V., "An Exact Condensation Procedure for Chain-Like Structures Using a Finite Element-Transfer Matrix Approach", submitted to the Century 2 Design Technology Transfer Conference, San Francisco, California, August 19-21, 1980.

51. Degen, E.E., Shephard, M.S., and Loewy, R.G., Combined Finite Element-Transfer Matrix Method Based on a Mixed Formulation", presented at the Symposium on Advances and Trends in Structures and Dynamics, Arlington, Virginia, October 22-25, 1984.

52. Nagamatsu, A., and Nagaike, M., "Vibration Analysis of Movable Part of Internal Combustion Engine (Part 1. Crank Shaft)", Bulletin of the JSME, Vol. 24, No. 198, pp. 2141-2146, Dec. 1981.

53. Friberg, P.O., "Coupled Vibrations of Beams - An exact Dynamic Element Stiffness Matrix", International Journal for Numerical Methods in Engineering, vol. 19, No. 4, pp. 479-493.

54. Ewin, D.J., Modal Testing: Theory and Practice, Research Studies Press Ltd. - John Wiley, New York, NY, 1984.

55. Carnahan, B., Luther, H. A., and Wilkes, J. O., Applied Numerical Methods, John Wiley and Sons, 1969.

56. Mitchell, L.D., personal communication January 4, 1985, Mechanical Engineering Deapartment, Virginia Tech, Blacksburg, Virginia.

57. Virginia Tech Computing Center, Computing Center Log, 134 Burus Hall, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, Vol. 18, No. 2, 1985

58. Kamat, M.P., Personal Communication, Engineering Science and Mechanics Department, Virginia Polytechnic Institute and State University, June 1984.

59. Tektronix Inc., PLOT 10 Terminal Control System User's Manual, Manual Part No. 070-2241-00, Beaverton, Oregon, June 1979.

60. IMSL, User's Manual IMSL Library, Edition 9.2, Revised November, 1984, IMSL, 7500 Bellaire Boulevard, Houston, Texas, 1984.

61. Hallauer, W.L., Personal Communication, Aerospace and Ocean Engineering Department, Virginia Polytechnic Institute and State University, March 1985.

62. Tektronix Inc., Plot 50 Statistics, Volume 1, Tape 1, Beaverton, Oregon, 1976.

APPENDIX A

DYNAMIC ARRAY POINTERS IN DSTAP

The tables in this appendix contain the dynamic storage allocation for the A-array stored in the unnamed common block of program DSTAP. An explanation of the basics behind dynamic storage and matrix skyline storage may be found in Ref. [9]. Thus, they are not reproduced here.

Each of the attached tables corresponds to a particular section of program DSTAP. This section can be determined by the title of the table. For example, Table A-1 is applicable only during the assembly of the [K] and [M] matrices due to finite elements, the second time subroutine FEBEAM is called.

Appendix B shows how the pointers and dynamic arrays are used in DSTAP. The pointers N1 through N30 are stored in common block DIM, while pointers N101 through N114 are stored in common block CONPTS.

Table A-1. Distribution of the Dynamic Storage Area (A-array) During the Finite Element Matrix Assembly Process

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N1	3(NUMNP)	ID	INTEGER	Equation numbers associated with each degree of freedom
N2	NEQ + 1	MAXA	INTEGER	Pointer to the main diagonal terms which are stored in skyline fashion
N3	NWK	K	REAL*8	FEM stiffness matrix, [K]
N4 N7	NWK	M	REAL*8	FEM consistent mass matrix, [M]
N5 N101	NUMMAT	E	REAL*8	Modulus of elasticity
N102	NUMMAT	AREA	REAL*8	Cross section area
N103	NUMMAT	IXX	REAL*8	Area moment of inertia
NIXX	NUMMAT	RHO	REAL*8	Density of the material
NSYT	6(NUME)	LM	INTEGER	Equation numbers associated with each beam finite element
N104	6(NUME)	XYZ	REAL*8	Coordinates of the end-points of each beam finite element
N105	NUME	MATP	INTEGER	Material number of each element
N6 NLAST				

*Definition of terms in column two:
NEQ - Number of system equations
NWK - Number of elements in upper skyline storage
NUMNP - Number of nodal points
NUMMAT- Number of different materials
NUME - Number of finite elements

Table A-2. Distribution of the Dynamic Storage Area (A-array) During the Exact Beam Element Matrix Assembly Process

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N1	3(NUMNP)	ID	INTEGER	Equation numbers associated with each degree of freedom
N2	NEQ + 1	MAXA	INTEGER	Pointer to the main diagonal terms which are stored in skyline fashion
N3	NWK	K	REAL*8	FEM stiffness matrix, [K]
N4	NWK	M	REAL*8	FEM consistent mass matrix, [M]
N5 NE	NUMMAT	E	REAL*8	Modulus of elasticity
NG	NUMMAT	G	REAL*8	Shear modulus of rigidity
NARE	NUMMAT	AREA	REAL*8	Cross section area
NIXX	NUMMAT	IXX	REAL*8	Area moment of inertia
NRHO	NUMMAT	RHO	REAL*8	Density of the material
NLM	6(NUME)	LM	INTEGER	Equation numbers associated with each beam finite element
NXYZ	6(NUME)	XYZ	REAL*8	Coordinates of the end-points of each beam finite element
NMAT	NUME	MATP	INTEGER	Material number of each element
N6				

*Definition of terms in column two:
NEQ - Number of system equations
NWK - Number of elements in upper skyline storage
NUMNP - Number of nodal points
NUMMAT- Number of different materials
NUME - Number of finite elements

Table A-3. Distribution of the Dynamic Storage Area (A-array) During An Eigenvalue Extraction using EIGENR

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N1	3(NUMNP)	ID	INTEGER	Equation numbers associated with each degree of freedom
N2	NEQ + 1	MAXA	INTEGER	Pointer to the main diagonal terms which are stored in skyline fashion
N3	NWK	K	REAL*8	FEM stiffness matrix, [K]
N4	NWK	M	REAL*8	FEM consistent mass matrix, [M]
N5	NEQ*NEQ	A	REAL*8	Square [K] matrix
N6	NEQ*NEQ	B	REAL*8	Square [M] matrix
N7	NEQ	EIG	REAL*8	Eigenvalues returned by EIGENR
N8	NEQ*NEQ	W	REAL*8	Work area and eigenvectors
N9				

*Definition of terms in column two:
NEQ - Number of system equations
NUMNP - Number of nodal points
NWK - Number of matrix element in skylin storage

Table A-4. Distribution of the Dynamic Storage Area (A-array) During the Search for Poles

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N1	3(NUMNP)	ID	INTEGER	Equation numbers associated with each degree of freedom
N2	NEQ + 1	MAXA	INTEGER	Pointer to the main diagonal terms which are stored in skyline fashion
N3	NWK	K	REAL*8	FEM stiffness matrix, [K]
N4	NWK	M	REAL*8	FEM consistent mass matrix, [M]
N5 NE	NUMMAT	E	REAL*8	Modulus of elasticity
NG	NUMMAT	G	REAL*8	Shear modulus of rigidity
NARE	NUMMAT	AREA	REAL*8	Cross section area
NIXX	NUMMAT	IXX	REAL*8	Area moment of inertia
NRHO	NUMMAT	RHO	REAL*8	Density of the material
NLM	6(NCEL)	LM	INTEGER	Equation numbers associated with each continuous element
NXYZ	6(NCEL)	XYZ	REAL*8	Coordinates of the end-points of each continuous element
NMAT	NCEL	MATP	INTEGER	Material number of each element
N101	NCEL	SN	REAL*8	Sine of the orientation angle for each continuous element
N102	NCEL	CS	REAL*8	Cosine of the orientation angle for each continuous element
N103	NCEL	S2	REAL*8	S2=SN*SN
N104	NCEL	C2	REAL*8	C2=CS*CS
N105	NCEL	SC	REAL*8	SC=CS*SN
N106	NCEL	XL	REAL*8	Length of each continuous element

Table A-4. Distribution of the Dynamic Storage Area (A-array) During the Search for Poles (continued).

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N107				
	NCEL	MU	REAL*8	Density per unit length
N108				
	NCEL	SF	REAL*8	Constant needed to compute [K(w)]
N109				
	NCEL	TF	REAL*8	Constant needed to compute [K(w)]
N110				
	NCEL	BF	REAL*8	Constant needed to compute [K(w)]
N111				
	NCEL	AXB	REAL*8	Constant needed to compute [K(w)]
N112				
	NCEL	AXKF	REAL*8	Constant needed to compute [K(w)]
N113				
	6(NCEL)	LM	INTEGER	Equation number associated with each element
N6 N114				
	(MTOT-N114)/3	IEQ	INTEGER	Work array
N7 IAUX1				
	4(MTOT-N114)/9	IARRAY	REAL*8	Work array
N8 IAUX2				
	(MTOT-N114)/9	POL	REAL*8	Poles of [K(w)]
N9				

* Definition of terms for the column two:
NCEL - Number of exact elements
NEQ - Number of system equations
NPOL - Number of poles extracted
NUMNP - Number of nodal points
NUMMAT- Number of different materials
NWK - Number of terms stored in skyline fashion
MTOT - Total amount of dynamic storage

Table A-5. Distribution of the Dynamic Storage Area (A-array) During the Execution of Subroutine BFMTHD

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N1				
	3(NUMNP)	ID	INTEGER	Equation numbers associated with each degree of freedom
N2				
	NEQ + 1	MAXA	INTEGER	Pointer to the main diagonal terms which are stored in skyline fashion
N3				
	NWK	K	REAL*8	FEM stiffness matrix, [K]
N4				
	NWK	M	REAL*8	FEM consistent mass matrix, [M]
N5 N101				
	NCEL	SN	REAL*8	Sine of the orientation angle for each continuous element
N102				
	NCEL	CS	REAL*8	Cosine of the orientation angle for each continuous element
N103				
	NCEL	S2	REAL*8	S2=SN*SN
N104				
	NCEL	C2	REAL*8	C2=CS*CS
N105				
	NCEL	SC	REAL*8	SC=CS*SN
N106				
	NCEL	XL	REAL*8	Length of each continuous element
N107				
	NCEL	MU	REAL*8	Density per unit lenght
N108				
	NCEL	SF	REAL*8	Constant needed to compute [K(ω)]
N109				
	NCEL	TF	REAL*8	Constant needed to compute [K(ω)]
N110				
	NCEL	BF	REAL*8	Constant needed to compute [K(ω)]
N111				
	NCEL	AXB	REAL*8	Constant needed to compute [K(ω)]
N112				
	NCEL	AXKF	REAL*8	Constant needed to compute [K(ω)]
N113				
	6(NCEL)	LM	INTEGER	Equation number associated with each element
N6 N114				
	NPOL	MULT	INTEGER	Multiplicity of each pole
N7				

Table A-5. Distribution of the Dynamic Storage Area (A-array) During the Execution of Subroutine BFMTHD (continued).

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N7				
	NPOL	POL	REAL*8	Poles of $[K(\omega)]$
N8				
	NWK		REAL*8	$[K(\omega)]$ and $[K(\omega)]+[K]-\omega^2[M]$ stored in skyline fashion
N9				
	(NEQ)(NEQ)		REAL*8	$[K(\omega)]+[K]-\omega^2[M]$ converted to a square storage scheme
N10				
	NEQ	JP	INTEGER	Columns of $[K(\omega)]+[K]-\omega^2[M]$ pivoted during upper-triangularization
N11				
	NEQ	EIGVEC	REAL*8	Eigenvector Extracted from the matrix $[K(\omega)]+[K]-w^2[M]$
N12				

* Definition of terms for the column two:
NCEL - Number of exact elements
NEQ - Number of system equations
NPOL - Number of poles extracted
NUMNP - Number of nodal points
NWK - Number of terms stored in skyline fashion

Table A-6. Distribution of the Dynamic Storage Area (A-array) During the Post-Processing of Eigenvectors with Exact Elements

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N1				
	3(NUMNP)	ID	INTEGER	Equation numbers associated with each degree of freedom
N2				
	NEQ + 1	MAXA	INTEGER	Pointer to the main diagonal terms which are stored in skyline fashion
N3				
	NWK	K	REAL*8	FEM stiffness matrix, [K]
N4				
	NWK	M	REAL*8	FEM consistent mass matrix, [M]
N5 N101				
	NCEL	SN	REAL*8	Sine of the orientation angle for each continuous element
N102				
	NCEL	CS	REAL*8	Cosine of the orientation angle for each continuous element
N103				
	NCEL	S2	REAL*8	S2=SN*SN
N104				
	NCEL	C2	REAL*8	C2=CS*CS
N105				
	NCEL	SC	REAL*8	SC=CS*SN
N106				
	NCEL	XL	REAL*8	Length of each continuous element
N107				
	NCEL	MU	REAL*8	Density per unit lenght
N108				
	NCEL	SF	REAL*8	Constant needed to compute $[K(\omega)]$
N109				
	NCEL	TF	REAL*8	Constant needed to compute $[K(\omega)]$
N110				
	NCEL	BF	REAL*8	Constant needed to compute $[K(\omega)]$
N111				
	NCEL	AXB	REAL*8	Constant needed to compute $[K(\omega)]$
N112				
	NCEL	AXKF	REAL*8	Constant needed to compute $[K(\omega)]$
N113				
	6(NCEL)	LM	INTEGER	Equation number associated with each element
N6 N114				
	NUMNP	X	REAL*8	X coordinates of the model nodes
N7				

Table A-6. Distribution of the Dynamic Storage Area (A-array) During the Post-Processing of Eigenvectors with Exact Elements (continued)

A-array Pointers	Number of terms*	Array Name	Array Type	Array Content
N7				
	NUMNP	Y	REAL*8	Y coordinates of the model nodes
N8				
	NFEL*2	FELM	INEGER	FEM beam elements
N9				
	NCEL*2	CELM	INTEGER	Exact beam elements
N10				
	NEQ	XV	REAL*8	Eigenvector read in from tape
N11				
	MAXENT	X	REAL*8	X coordinate of plot point
N12				
	MAXENT	Y	REAL*8	Y coordinate of plot point
N13				
	MAXENT	DX	REAL*8	X displacement of plot point
N14				
	MAXENT	DY	REAL*8	Y displacement of plot point
N15				

* Definition of terms for the column two:

NCEL - Number of exact elements
NFEL - Number of FEM elements
NEQ - Number of system equations
NUMNP - Number of nodal points
NWK - Number of terms stored in skyline fashion
MAXENT- Maximum number of plot entries:
MAXENT=(MTOT-N11)/5
MTOT - Dimension of the A-array

APPENDIX B

Program DSTAP

Table B-1 contains a list of the modules necessary to generate the version of program DSTAP presented in the Program Description section of this dissertation. The first column of Table B-1 contains the module name. The second column contains the file in which the source code of the module may be found, and finally the third column contains the compiler and version number under which the module was compiled.

The files listed in Table B-1 are all on the VAX cluster at the Virginia Tech Computing Center. The subdirectory [JARAALJA.DISS] is on disk DUA3, while the SYS$COMMON and VPI$SYSROOT disks are system defined disks.

The modules appearing in bold in Table B-1 are presented following Table B-1. These subroutines are unique for DSTAP. The remaining subroutines in Table B-1 are subroutines that may be found in Ref. [9] (file STAP3.OLB), Tektronix PLOT10 library (file TCS.OLB), and the IMSL library (file IMSLIBD.OLB). For this reason these subroutines are not included here.

Table B-1. Program Modules of Program DSTAP.

Module Name	File containing module	Compiler - version
DSTAP	[JARAALJA.DISS]DSTAP.OBJ;3	VAX FORTRAN V4.1-45
CPUSEC	[JARAALJA.DISS]DSTAP.OBJ;3	VAX FORTRAN V4.1-45
PLOTB	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
INSTVC	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
PLOTC	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
MATVEC	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
MATMPL	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
QTRANM	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
ADDVEC	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
EXPVEC	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
PGDRW	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
QINSTVC	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
QTRANM2	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
QMATVEC	[JARAALJA.DISS]PLOTB2.OBJ;53	VAX FORTRAN V4.1-45
FEASEM	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
GETCDT	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
EIGEN	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
CPTVAL	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
CONDIM	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
COMPUT	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
POLES	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
ARRPOL	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
DETU2	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
BFMTHD	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
RPRT2	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
REPORT	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
WITWIL	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
COLSL	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
CNTSGN	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
FCN	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
CHKPOL	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
DETERM	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
CASSEM	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
AXIALK	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
BENDK	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
INCFEM	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
CVTSQR	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
DETER3	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
CVTDIA	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
UPPDET	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
SOLVEC	[JARAALJA.DISS]NEMEIG.OBJ;3	VAX FORTRAN V4.3-145
ANCHO	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
ANMODE	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
AOUTST	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2

Table B-1. Program Modules of Program DSTAP (continued).

Module Name	File containing module	Compiler - version
BAKSP	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
DASHA	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
DRAWA	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
DRWABS	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
DRWREL	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
DSHMOD	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
DWINDO	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
INITT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
KAM2AS	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
LINEF	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
LVLCHT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
MOVABS	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
MOVEA	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
MOVREL	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
NEWLIN	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
NEWPAG	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
RESCAL	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
RESET	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
REVCOT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
SETBUF	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
TINPUT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
TKDASH	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
TOUTPT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
TOUTST	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
TSEND	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
TWINDO	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
V2ST	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
VECMOD	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
WINCOT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
XCHAR	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
XYCNVT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
ALFMOD	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
ANSTR	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
BAKSPFI	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
BUFDMP	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
BUFFPK	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
CARTN	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
CHRSIZ	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
CLIPT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
CWSEND	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
FRAMEX	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
IOWAIT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
PARCLT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
PCLIPT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2

Table B-1. Program Modules of Program DSTAP (continued)

Module Name	File containing module	Compiler - version
PLTCHR	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
POUT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
PSCAL	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
TINSTR	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
ADEIN	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
ADEOUT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
DATOUT	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
DBOPEN	HSC000$DUA3:[JARAALJA.DISS]TCS.OLB;1	VAX-11 FORTRAN V3.0-2
ADDBAN	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
ADDRES	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
ASSEM	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
CLEAR	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
COLSOL	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
DSPACE	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
EIGENR	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
ELCAL	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
ELEMNT	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
ERROR	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
FALSI	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
HALF	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
INPUT	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
JACOBI	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
LOADS	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
LOADV	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
MULT	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
REDBAK	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
SCHECK	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
STRESS	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
TBEAM	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
TQL2	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
TRED2	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
TRUSS	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
WRITE	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
CBEAM	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
DECOMP	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
FEBEAM	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
COLHT	[JARAALJA.DISS]STAP3.OLB;1	VAX FORTRAN V4.1-45
LIB$ESTEMU	SYS$COMMON:[SYSLIB]STARLET.OLB;2	VAX/VMS Macro V04-00
EIGZF	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66
UERTST	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66
EQZQF	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66
EQZTF	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66
EQZVF	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66
UGETIO	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66

Table B-1. Program Modules of Program DSTAP (continued)

Module Name	File containing module	Compiler - version
USPKD	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66
VHSH2C	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66
VHSH2R	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66
VHSH3R	VPI$SYSROOT:[VPILIB]IMSLIBD.OLB;1	VAX-11 FORTRAN V2.5-66

```
C*****************************************************************
C                                                                *
C                            DSTAP                               *
C                                                                *
C                         (Dynamic STAPP)                        *
C                          version 2.11                          *
C                                                                *
C    Modified version of STAP to include static and free-vibration*
C    of a beam element.                                          *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)  *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C----------------------------------------------------------------*
C                                                                *
C                     *** DISCLAIMER ***                         *
C                                                                *
C         This program was developed for research purposes       *
C         only.  It has not been optimized for general use.      *
C                                                                *
C*****************************************************************
      REAL*8 FREQ1,FREQ2,STEP,WSTP
      CHARACTER*50 TIMHED(10)
      COMMON A(600000)
      COMMON /SOL/   NUMNP,NEQ,NWK,NUMEST,MIDEST,MAXEST,MK
      COMMON /DIM/   N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
     1,N16,N17,N18,N19,N20,N21,N22,N23,N24,N25,N26,N27,N28,N29,N30
      COMMON /EL/    IND,NPAR(10), NUMEG,MTOT,NFIRST,NLAST,ITWO
      COMMON /VAR/   NG,MODEX
      COMMON /DEBUG/           ! Contains the debug flags 1=on 0=off
     1         ID1,            ! The determinant is output to unit 12
     2         ID2,            ! Not assigned
     3         ID3,            ! Eigenvector in PLOTB to unit 17
     4         ID4,            ! Mode shapes of continuum elements are
*                              ! output from EXPVEC to unit 13
     5         ID5,            ! Mode shapes of continuum elements are
*                              ! output from PLOTC to unit 15
     6         ID6,            ! State vector used for mode shapes are
*                              ! output from PLOTC to unit 16
     7         ID7             ! Initial state vector computations out-
*                              ! put to unit 14
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IVB,IGEOM,IEIGV
      COMMON /CONPAR/NFEL,NCEL,NPOL,IFPOL,FREQ1,FREQ2,STEP,IWTWL,WSTP
      COMMON /SOLTIM/TIMEON,TPOL,TIM(10),KEIG,TIMEIG(60)
      COMMON /CONPTS/N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1              N110,N111,N112,N113,N114
      COMMON /PLTPAR/MAXENT,NPENTRY,IPLOTF

      DIMENSION HED(20),IA(1)

      DATA TIMHED /
     *'The input of geometry . . . . . .',
     *'The input of loads . . . . . .',
     *'Computation of global F.E.M. matrices . . . . . .',
     *'Triangularization of F.E.M. [K] matrix . . . . . .',
     *'Solution of load case(s) . . . . . .',
     *'Eigensolution of a purely F.E.M. problem . . . . .',
     *'Set-up for a Continuum-F.E. problem . . . . . .',
     *'Pole extraction of Continuum elements . . . . . .',
     *'Brute Force - Falsi extraction . . . . . .',
     *'Post-processing of eigenvectors . . . . . .'/

      EQUIVALENCE (A(1),IA(1))
C
      MTOT  = 600000
      ITWO  = 2

      IELMNT= 1
      ILOAD = 2
      NSTIF = 3
      IGEOM = 23
      IEIGV = 24
      IDEB  = 12
      IIN   = 25
      IOUT  = 26
C
  200 TIMEON = CPUSEC(0.)
      NUMEST = 0
      MAXEST = 0
      IFPOL=0
      READ (IIN,1000)  HED,NUMNP,NUMEG,NLCASE,MODEX,IVB,IPLOTF,
     1                 ID1,ID2,ID3,ID4,ID5,ID6,ID7
      IF (NUMNP.EQ.0) STOP 'SUCCESFUL COMPLETION OF PROGRAM STAPP'
      WRITE (IOUT,2000) HED,NUMNP,NUMEG,NLCASE,MODEX,IVB,IPLOTF
      IF (IVB.GT.0) THEN
         READ (IIN,*) METHOD,FREQ1,FREQ2,STEP,IWTWL,WSTP
         WRITE(IOUT,2002) METHOD,FREQ1,FREQ2,STEP,IWTWL,WSTP
      END IF
      IF (ID1) WRITE (IOUT,2003) 1,'Determinant output sent to unit 12'
      IF (ID3) WRITE (IOUT,2003) 3,'PLOTB Eigenvector sent to unit 17'
      IF (ID4) WRITE (IOUT,2003) 4,'EXPVEC mode shapes to unit 13'
      IF (ID5) WRITE (IOUT,2003) 5,'State vectors from PLOTC to unit 15'
      IF (ID6) WRITE (IOUT,2003) 6,'Mode shapes from PLOTC to unit 16'
      IF (ID7) WRITE (IOUT,2003) 7,'State vector calculations-unit 14'
```

```
      IF (ID1) WRITE (IOUT,2003) 1,'Determinant output sent to unit 12'
      IF (ID3) WRITE (IOUT,2003) 3,'PLOTB Eigenvector sent to unit 17'
      IF (ID4) WRITE (IOUT,2003) 4,'EXPVEC mode shapes to unit 13'
      IF (ID5) WRITE (IOUT,2003) 5,'State vectors from PLOTC to unit 15'
      IF (ID6) WRITE (IOUT,2003) 6,'Mode shapes from PLOTC to unit 16'
      IF (ID7) WRITE (IOUT,2003) 7,'State vector calculations-unit 14'
      IF (ID1) WRITE (6,2003) 1,'Determinant output sent to unit 12'
      IF (ID3) WRITE (6,2003) 3,'PLOTB Eigenvector sent to unit 17'
      IF (ID4) WRITE (6,2003) 4,'EXPVEC mode shapes to unit 13'
      IF (ID5) WRITE (6,2003) 5,'State vectors from PLOTC to unit 15'
      IF (ID6) WRITE (6,2003) 6,'Mode shapes from PLOTC to unit 16'
      IF (ID7) WRITE (6,2003) 7,'State vector calculations to unit 14'

      N1 = 1
      N2 = N1 + 3 * NUMNP
      N3 = N2 + NUMNP * ITWO
      N4 = N3 + NUMNP * ITWO
      N5 = N4 + NUMNP * ITWO
      IF (N5.GT.MTOT) CALL ERROR (N5 - MTOT,1)
      CALL INPUT (A(N1),A(N2), A(N3), A(N4), NUMNP, NEQ)
      TIM(1) = CPUSEC(TIMEON)
      NEQ1 = NEQ + 1
C
      IF (IVB.EQ.0) THEN
        N6 = N5 + NEQ * ITWO
        WRITE(IOUT,2005)
        REWIND ILOAD
        DO L= 1, NLCASE
          READ(IIN,1010) LL,NLOAD
          WRITE(IOUT,2010) LL, NLOAD
          IF (LL.NE.L) THEN
            WRITE(IOUT,2020)
            STOP
          END IF
          N7 = N6 + NLOAD
          N8 = N7 + NLOAD
          N9 = N8 + NLOAD * ITWO
          IF (N9.GT.MTOT) CALL ERROR (N9 - MTOT,2)
          CALL LOADS (A(N5),A(N6),A(N7),A(N8),A(N1),NLOAD,NEQ)
        END DO
      TIM(2)= CPUSEC(TIMEON)
      END IF
      N6 = N5 + NEQ
      DO I = N5,N6
        IA(I)=0
      END DO
C
      IND=1
      CALL ELCAL
      CALL ADDRES (A(N2),A(N5))
      MM = NWK/NEQ
      N3 = N2 + NEQ +1
      N4 = N3 + ITWO*NWK
      N5 = N4 + ITWO*((NWK-NEQ)*IVB + NEQ)
      WRITE(IOUT,2025) NEQ,NWK,MK,MM
      IF (MODEX.LT.1) GOTO 800

      IF (IVB.EQ.0) THEN
        N6 = N5 + MAXEST
        IF (N6.GT.MTOT) CALL ERROR (N6 - MTOT,4)
        NWL = NWK*2 + NEQ
        CALL CLEAR(A(N3), NWL)
        IND = 2
        CALL ASSEM (A(N5))
        TIM(3)= CPUSEC(TIMEON)
        CALL COLSOL (A(N3),A(N4),A(N2),NEQ,NWK,NEQ1,1)
        TIM(4)= CPUSEC(TIMEON)
        IND = 3
        REWIND ILOAD
        DO L=1,NLCASE
          CALL LOADV (A(N4),NEQ)
          CALL COLSOL (A(N3),A(N4),A(N2),NEQ,NWK,NEQ1,2)
          WRITE(IOUT,2015) L
          CALL WRITE (A(N4),A(N1),NEQ,NUMNP)
          CALL STRESS (A(N5))
        END DO
        TIM(5)= CPUSEC(TIMEON)
        DO 400 I=6,10
400       TIM(I)=TIM(5)
      ELSE                        ! eigenvalue problem
        IND = 2
        IF (NCEL.EQ.0) THEN   ! FEM-type eigenextraction
          CALL FEASEM (1)
          TIM(2)=TIM(1)
          TIM(3)= CPUSEC(TIMEON)
          CALL EIGEN (METHOD)
          TIM(4)=TIM(3)
          TIM(5)=TIM(3)
          TIM(6)= CPUSEC(TIMEON)
          DO 490 I=7,10
            TIM(I)=TIM(6)
490       END DO
        ELSE                      ! frequency search extraction
          CALL FEASEM (0)
          TIM(2)=TIM(1)
          TIM(3)= CPUSEC(TIMEON)
```

```
      CALL BFMTHD
      TIM(9)=CPUSEC (TIMEON)
         N7=N6+NUMNP*ITWO        ! Y-COORDINATE
         N8=N7+NUMNP*ITWO        ! F.E.'S
         N9=N8+NFEL*2            ! C.E.'S
         N10=N9+NCEL*2           ! Eigenvectors
         N11=N10+NEQ*ITWO        ! Graphic flag
         MAXENT=(MTOT-N11)/5     ! Maximum No. of Plot. elements
         N12=N11+MAXENT          ! X-Coor.
         N13=N12+MAXENT          ! Y-Coor.
         N14=N13+MAXENT          ! Delta X
         N15=N14+MAXENT          ! Delta Y
         IF (MAXENT.LT.(NUMNP + 20*NCEL))
     1      STOP 'STOP - Not enough space for plotting eigenvects'
      IF (IPLOTF) CALL PLOTB (A(N6),A(N7),A(N8),A(N9),A(N10),A(N1)
     1                       ,HED,A(N101),A(N102))
      TIM(10)= CPUSEC(TIMEON)
    END IF
  END IF

800 TTT=TIM(1)
  WRITE (IOUT,2030) HED
  IF (NCEL.EQ.0.) GOTO 805

  DO 802 I=KEIG,2,-1
802   TIMEIG(I)=TIMEIG(I)-TIMEIG(I-1)
  TIMEIG(1)=TIMEIG(1)-TIM(7)

805 DO 810 I=10,2,-1
      TIM(I)=TIM(I)-TIM(I-1)
810   TTT = TTT + TIM(I)
  TIM(7)=TIM(7)-TPOL
  TIM(8)=TPOL
  DO 820 I=1,10
    IF (TIM(I).EQ.0.) GOTO 820
    WRITE (IOUT,2032) TIMHED(I),TIM(I)
820 END DO
  WRITE (IOUT,2038) TTT
  IF (NCEL.EQ.0) GOTO 999
  WRITE (IOUT,2040)
  DO I=1,KEIG
    WRITE (IOUT,2042) I,TIMEIG(I)
  END DO

999 CONTINUE
C
  GO TO 200
C
```

```
1000 FORMAT (20A4/16I5)
1010 FORMAT (2I5)
2000 FORMAT (1H1,20A4///
   155H C O N T R O L  I N F O R M A T I O N                    //5X,
   255HNUMBER OF NODAL POINTS . . . . . (NUMNP)  = ,I5//5X,
   355HNUMBER OF ELEMENT GROUPS . . . . . (NUMEG)  = ,I5//5X,
   455HNUMBER OF LOAD CASES . . . . . (NLCASE) = ,I5//5X,
   555HSOLUTION MODE . . . . . (MODEX)  = ,I5 /5X,
   655H   EQ.0, DATA CHECK                                     /5X,
   755H   EQ.1, EXECUTION                                      //5X,
   155HVIBRATION ANALYSIS . . . . . (IVB) = ,I5 /5X,
   255H   EQ.0, STATIC  ANALYSIS                               /5X,
   355H   EQ.1, EXTRACT EIGENVALUES                            //5X,
   455HEIGENVECTOR RECONSTRUCTION FOR PLOTTING . . (IPLOTF) = ,I5 /5X,
   555H   EQ.0, OFF - NO RECONSTRUCTION                        ,  /5X,
   555H   EQ.1, ON  - RECONSTRUCT MODE SHAPES AND PLOT          //5X)
2002 FORMAT (5X,
   255HMETHOD USED IN EIGEN EXTRACTION. . . . . (METHOD) = ,I5//5X,
   355HSTARTING FREQUENCY . . . . . (FREQ1) = ,E15.7,
   4 //,5X,
   555HENDING FREQUENCY . . . . . (FREQ2) = ,E15.7,
   6 //,5X,
   755HSTEP SIZE FOR FREQUENCY SEARCH . . . . . (STEP ) = ,E15.7,
   8 //,5X,
   955HWITTRICK-WILLIAMS CHECK (1=ON, 0=OFF) . . . .(IWTWL) = ,I5,
   * //,5X,
   155HWITTRICK-WILLIAMS STEP-OFF . . . . . (WSTP)  = ,E15.7)
2003 FORMAT (/' Debug flag ID',I1,' was turned on',A40)
2005 FORMAT (1H1,26H L O A D  C A S E  D A T A)
2010 FORMAT(////4X,33H LOAD CASE NUMBER . . . . . =,I5//5X,
   132HNUMBER OF CONCENTRATED LOADS . = ,I5)
2015 FORMAT(1H1,9HLOAD CASE,I3)
2020 FORMAT (1X,40H*** ERROR LOAD CASES ARE NOT IN ORDER  )
2025 FORMAT (1H1,
   155HTOTAL SYSTEM DATA                                       ///5X,
   255HNUMBER OF EQUATIONS . . . . . (NEQ) = ,I5//5X,
   355HNUMBER OF MATRIX ELEMENTS . . . . . (NWK) = ,I5//5X,
   455HMAXIMUM HALF BANDWIDTH . . . . . (MK ) = ,I5//5X,
   555HMEAN HALF BANDWIDTH . . . . . (MM ) = ,I5)
2030 FORMAT (1H1,14X,52H S O L U T I O N  T I M E  L O G  I N  S E C O
   1N D S,//,5X,12HFOR PROBLEM:,1X,20A4 //5X)
2032 FORMAT (5X,A50,3H = ,F10.2)
2038 FORMAT (/,46X,'T O T A L = ',F10.2)
2040 FORMAT (//,5X,'ADDITIONAL TIMES OF INTEREST:,'/)
2042 FORMAT (5X,'Time to compute Eigenvalue number:',i2,'. . . . . ='
   1,F6.2)
    END
```

```
C*****************************************************************
C                                                                *
C                        FUNCTION CPUSEC                         *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|        (V. 4.1-45)  *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 3.2         *
C Date:       December 1984             |                       *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This function was developed for research purposes *
C               only.  It has not been optimized for general use.*
C                                                                *
C FUNCTION PURPOSE:                                              *
C               Evaluate the elapsed number of CPU seconds       *
C*****************************************************************
      REAL*4 FUNCTION CPUSEC (TIMEON)
C     ROUTINE TO OBTAIN CPUTIME ACCUMULATED BY THE CURRENT PROCESS
C
      INTEGER*2 LENGTH,TIMECODE
      INTEGER*4 CPUTIME,LENADDR,ADDRESS
C
C     DEFINE SYMBOLIC VALUES USED IN THE GETJPI CALL
C
      INCLUDE '($JPIDEF)'
C
C     SET UP DATA STRUCTURE FOR PASSING INFO TO GETJPI
C
      COMMON /JPILIST/ LENGTH,TIMECODE,ADDRESS,LENADDR
      DATA LENGTH/4/
      DATA TIMECODE/JPI$_CPUTIM/
      DATA LENADDR/0/
      ADDRESS=%LOC(CPUTIME)
C
C     INCLUDE THE PRECEEDING LINES IN YOUR PROGRAM DECLARATION SECTION
C     WHENEVER YOU WANT A TIME STAMP, ISSUE THE COMMAND BELOW
C     CPUTIME IS RETURNED IN THE VARIABLE "CPUTIME"
C
      CALL SYS$GETJPI(,,,LENGTH,,,)
C
C     CUMULATIVE CPUTIME IS GIVEN IN 10 MILLISECOND TICS
      CPUSEC = FLOAT(CPUTIME)/100. - TIMEON
      RETURN
      END
C*****************************************************************
C                                                                *
```

```
C                        SUBROUTINE ELEMNT                       *
C                                                                *
C----------------------------------------------------------------*
C Source: Numerical Methods in Finite Element Analysis           *
C         K. L. Bathe and E. L. Wilson                           *
C         Prentice Hall, Inc.                                    *
C Date:   1976                                                   *
C----------------------------------------------------------------*
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               This subrouine calls the appropriate element     *
C*****************************************************************

      SUBROUTINE ELEMNT
      COMMON /EL/ IND,NPAR(10),NUMEG,MTOT,NFIRST,NLAST,ITWO
C
      NPAR1=NPAR(1)
      GO TO (1,2,3,4),NPAR1
    1 CALL TRUSS
      RETURN
    2 CALL TBEAM                ! FEM-beam element
      RETURN
    3 CALL TBEAM                ! Exact-Beam elment
    4 RETURN
      END
      SUBROUTINE TBEAM
```

```
C*****************************************************************
C                                                                *
C                        SUBROUTINE TBEAM                        *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|        (V. 3.5-62)  *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 3.7         *
C Date:       Aug. 1984                 |                       *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use.*
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               This subrouine sets up dynamic storage allocation for *
C               both the exact and FEM beam elements.            *
C*****************************************************************
      COMMON /SOL/NUMNP,NEQ,NWK,NUMEST,MIDEST,MAXEST,MK
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
```

```
      COMMON /EL/ IND,NPAR(10),NUMEG,MTOT,NFIRST,NLAST,ITWO
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IVB,IGEOM,IEIGV
      COMMON A(1)
      INTEGER ELTYP
C
      EQUIVALENCE (NPAR(1),ELTYP),(NPAR(2),NUME),(NPAR(3),NUMMAT)
      IF (NUME.LT.1) THEN
         MIDEST = 0
         RETURN
      END IF
      IF (ELTYP.EQ.3) GOTO 500

C-----------------------------------------------------------------------
C                  FINITE ELEMENT METHOD BEAM ELEMENT
C-----------------------------------------------------------------------
      NFIRST=N6
      IF(IND.GT.1) NFIRST=N5
      N101=NFIRST
           NLAST=NUMMAT*ITWO
      N102=N101 + NLAST
      N103=N102 + NLAST
      NIXX=N103 + NLAST
      NRHO=NIXX + NLAST
      NSYT=NRHO + NLAST
           NLAST=6*NUME
      N104=NSYT + NLAST
      N105=N104 + NLAST*ITWO
      N106=N105 + NUME
      NLAST=N106
C
      IF (IND.GT.1) GO TO 100
         IF (NLAST.GT.MTOT) CALL ERROR(NLAST-MTOT,3)
         GO TO 200
C
  100 IF (NLAST.GT.MTOT) CALL ERROR(NLAST-MTOT,4)
C
  200 MIDEST=NLAST - NFIRST
C
      CALL FEBEAM (A(N1),A(N2),A(N3),A(N4),A(N4),A(N5),A(N101),A(N10
     12),A(N103),A(NIXX),A(NRHO),A(NSYT),A(N104),A(N105))
      RETURN
C
  500 CONTINUE
C-----------------------------------------------------------------------
C                  CONTINUUM BEAM ELEMENT
C          Set up the pointers for the continuum beam
C          and call the the CBEAM subroutine
C-----------------------------------------------------------------------
C   A(NE) = A(NFIRST)
C         <-- E Young's Modulus
C   A(NG)
C         <-- G Modulus of Rigidity
C   A(NAREA)
C         <-- AREA Cross-sectional area (assumed constant)
C   A(NIXX)
C         <-- IXX  Area moment of inertia
C   A(NRHO)
C         <-- RHO  Mass density
C   A(NLM)
C         <-- [LM]
C   A(NXYZ)
C         <-- [XYZ]
C   A(MATP)
C         <-- [MATP]
C   A(NLAST)
C-----------------------------------------------------------------------
C
C                  put these pointers in a if-then-else loop
      NFIRST=N6
      IF(IND.GT.1) NFIRST=N5
      NE   = NFIRST
                        NLAST=NUMMAT*ITWO
      NG   = NE    + NLAST
      NAREA= NG    + NLAST
      NIXX = NAREA + NLAST
      NRHO = NIXX  + NLAST
      NLM  = NRHO  + NLAST
                        NLAST=6*NUME
      NXYZ = NLM   + NLAST
      NMATP= NXYZ  + NLAST*ITWO
      NLAST= NMATP + NUME
      IF (IND.EQ.4) N6=NLAST
C
      IF (IND.EQ.1) THEN
         IF (NLAST.GT.MTOT) CALL ERROR(NLAST-MTOT,3)
      ELSE
         IF (NLAST.GT.MTOT) CALL ERROR(NLAST-MTOT,4)
      END IF
      MIDEST=NLAST - NFIRST
C
      CALL CBEAM (A(N1),A(N2),A(N3),A(N4),A(N4),A(N5),A(NE),A(NG)
     1,A(NAREA),A(NIXX),A(NRHO),A(NLM),A(NXYZ),A(NMATP))
C
      RETURN
      END
```

```
C*****************************************************************
C                                                                *
C                      SUBROUTINE FEBEAM                         *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|        (V. 3.5-62)  *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 3.7         *
C Date:       Sept. 1984                |                       *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use.  *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               This subrouine assembles the element mass and stiffness*
C               matrices, in global coordinates for the FEM-beam   *
C               element                                          *
C*****************************************************************
      SUBROUTINE FEBEAM(ID,X,Y,Z,U,MHT,E,AREA,IXX,RHO,SYT,LM,XYZ,MATP)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 IXX
      REAL A
      COMMON /SOL/       NUMNP,NEQ,     NWK,NUMEST,MIDEST,MAXEST,MK
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      COMMON /EL/ IND,NPAR(10),NUMEG,MTOT,NFIRST,NLAST,ITWO
      COMMON /VAR/ NG,MODEX
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IVB,IGEOM,IEIGV
      COMMON A(1)
C
      DIMENSION X(1),Y(1),Z(1),ID(3,1),E(1),AREA(1),LM(6,1),
     1XYZ(6,1),MATP(1),U(1),MHT(1)
      DIMENSION S(21),ST(6),D(3),SYT(1),IXX(1),RHO(1)
C
      EQUIVALENCE (NPAR(1),NPAR1),(NPAR(2),NUME),(NPAR(3),NUMMAT)
      SQRT(X)=DSQRT(X)
      ND=6
C
      GO TO (300,610),IND
C
  300 WRITE (IOUT,2000) NPAR1,NUME      ! Read material property sets
      IF (NUMMAT.EQ.0) NUMMAT=1
      WRITE (IOUT,2010) NUMMAT
      WRITE (IOUT,2020)
      DO 10 I=1,NUMMAT
      READ (IIN,*) N,E(N),AREA(N),IXX(N),RHO(N),SYT(N)
   10 WRITE (IOUT,2030) N,E(N),AREA(N),IXX(N),SYT(N),RHO(N)
      WRITE (IOUT,2040)
      N=1
      WRITE (IGEOM) NPAR1,NUME
  100 READ (IIN,1020) M,II,JJ,MTYP,KG
      IF (KG.EQ.0) KG=1
  120 IF(M.NE.N) GO TO 200
      I=II
      J=JJ
      MTYPE=MTYP
      KKK=KG
  200 XYZ(1,N)=X(I)
      XYZ(2,N)=Y(I)
      XYZ(3,N)=Z(I)
      XYZ(4,N)=X(J)
      XYZ(5,N)=Y(J)
      XYZ(6,N)=Z(J)
C
      WRITE (IGEOM) N,I,J       ! Save the geometry
      MATP(N)=MTYPE
      DO 390 L=1,6
  390   LM(L,N)=0
      DO 400 L=1,3
        LM(L,N)=ID(L,I)
        LM(L+3,N)=ID(L,J)
  400 CONTINUE
      CALL COLHT (MHT,ND,LM(1,N))       ! Update column heights
      WRITE (IOUT,2050) N,I,J,MTYPE
    3 FORMAT(2I5,F5.3)
      IF (N.EQ.NUME) RETURN
      N=N+1
      I=I+KKK
      J=J+KKK
      IF(N.GT.M) GO TO 100
      GO TO 120

  610 CONTINUE
C             Initialize the pointer for the mass matrix if needed
      IF (IVB.GT.0) N7 = N3 + NWK*ITWO
C             P=Volume
      P=0.
      DO 500 N=1,NUME
        MTYPE=MATP(N)
        XL2=0.
        DO 505 L=1,3
          D(L)=XYZ(L,N) - XYZ(L+3,N)
  505   XL2=XL2+D(L)*D(L)
        XL=SQRT(XL2)
        IF (XL.LT.1.E-27) THEN
```

```
      WRITE (6,6)
    6 FORMAT(1X,30HYOU HAVE A ZERO LENGTH ELEMENT)
      END IF
      P=P+XL*AREA(MTYPE)
C
C     Compute the element stiffness matrix in global coor.
C
C                     S(20) = C
      S(20)=-D(1)/XL
C                     S(21) = S
      S(21)=-D(2)/XL
      C2=S(20)*S(20)
      S2=S(21)*S(21)
      AE=E(MTYPE)/XL*AREA(MTYPE)
      EI2=2.*E(MTYPE)/XL*IXX(MTYPE)
      EI6=3.*EI2/XL
      EI12=2.*EI6/XL
C
      S( 1) =  AE*C2 + EI12*S2
      S( 2) =  S(20)*S(21)*(AE - EI12)
      S( 3) = -EI6*S(21)
      S( 4) = -S(1)
      S( 5) = -S(2)
      S( 6) =  S(3)
      S( 7) =  AE*S2 + EI12*C2
      S( 8) =  EI6*S(20)
      S( 9) = -S(2)
      S(10) = -S(7)
      S(11) =  S(8)
      S(12) =  2*EI2
      S(13) = -S(3)
      S(14) = -S(8)
      S(15) =  EI2
      S(16) =  S(1)
      S(17) =  S(2)
      S(18) = -S(3)
      S(19) =  S(7)
C                         EI2=SIN(angle)
                  EI2=S(21)
C                         EI6=COS(angle)
                  EI6=S(20)
      S(20) = -S(8)
      S(21) =  S(12)
C
      CALL ADDBAN (A(N3),A(N2),S,LM(1,N),ND)
C
C     If a dynamic analysis is being performed then go ahead and
C     assemble the consistent mass matrix.
      IF (IVB.LT.1) GOTO 500
C                     IVB = 0 static analysis
C                     IVB = 1 dynamic analysis
C
C           EI2 = SIN
C           EI6 = COS
            AE = AREA(MTYPE)*RHO(MTYPE)*XL
C
            S( 1) = AE*(C2/3. + 0.3714285714285714*S2)
            S( 2) = -AE*0.0380952380952380*CS
            S( 3) = -AE*XL*EI2*0.0523809523809523
            S( 4) = AE*(C2/6. + 0.1285714285714285*S2)
            S( 5) = -S(2)
            S( 6) = AE*XL*EI2*0.0309523809523809
            S( 7) = AE*(0.3714285714285714*C2 + S2/3.)
            S( 8) = AE*XL*EI6*0.0523809523809523
            S( 9) = -S(2)
            S(10) = AE*(0.1285714285714285*C2 + S2/6.)
            S(11) = -AE*XL*EI6*0.0309523809523809
            S(12) = AE*XL*XL/105.
            S(13) = S(6)
            S(14) = -S(11)
            S(15) = -AE*XL*XL/140.
            S(16) = AE*(C2/3. + 0.3714285714285714*S2)
            S(17) = S(2)
            S(18) = -S(3)
            S(19) = S(7)
            S(20) = -S(8)
            S(21) = S(12)
C
            CALL ADDBAN (A(N7),A(N2),S,LM(1,N),ND)
C
  500 CONTINUE
      WRITE (IOUT,2045) P
      RETURN
C
 1000 FORMAT (I5,5F10.0)
 1010 FORMAT(2F10.0)
 1020 FORMAT(5I5)
 2000 FORMAT (36H E L E M E N T  D E F I N I T I O N //,
     1     ' ELEMENT TYPE . . . . . . . . . . . . ( NPAR(1) ) . .= ',I5/,
     2     '     EQ.2,  FINITE ELEMENT METHOD',/,
     3     '            B-E-A-M ELEMENTS',//,
     5     ' NUMBER OF ELEMENTS. . . . . . . . .( NPAR(2) ) . .= ',
     6     I5//)
 2010 FORMAT (42H M A T E R I A L  D E F I N I T I O N     //,
     1     ' NUMBER OF DIFFERENT SETS OF MATERIAL',/,
     2     ' AND CROSS-SECTIONAL CONSTANTS. . . ( NPAR(3) ) . . = ',
```

```
     3I5//)
 2020 FORMAT(/2X,3HSET,7X,6HYOUNGS,6X,15HCROSS-SECTIONAL,7X,
     1 9HMOMENT OF,10X,5HYIELD,10X,7HDENSITY,/,1X,6HNUMBER,5X,7HMODULUS
     2 ,10X,4HAREA,12X,7HINERTIA,10X,8HSTRENGTH,//,15X,1HE,14X,1HA)
 2030 FORMAT (1X,I5,4X,E12.5,4(2X,E14.6))
 2040 FORMAT (/,1H1,40H E L E M E N T  I N F O R M A T I O N ///,
     1 8H ELEMENT ,5X,4HNODE,5X,4HNODE,7X,8HMATERIAL/,
     2 9H NUMBER-N,6X,1HI,8X,1HJ,7X,10HSET NUMBER/)
 2045 FORMAT(/5X,48HSTRUCTURE VOLUME . . . . . .
     1,7H . . =  ,F10.4)
 2050 FORMAT(I5,6X,I5,4X,I5,7X,I5)
 2060 FORMAT (////,46H S T R E S S  C A L C U L A T I O N S  F O R ,
     1 24HE L E M E N T  G R O U P,I4//,2X,6HNUMBER,13X,5HFORCE,
     213X,6HSTRESS,10X,10HP-CRITICAL,10X,11HTHEORY USED/)
 2070 FORMAT (1X,I5,11X,F13.0,2(4X,F13.0),11X,A8)
 2080 FORMAT (3I5)
      END
C
C*************************************************************************
C                                                                        *
C                          SUBROUTINE CBEAM                              *
C                                                                        *
C------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN          *
C Location:   Mechanical Engineering Dept.|         (V. 3.5-62)          *
C             V.P.I & S.U.              |                                *
C             Blacksburg, Virginia      | O.S.: VMS 3.7                  *
C Date:       Nov 1985                  |                                *
C------------------------------------------------------------------------*
C                                                                        *
C  DISCLAIMER:  This subroutine was developed for research purposes      *
C               only.  It has not been optimized for general use.        *
C                                                                        *
C SUBROUTINE PURPOSE:                                                    *
C               This subrouine assembles the element mass and stiff-     *
C               ness matrices for the exact-elements.                    *
C*************************************************************************
      SUBROUTINE CBEAM(ID,X,Y,Z,U,MHT,E,G,AREA,IXX,RHO,LM,XYZ,MATP)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 MU,E,G,AREA,RHO,IXX,RGY,X,Y,Z,XYZ,U
      REAL A
      COMMON /TESTC/ OMEGA
C
      COMMON /SOL/  NUMNP,NEQ,NWK,NUMEST,MIDEST,MAXEST,MK
      COMMON /DIM/  N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      COMMON /EL/   IND,NPAR(10),NUMEG,MTOT,NFIRST,NLAST,ITWO
      COMMON /VAR/  NG,MODEX
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IVB,IGEOM,IEIGV
      COMMON A(1)
C           external arrays
      DIMENSION X(1),Y(1),Z(1),ID(3,1),E(1),AREA(1),LM(6,1),
     1XYZ(6,1),MATP(1),U(1),MHT(1),G(1),IXX(1),RHO(1)
C           local arrays
      DIMENSION S(21),D(3),AK(2,2),BK(0:4,0:4)
C
      EQUIVALENCE (NPAR(1),NPAR1),(NPAR(2),NUME),(NPAR(3),NUMMAT)
C
      ND=6
  300 WRITE (IOUT,2000) NPAR1,NUME
      IF (NUMMAT.EQ.0) NUMMAT=1
      WRITE (IOUT,2010) NUMMAT
C
      WRITE (IOUT,2020)
      DO 10 I=1,NUMMAT
         READ (IIN,*) N,E(N),G(N),AREA(N),IXX(N),RHO(N)
         IF (G(N).GT.0.D0) THEN
            WRITE (IOUT,2030) N,E(N),G(N),AREA(N),IXX(N),RHO(N)
         ELSE
            WRITE (IOUT,2035) N,E(N),AREA(N),IXX(N),RHO(N)
         END IF
   10 CONTINUE
      WRITE (IOUT,2040)
      WRITE (IGEOM) NPAR1,NUME
      N=1
  100 READ (IIN,1020) M,II,JJ,MTYP,KG
      IF (KG.EQ.0) KG=1
  120 IF(M.NE.N) GO TO 200
      I=II
      J=JJ
      MTYPE=MTYP
      KKK=KG
C
C     ***** SAVE ELEMENT INFORMATION *****
C
  200 XYZ(1,N)=X(I)
      XYZ(2,N)=Y(I)
      XYZ(3,N)=Z(I)
      XYZ(4,N)=X(J)
      XYZ(5,N)=Y(J)
      XYZ(6,N)=Z(J)
C
      WRITE (IGEOM) N,I,J
C
      MATP(N)=MTYPE
C
      DO 390 L=1,6
```

```
  390    LM(L,N)=0
         DO 400 L=1,3
            LM(L,N)=ID(L,I)
            LM(L+3,N)=ID(L,J)
 400  CONTINUE
C
C     ***** UPDATE COLUMN HEIGHTS AND BANDWIDTH *****
C
      CALL COLHT (MHT,ND,LM(1,N))
C
      WRITE (IOUT,2050) N,I,J,MTYPE
    3 FORMAT(2I5,F5.3)
      IF (N.EQ.NUME) RETURN
      N=N+1
      I=I+KKK
      J=J+KKK
      IF(N.GT.M) GO TO 100
      GO TO 120
C
 1000 FORMAT (I5,5F10.0)
 1010 FORMAT(2F10.0)
 1020 FORMAT(5I5)
 2000 FORMAT (36H E L E M E N T  D E F I N I T I O N ///,
     1     14H ELEMENT TYPE ,13(2H .),16H( NPAR(1) ) . .=,I5/,
     2     '     EQ.3,  CONTINUUM BEAM ELEMENTS'/,
     5     20H NUMBER OF ELEMENTS.,10(2H .),16H( NPAR(2) ) . .=,
     6     I5//)
 2010 FORMAT (42H M A T E R I A L  D E F I N I T I O N     ///,
     1     37H NUMBER OF DIFFERENT SETS OF MATERIAL
     2       /31H AND CROSS-SECTIONAL CONSTANTS ,
     3                        4(2H .),17H( NPAR(3) ) . . =,I5//)
 2020 FORMAT(///2X,'SET',4X,'MODULUS OF',5X,'MODULUS OF',9X,'AREA',9X,
     1'MOMENT OF',9X,'DENSITY'/2X,'No.',4X,'ELASTICITY',6X,'RIGIDITY'
     2,24X,'INERTIA'/11X,'(E)',13X,'(G)',12X,'(AREA)',10X,'(IXX)',12X
     3,'(RHO)')
 2030 FORMAT (1X,I3,5(2X,E14.6))
 2035 FORMAT (1X,I3,2X,E14.6,'     Infinite   ',3(2X,E14.6))
 2040 FORMAT (/,1H1,40H E L E M E N T  I N F O R M A T I O N ///,
     1 8H ELEMENT ,5X,4HNODE,5X,4HNODE,7X,8HMATERIAL/,
     2 9H NUMBER-N,6X,1HI,8X,1HJ,7X,10HSET NUMBER/)
 2045 FORMAT(/5X,48HSTRUCTURE VOLUME . . . . . .
     1,7H . . = ,F10.4)
 2050 FORMAT(I5,6X,I5,4X,I5,7X,I5)
C
 2080 FORMAT (3I5)
      END
C*************************************************************************
C                                                                        *
```

```
C                          SUBROUTINE FALSI                              *
C                                                                        *
C------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN          *
C Location:   Mechanical Engineering Dept.|         (V. 3.5-62)         *
C             V.P.I & S.U.              |                                *
C             Blacksburg, Virginia      | O.S.: VMS 3.7                  *
C Date:       May 1984                  |                                *
C------------------------------------------------------------------------*
C                                                                        *
C  DISCLAIMER:  This subroutine was developed for research purposes     *
C               only.  It has not been optimized for general use.       *
C                                                                        *
C SUBROUTINE PURPOSE:                                                    *
C               This subroutine finds an approximation to a zero of      *
C               function F provided the interval [A,B] contains a root.*
C               The value of TOL is the tolerance with which the root    *
C               is to be approximated                                    *
C                                                                        *
C EXPECTED DIFFICULTIES:                                                 *
C               It is possible the approximated root lies outside the    *
C               interval [A,B].  In this case subroutine HALF is called*
C*************************************************************************
      SUBROUTINE FALSI (A,B,F,ROOT,TOL,ITERM,MESSAGE)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*(*) MESSAGE
      EXTERNAL F
      REAL*8 MIDPOINT
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP
      DATA EPSILON, ONE/1.D-16,1.D0/
C
C     check the tolerance:
      IF (TOL.LT.EPSILON) THEN
         MESSAGE='FAL-Tolerance exceeds the machine epsilon.'
         ITERM=3
         RETURN
      END IF
C
C     check the interval
      IF (.NOT.(A.LT.B)) THEN
         IF (A.GT.B) THEN
            MIDPOINT=A
            A=B
            B=MIDPOINT
         ELSE
            IF (A.EQ.B) THEN
               MESSAGE='FAL-The interval has zero length'
               ITERM=5
```

```
            RETURN
         END IF
      END IF
   END IF
C
C     calculate the maximum number of iterations allowed
C     this calculation is based on a pure bisection method.
      MAXITR=JIDNNT((DLOG10(B-A)-DLOG10(TOL))/.301029996)
C
C     make sure that there is a root in the interval
      FA=F(A)
      FB=F(B)
      IF (DSIGN(ONE,FA)*DSIGN(ONE,FB).GT.0.) THEN
         MESSAGE='FAL-There is no root in the specified interval'
         ITERM=2
         RETURN
      END IF
C
C     begin the regula falsi method
      I=1
C     regula falsi iteration
C
    1 OLDXN=XN
C     check to make sure that FA and FB are not zero
      IF (FA.EQ.0) THEN
         ROOT=A
         ITERM=6
         MESSAGE='FAL - Root was found exactly F(x)=0'
         RETURN
      ELSE
         IF (FB.EQ.0.D0) THEN
            ROOT=B
            ITERM=6
            MESSAGE='FAL - Root was found exactly F(x)=0'
            RETURN
         END IF
      END IF

      DENOM = 1.D0 - FB/FA
      IF (DENOM.EQ.0.D0) THEN
         CALL HALF (A,B,F,ROOT,TOL,ITERM,MESSAGE)
         RETURN
      END IF
      XN = A + (B - A)/(DENOM)
C     make sure XN lies within A and B:
      IF ((XN.LT.A).OR.(XN.GT.B)) THEN
         CALL HALF (A,B,F,ROOT,TOL,ITERM,MESSAGE)
         RETURN
      END IF
      FN=F(XN)
      IF (IFPOL) RETURN
C
C     see if iterations have been exceeded:
      IF (I.LT.MAXITR) THEN
C
C        see if a root has been found:
         IF (DABS(XN-OLDXN).LT.TOL) THEN
            ITERM=1
            MESSAGE='FAL-The tolerance test was met with ROOT.'
            ROOT=XN
            RETURN
         END IF
C        find the new bounds on the roots:
         IF (DSIGN(ONE,FA)*DSIGN(ONE,FN).LT.0.) THEN
            B=XN
            FB=FN
         ELSE
            A=XN
            FA=FN
         END IF
         I=I+1
         GOTO 1
      ELSE
         ITERM=2
         ROOT=XN
         MESSAGE='FAL-The number of iterations was exceeded.'
         RETURN
      END IF
      END
C
C*************************************************************
C                                                            *
C                      SUBROUTINE HALF                       *
C                                                            *
C------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|          (V. 3.5-62) *
C             V.P.I & S.U.              |                    *
C             Blacksburg, Virginia      | O.S.: VMS 3.7      *
C Date:       May 1984                  |                    *
C------------------------------------------------------------*
C                                                            *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                            *
C SUBROUTINE PURPOSE:                                        *
```

```
C             This subroutine finds an approximation to a zero of    *
C             function F provided the interval [A,B] contains a root.*
C             The value of TOL is the tolerance with which the root  *
C             is to be approximated                                  *
C*********************************************************************
      SUBROUTINE HALF(A,B,F,ROOT,TOL,ITERM,MESSAGE)
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER*(*) MESSAGE
      REAL*8 MIDPOINT, ONE
      EXTERNAL F
      DATA EPSILON,ONE/0.1D-16,1.D0/
C
C     check the tolerance:
      IF (TOL.LT.EPSILON) THEN
         MESSAGE='HAL-Tolerance exceeds the machine epsilon.'
         ITERM=3
         RETURN
      END IF
C
C     check the interval
      IF (.NOT.(A.LT.B)) THEN
         IF (A.GT.B) THEN
            MIDPOINT=A
            A=B
            B=MIDPOINT
         ELSE
            IF (A.EQ.B) THEN
               MESSAGE='HAL-The interval has zero length'
               ITERM=5
               RETURN
            END IF
         END IF
      END IF
C
C     calculate the maximum number of iterations allowed
      MAXITR=JIDNNT((DLOG10(B-A)-DLOG10(TOL))/0.301029996)
C
C     make sure that there is a root in the interval
      FA=F(A)
      FB=F(B)
      IF (DSIGN(ONE,FA)*DSIGN(ONE,FB).GT.0.) THEN
         MESSAGE='HAL-There is no root in the specified interval'
         ITERM=2
         RETURN
      END IF
C
C     start the bisection method
C
      ITR=0
    1 ITR=ITR+1
      MIDPOINT=(A+B)/2.
      IF (ITR.GT.MAXITR) THEN
         MESSAGE='HAL-The maximum number of iterations reached'
         ITERM=4
         ROOT=MIDPOINT
         RETURN
      END IF
      FM=F(MIDPOINT)
      IF (DSIGN(ONE,FM)*DSIGN(ONE,FA).LE.0.) THEN
         B=MIDPOINT
      ELSE
         A=MIDPOINT
         FA=FM
      END IF
C
C     see if the tolerance criteria was met
      IF ((B-A).LE.TOL) THEN
         ROOT=(A+B)/2.
         ITERM=1
         MESSAGE='HAL-Tolerance was met.'
         RETURN
      END IF
C
      GOTO 1
      END
C*********************************************************************
C                                                                     *
C                        SUBROUTINE FEASEM                            *
C                                                                     *
C---------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN       *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)      *
C             V.P.I & S.U.              |                             *
C             Blacksburg, Virginia      | O.S.: VMS 4.1               *
C Date:       May 1985                  |                             *
C---------------------------------------------------------------------*
C                                                                     *
C  DISCLAIMER:  This subroutine was developed for research purposes   *
C               only.  It has not been optimized for general use.     *
C                                                                     *
C SUBROUTINE PURPOSE:                                                 *
C               This subrouine assembles the global mass and stiffness*
C               matrices due to the finite elements, and it reads back*
C               the exact elements' data.                             *
C*********************************************************************
      SUBROUTINE FEASEM (FEM)
```

```
      LOGICAL FEM
      REAL*4 A
      COMMON A(1)
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      COMMON /SOL/ NUMNP,NEQ,NWK,NUMEST,MIDEST,MAXEST,MK
      COMMON /EL/ IND,NPAR(10),NUMEG,MTOT,NFIRST,NLAST,ITWO
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV

      NNL=2*NWK
      CALL CLEAR (A(N3),NNL)
      REWIND (IELMNT)
      CALL GETCDT (A(N5))
      CALL TBEAM
      IF (FEM) RETURN
      CALL GETCDT (A(N5))
      RETURN
      END
C
C*****************************************************************
C                                                               *
C                      SUBROUTINE GETCDT                        *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte      | Language: VAX FORTRAN     *
C Location:   Mechanical Engineering Dept.  (V. 4.1-45)       *
C             V.P.I & S.U.                                      *
C             Blacksburg, Virginia  | O.S.: VMS 4.1            *
C Date:       May 1985              |                           *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C               Get the exact element data from tape.           *
C*****************************************************************
      SUBROUTINE GETCDT (AA)
      COMMON /EL/ IND,NPAR(10),NUMEG,MTOT,NFIRST,NLAST,ITWO
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      COMMON /SOL/ NUMNP,NEQ,NWK,NUMEST,MIDEST,MAXEST,MK
      DIMENSION AA(1)
      READ (IELMNT) NUMEST,NPAR,(AA(I),I=1,NUMEST)
      RETURN
      END
C*****************************************************************
C                                                               *
C                      SUBROUTINE EIGEN                         *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte      | Language: VAX FORTRAN     *
C Location:   Mechanical Engineering Dept.  (V. 4.1-45)       *
C             V.P.I & S.U.                                      *
C             Blacksburg, Virginia  | O.S.: VMS 4.1            *
C Date:       May 1985              |                           *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C               Prepare the storage allocation for an eigenvalue- *
C               vector extraction of a purely finite element model. *
C               Also call the appropriate subprograms.          *
C*****************************************************************
      SUBROUTINE EIGEN (METHOD)
      COMMON /SOL/ NUMNP,NEQ,NWK,NUMEST,MIDEST,MAXEST,MK
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      COMMON /EL/ IND,NPAR(10), NUMEG,MTOT,NFIRST,NLAST,ITWO
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      COMMON A(1)
      REAL*8  RTOL

      GOTO (100,200,300),METHOD
  100 CONTINUE
C---------------------------------------------------------------
C         Set up the pointers for a sub-space iteration and
C         call the appropriate subroutine
C---------------------------------------------------------------
C   A(N1)                          A(N8)
C        <-- [ID]                        <-- {W}
C   A(N2)                          A(N9)
C        <-- {MAXA}                      <-- {AR}
C   A(N3)                          A(N10)
C        <-- [K]                         <-- {BR}
C   A(N4)                          A(N11)
C        <-- [M]                         <-- [VEC]
C   A(N5)                          A(N12)
C        <-- [R]                         <-- {D}
C   A(N6)                          A(N13)
C        <-- {EIGV}                      <-- {BUP}
C   A(N7)                          A(N14)
C        <-- {TT}                        <-- {BLO}
C   A(N8)                          A(N15)
C                                       <-- {BUPC}
C                                  A(N16)
C                                       <-- {RTOLV}
```

```
C                                A(N17)                              |
C-------------------------------------------------------------------
      NC=JMIN0(2*NROOT,NROOT+8)
      IF (NC.GT.NEQ) NC=NEQ
      NNC=NC*(NC + 1)/2
C
      N4=N7
      N5=N4 + ITWO*NWK
      N6=N5 + ITWO*NEQ*NC
      N7=N6 + ITWO*NC
      N8=N7 + ITWO*NEQ
      N9=N8 + ITWO*NEQ
      N10=N9 + ITWO*NNC
      N11=N10 + ITWO*NNC
      N12=N11 + ITWO*NC*NC
      N13=N12 + ITWO*NC
      N14=N13 + ITWO*NC
      N15=N14 + ITWO*NC
      N16=N15 + ITWO*NC
      N17=N16 + ITWO*NC
      IF (N16.GT.MTOT) CALL ERROR (N17-MTOT,5)
C
C                MAXITR = Maximum number of iterations
      MAXITR=16
C                RTOL = Tolerance for root
      RTOL=1.D-6
C                ISSC = 1 Perform Sturm sequence check
C                     = 0 Don't perform the check
      ISSC=1
C                IPRT = 1 Print results during iteration
C                     = 0 Don't print intermediate results
      IPRT=1
C
      CALL DSPACE(A(N3),A(N4),A(N2),A(N5),A(N6),A(N7),A(N8),A(N9),A(N10)
     1           ,A(N11),A(N12),A(N16),A(N13),A(N14),A(N15),NEQ,NEQ + 1
     2           ,NWK,NWK,NROOT,RTOL,NC,NNC,MAXITR,ISSC,IPRT,NSTIF,IOUT)
      RETURN

  200 CONTINUE
      NC=NEQ*NEQ*ITWO
      N6=N5+NC
      N7=N6+NC
      N8=N7+NEQ*ITWO
      N9=N8+NC
      IF (MTOT.LT.N9) stop 'STOP - Not enough space to implement the Q_R
     1algorithm'
      CALL CVTSQR (A(N3),A(N5),A(N2),NWK,NEQ)
      CALL CVTSQR (A(N4),A(N6),A(N2),NWK,NEQ)
      CALL EIGENR (NEQ,NEQ,A(N5),A(N6),A(N7),A(N8))
      WRITE (IOUT,1000)
      DO I=N7,N8-2,2
         CALL REPORT (A(I),-1,11111111)
      END DO
      RETURN

  300 CONTINUE
                                 NC=NEQ*NEQ*ITWO
      N6=N5 + NC                ! A(N6)=MASS
      N7=N6 + NC                ! A(N7)=Alfa array
                      NC= NEQ*ITWO
      N8=N7 + 2*NC              ! A(N8)=Beta array
      N9=N8 +   NC              ! A(N9)=work array
      N10=N9 +  NC
      IF (N10.GT.MTOT) STOP 'STOP - no space for calling EIGZF in IMSL'
      CALL CVTSQR (A(N3),A(N5),A(N2),NWK,NEQ)
      CALL CVTSQR (A(N4),A(N6),A(N2),NWK,NEQ)
      CALL EIGZF (A(N5),NEQ,A(N6),NEQ,NEQ,0,A(N7),A(N8),A(N10),1,A(N9)
     1           ,IER)
      WRITE (IOUT,1010) IER
      CALL RPRT2 (A(N7),A(N8),A(N9),NEQ)
      RETURN
 1000 FORMAT (1H1,10X,'EIGENVALUES BY EIGENR ROUTINE',//,6X,'Eig.',
     110X,' Lambda ',14X,'Eigenvalue',/,6X,3HNo.)
 1010 FORMAT (1H1,10X,'EIGENVALUES BY EIGZS ROUTINE',//,2X,'Returned fr'
     1,'om EIGZF in IMSL with IER=',I3)
      END
C*******************************************************************
C                                                                  *
C                      SUBROUTINE CPTVAL                           *
C                                                                  *
C------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN    *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)   *
C             V.P.I & S.U.              |                          *
C             Blacksburg, Virginia      | O.S.: VMS 4.1            *
C Date:       May 1985                  |                          *
C------------------------------------------------------------------*
C                                                                  *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use.  *
C                                                                  *
C SUBROUTINE PURPOSE:                                              *
C               Prepare storage allocation and constants for eigen-*
C               value/vector extraction of a nonlinear eigenvalue  *
C               problem.  Constants include values independent of  *
C               frequency and poles.                               *
```

```
C*****************************************************************
      SUBROUTINE CPTVAL
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 A,TIMEON,TPOL,TIM,TIMEIG
      COMMON A(1)
      COMMON /EL/ IND,NPAR(10),NUMEG,MTOT,NFIRST,NLAST,ITWO
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      COMMON /SOL/ NUMNP,NEQ,NWK,NUEST,MIDEST,MAXEST,MK
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      COMMON /CONPTS/ N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1                N110,N111,N112,N113,N114
      COMMON /SOLTIM/ TIMEON,TPOL,TIM(10),KEI,TIMEIG(60)

      EQUIVALENCE (NUME,NPAR(2)),(NUMMAT,NPAR(3))

      NFIRST=N5
                              IAUX=NUMMAT*ITWO
      NE    =NFIRST
      NG    =NE    +IAUX
      NARE  =NG    +IAUX
      NIXX  =NARE  +IAUX
      NRHO  =NIXX  +IAUX
      NLM   =NRHO  +IAUX
                              IAUX=6*NUME
      NXYZ  =NLM   +IAUX
      NMATP =NXYZ  +IAUX*ITWO
      NLAST =NMATP +NUME
                              IAUX=NUME*ITWO
      CALL CONDIM (NLAST,IAUX)
      N114  =N113 + 6*NUME
      N6    =N114
      IF (N114.GT.MTOT) STOP 'STOP - Not enough space to set-up values'

      CALL COMPUT (A(NE),A(NG),A(NARE),A(NIXX),A(NRHO),A(NXYZ),
     1             A(NMATP),A(N101),A(N102),A(N103),A(N104),A(N105),
     2             A(N106) ,A(N107),A(N108),A(N109),A(N110),A(N111),
     3             A(N112) )

      TPOL=CPUSEC(TIMEON)
      IAUX2=(MTOT-N114)/3
      NPMAX=IAUX2/3
      IAUX1=N114+IAUX2
      IAUX2=IAUX1+IAUX2*4/3
      CALL POLES (A(NRHO),A(NARE),A(NE),A(NMATP),A(N106),A(N110),
     1            A(N108),A(N109),A(IAUX1),A(IAUX2),A(N6),NPMAX)
      TPOL=CPUSEC(TIMEON) - TPOL

      IAUX=N114 - N113
      NE=N113-1
      NG=NUM-1
      DO I=1,IAUX
         A(NE+I)=A(NG+I)
      END DO

      IAUX=N114 + ITWO*NPOL + NPOL - N101
      NE=N5-1
      NG=N101-1
      DO I=1,IAUX
         A(NE+I)=A(NG+I)
      END DO
      IAUX=NUME*ITWO
      CALL CONDIM (N5,IAUX)
      N114= N113 + 6*NUME
      N6  = N114
      N7  = N6 + NPOL
      N8  = N7 + NPOL*ITWO
      N9  = N8 + NWK*ITWO
      N10 = N9 + NEQ*NEQ*ITWO
      N11 = N10 + NEQ -1
      N12 = N11 + NEQ*ITWO
      IF (N12.GT.MTOT) STOP 'STOP - No space to solve Cont.-FE mats.'
      RETURN
      END
C
C*****************************************************************
C                                                               *
C                      SUBROUTINE CONDIM                        *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
C Date:       May 1985                  |                       *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C             Compute the new pointers.                         *
C*****************************************************************
      SUBROUTINE CONDIM (IBEG,ISTP)
      COMMON /CONPTS/ N(14)
      N(1)=IBEG
```

```
      DO I=2,13
         N(I)=N(I-1)+ISTP
      END DO
      RETURN
      END

C*****************************************************************
C                                                                *
C                    SUBROUTINE COMPUT                           *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|       (V. 4.1-45)   *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Compute values independent of frequency.         *
C*****************************************************************
      SUBROUTINE COMPUT (E,G,AREA,IXX,RHO,XYZ,MATP,SI,CO,S2,C2,CS,
     1                   XL,MU,SF,TF,BF,AXB,AXKF)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 IXX,MU,IY2
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      DIMENSION E(1),G(1),AREA(1),IXX(1),RHO(1),XYZ(6,1),MATP(1),SI(1),
     1          CO(1),S2(1),C2(1),CS(1),XL(1),MU(1),SF(1),TF(1),BF(1),
     2          AXB(1),AXKF(1)
      DIMENSION D(3)

      DO I=1,NCEL
         MTYPE=MATP(I)
         XL2=0.D0
         DO L=1,3
            D(L)=XYZ(L,I)-XYZ(L+3,I)
            XL2=XL2+D(L)*D(L)
         END DO
         XL(I)=DSQRT(XL2)
         CO(I)=-D(1)/XL(I)
         SI(I)=-D(2)/XL(I)
         S2(I)=SI(I)*SI(I)
         C2(I)=CO(I)*CO(I)
         CS(I)=CO(I)*SI(I)
         MU(I)=RHO(MTYPE)*AREA(MTYPE)
         IY2  =IXX(MTYPE)/AREA(MTYPE)
         BF(I)=MU(I)*XL2*XL2/E(MTYPE)/IXX(MTYPE)
         IF (G(MTYPE).LT.0.D0) THEN
            SF(I)=0.D0
            TF(I)=0.D0
         ELSE
            SF(I)=MU(I)*XL2/G(MTYPE)/AREA(MTYPE)
            TF(I)=BF(I)/XL2*IY2
         END IF
         AXB(I)=XL(I)*DSQRT(RHO(MTYPE)/E(MTYPE))
         AXKF(I)=E(MTYPE)*AREA(MTYPE)/XL(I)
      END DO
      RETURN
      END

C*****************************************************************
C                                                                *
C                    SUBROUTINE POLES                            *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|       (V. 4.1-45)   *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Compute the poles.                               *
C*****************************************************************
      SUBROUTINE POLES (RHO,AREA,E,MATP,L,BF,SF,TF,POL,IARRY,IEQ,
     1                  NPMAX)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      REAL*4 AARRAY
      REAL*8 MU,L,LA
      CHARACTER*40 MES
      EXTERNAL DETU2
      PARAMETER (TOL=0.05D0)

      COMMON AARRAY(1)
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      COMMON /DETU2/  BFI,SFI,TFI
```

```
      DIMENSION RHO(1),AREA(1),E(1),MATP(1),L(1),BF(1),SF(1),TF(1)
     1 ,POL(2,1),IARRY(2,1),IEQ(1),LA(20)

      DATA LA /22.3732854020597476, 61.6728228035893444,
     1 120.903393090542410, 199.8594481269898538, 298.5555352981764866,
     2 417.3584331758640B9, 555.1652476302116380, 713.0789180565649956,
     3 890.731797285636347, 1088.123885316612428, 1305.255182149771741,
     4 1542.12568778510775, 1798.735402222622438, 2075.0843254623158011,
     5 2371.17245750418881, 2686.999798348231300, 3022.5663479944690497,
     6 3377.87210644287834, 3752.917073690382560, 4147.7012497462320523/

      IEQ(1)=1
      DO I=2,NCEL                ee if there are equivalent elements
         IEQ(I)=I                n this model.
         DO J=I-1,1,-1
            IF ((L(I).EQ.L(J)).AND.(MATP(I).EQ.MATP(J))) IEQ(I)=J
         END DO
      END DO

      NPOL=0
      WRITE (IOUT,100)

      DO I=1,NCEL

         IF (IEQ(I).NE.I) THEN
            DO II=1,NPOL
               IF (IARRY(1,ii).EQ.IEQ(I)) IARRY(2,II)=IARRY(2,II)+1
            END DO
            GOTO 20
         END IF

         BFI=BF(I)
         SFI=SF(I)
         TFI=TF(I)
         XL2=L(I)*L(I)
         IEL=MATP(I)
         MU=RHO(IEL)*AREA(IEL)
         BFM=DSQRT(1.D0/BF(I))
         AF =DSQRT(E(IEL)/RHO(IEL)/XL2)*3.14159265358979323846
         F=BFM*LA(1) + 1.D0
         N=1
         OLDF=0.D0
         DO WHILE (OLDF.LT.F2)
            IF (SF(I).GT.0.D0) THEN
               PF=F
               R1=DETU2(F)
               DF=(F - OLDF)/10.D0
               OF=F - DF
               DO WHILE (OF.GT.OLDF)
                  R2=DETU2(OF)
                  IF (DSIGN(1.D0,R1)*DSIGN(1.D0,R2).LT.0.D0) THEN
                     A=OF
                     B=PF
                     CALL FALSI (A,B,DETU2,RF,TOL,ITER,MES)
                     NPOL=NPOL + 1
                                IF (NPMAX.LT.NPOL) GOTO 900
                     IARRY(1,NPOL)=I
                     IARRY(2,NPOL)=1
                     POL(1,NPOL)=RF
                     POL(2,NPOL)=F
                  END IF
                  PF=OF
                  R1=R2
                  OF=OF - DF
               END DO
            ELSE
               NPOL=NPOL+1
                        IF (NPMAX.LT.NPOL) GOTO 900
               IARRY(1,NPOL)=I
               IARRY(2,NPOL)=1
               POL(1,NPOL)=F - 1.D0
               POL(2,NPOL)=F
            END IF
            OLDF=F
            N=N + 1
            F=BFM*LA(N) + 1.D0
         END DO
         N=1
         F=AF
         DO WHILE (F.LT.F2)
            NPOL=NPOL + 1
                        IF (NPMAX.LT.NPOL) GOTO 900
            IARRY(1,NPOL)=I
            IARRY(2,NPOL)=1
            POL(1,NPOL)=F
            POL(2,NPOL)=F
            N=N + 1
            F=F + AF
         END DO
      20 END DO

         DO I=1,NPOL
            DO J=I+1,NPOL
               IF (POL(1,I).GT.POL(1,J)) THEN
                  DO K=1,2
                     F=POL(K,I)
```

```
                  POL(K,I)=POL(K,J)
                  POL(K,J)=F
                  IAUX=IARRY(K,I)
                  IARRY(K,I)=IARRY(K,J)
                  IARRY(K,J)=IAUX
               END DO
            END IF
         END DO
      WRITE (IOUT,110) I,POL(2,I),POL(1,I),IARRY(2,I)
      END DO
      WRITE (IOUT,120) TOL

      N7 = N6 + NPOL
      CALL ARRPOL (AARRAY(N6),AARRAY(N7),IARRY,POL)

      RETURN
  100 FORMAT(1H1,25X,'P O L E S (Infinity wraps)'/
     *29x,'(radians/second)'//
     128X,1H*,16X,1H+,/,10X,4HPOLE,9X,5HUPPER,13X,4HPOLE,8X,
     212HMULTIPLICITY /,11X,3HNo.,9X,5HBOUND,/)
  110 FORMAT(11X,I3,6X,F12.5,5X,F12.5,5X,I3)
  120 FORMAT(/5X,'+ Natural frequencies of the fixed-fixed continuum'
     1/,7x,'elements, computed with an accuracy of ',E8.3,1H.,/,
     25X,'* Based on the natural frequencies of similar elements',/,7X,
     3'without shear deformation or rotatory inertia (Natural ',/,7x,
     4'frequency plus one)')
  200 FORMAT (1X,'ERROR - Not enough space while computing pole:',I3)
  900 WRITE (IOUT,200) NPOL
      STOP 'STOP - Not enough space while computing poles'
      END
C
C*****************************************************************
C                                                               *
C                      SUBROUTINE ARRPOL                        *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
C Date:       May 1985                  |                       *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C               Arrange memory locations for poles              *
C*****************************************************************
      SUBROUTINE ARRPOL (MULT,PPOL,IARRY,POL)
      REAL*8 POL,PPOL
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      DIMENSION POL(2,1),PPOL(1),MULT(1),IARRY(2,1)
      DO I=1,NPOL
         MULT(I)=IARRY(2,I)
         PPOL(I)=POL(1,I)
      END DO
      RETURN
      END

C*****************************************************************
C                                                               *
C                      FUNCTION DETU2                           *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
C Date:       May 1985                  |                       *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This function was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C FUNCTION PURPOSE:                                             *
C               Compute the determinant of submatrix U2.        *
C*****************************************************************
      REAL*8 FUNCTION DETU2(W)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L1,L2
      REAL*16 Q84,QS,QT,QC1,QC2,QC3,QD
      COMMON /DETU2/ BF,SF,TF

      W2=W**W
      B4=BF**W2
      S=SF**W2
      T=TF**W2
      SPT=(S + T)/2.D0
      SMT=S - T
      R1=DSQRT(B4 + SMT*SMT/4.D0)
      L1=DSQRT(R1 - SPT)
      L2=DSQRT(R1 + SPT)
      SHL1=DSINH(L1)/L1
      SIL2=DSIN(L2)/L2
      C1=L2*L2*SHL1 + L1*L1*SIL2
```

```
      C2=DCOSH(L1) - DCOS(L2)
      C3=SHL1 - SIL2
      IF (B4.LT.65536.D0) THEN
         DETU2=B4*C2*C2 + (S*C1 - (B4 + S*S)*C3) * (C1 - T*C3)
      ELSE
         QB4=QEXTD(B4)
         QS=QEXTD(S)
         QT=QEXTD(T)
         QC1=QEXTD(C1)
         QC2=QEXTD(C2)
         QC3=QEXTD(C3)
         QD=QB4*QC2*QC2 + (QS*QC1 - (QB4 + QS*QS)*QC3) * (QC1 - QT*QC3)
         IF (QABS(QD).LT.1.Q35) THEN
            DETU2=D8LEQ(QD)
         ELSE
            DETU2=D8LEQ(QSIGN(1Q35,QD))
         END IF
      END IF
      RETURN
      END

C*************************************************************************
C                                                                        *
C                          SUBROUTINE BFMTHD                             *
C                                                                        *
C------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN          *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)          *
C             V.P.I & S.U.              |                                *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                  *
C Date:       May 1985                  |                                *
C------------------------------------------------------------------------*
C                                                                        *
C  DISCLAIMER:  This subroutine was developed for research purposes     *
C               only.  It has not been optimized for general use.       *
C                                                                        *
C SUBROUTINE PURPOSE:                                                    *
C               Extract structure eigenvalues using an Incremental-     *
C               Falsi-Bisection technique and assuming that the struc-  *
C               ture eigenvalues are at least 1 rad/sec apart.          *
C*************************************************************************
      SUBROUTINE BFMTHD
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*40 MES
      REAL*4 TIMEON,TPOL,TIM,TIMEIG,TSTART
      EXTERNAL FCN
      COMMON /DEBUG/ ID1,ID2,ID3,ID4,ID5,ID6,ID7
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,FMIN,FMAX,STEP,IWW,WST
      COMMON /SOLTIM/ TIMEON,TPOL,TIM(10),NEIG,TIMEIG(60)

      NEIG=0
      REWIND (IEIGV)

      DF=STEP/3.D0
      F1=FMIN
   10 DF1=FCN(F1)
      IF (IFPOL) THEN
         F1=F1+1.D0
         IFPOL=0
         GOTO 10
      ELSE
         IF (DF1.EQ.0.D0) THEN
            CALL DETERM (F1,B,1)
            CALL REPORT (F1,ITER,MES)
            IF (ID1) WRITE (12,1000)
            TIMEIG(NEIG)=CPUSEC(TIMEON)
            F1=F1+1.D0
            GOTO 10
         END IF
      END IF

      F2=F1+DF

      DO WHILE (F2.LT.FMAX)
   20    DF2=FCN(F2)
         IF (IFPOL) THEN
            F2=F2+1.D0
            IFPOL=0
            GOTO 20
         ELSE
            IF (DF2.EQ.0.D0) THEN
               CALL DETERM (F2,B,1)
               CALL REPORT (F2,ITER,MES)
               IF (ID1) WRITE (12,1000)
               TIMEIG(NEIG)=CPUSEC(TIMEON)
               F1=F2+1.D0
               DF1=FCN(F1)
               F2=F1+DF
               GOTO 20
            END IF
         END IF

         IF (DSIGN(1.D0,DF1)*DSIGN(1.D0,DF2).LT.0.D0) THEN
            A=F1
            B=F2
```

```
            CALL FALSI (A,B,FCN,EIG,1.E-16,ITER,MES)
            IF (IFPOL) THEN
               IFPOL=0
               GOTO 400
            END IF
            CALL DETERM (EIG,B,1)
            CALL REPORT (EIG,ITER,MES)
            IF (ID1) WRITE (12,1000)
            TIMEIG(NEIG)=CPUSEC(TIMEON)
            F2=EIG+1.0000
            DF2=FCN(F2)
         END IF

  400    F1=F2
         DF1=DF2
         F2=F1+DF
  900 END DO

      RETURN
 1000 FORMAT (1H1)
      END
C
C*****************************************************************
C                                                                *
C                      SUBROUTINE RPRT2                          *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|       (V. 4.1-45)   *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C              Report eigenvalues obtained from IMSL routine.    *
C*****************************************************************
      SUBROUTINE RPRT2 (ALFA,BETA,WK,N)
      COMPLEX*16 AUXIMG,ALFA(N)
      REAL*8 AUX,BETA(N),WK(N)
      COMMON /SOLTIM/ TIMEON,TPOL,TIM(10),IROOT,TIMEIG(60)
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV

      WRITE (IOUT,999) WK(1)
      DO I=1,N
         IF (BETA(I).EQ.0.) THEN
            WK(I)=1.D36
         ELSE
            WK(I)=DREAL(ALFA(I))/BETA(I)
         END IF
      END DO

      DO I=1,N-1
         DO J=I+1,N
            IF (WK(I).GT.WK(J)) THEN
               AUX=WK(I)
               WK(I)=WK(J)
               WK(J)=AUX
                AUXIMG=ALFA(I)
                ALFA(I)=ALFA(J)
                ALFA(J)=AUXIMG
               AUX=BETA(I)
               BETA(I)=BETA(J)
               BETA(J)=AUX
            END IF
         END DO
      END DO

      DO I=1,N
         IROOT=IROOT+1
         IF (WK(I).EQ.1.D36) THEN       ! It's an infinite eigen.
            WRITE (IOUT,1010) IROOT
         ELSE
            IF (DIMAG(ALFA(I)).EQ.0.D0) THEN    ! a real eigen.
               IF (WK(I).LT.0.D0) THEN
                  WRITE (IOUT,1000) IROOT,WK(I),-DSQRT(DABS(WK(I)))
               ELSE
                  WRITE (IOUT,1000) IROOT,WK(I),DSQRT(WK(I))
               END IF
            ELSE                                ! an imag. eigen.
               AUX=DIMAG(ALFA(I))/BETA(I)
               WRITE (IOUT,1020) IROOT,
     1                           DSIGN(1.D0,WK(I))*DSQRT(DABS(WK(I))),
     2                           DSIGN(1.D0,AUX)*DSQRT(DABS(AUX))
            END IF
         END IF
      END DO
      RETURN
  999 FORMAT (2X,'Returned with a performance index =',F10.2,//,6X,
     1'Eig.',10X,'Lambda ',14X,'Eigenvalue',/,6X,3HNo.)
 1000 FORMAT (6X,I3,8X,E15.8,8X,F12.5)
 1020 FORMAT (6X,I3,20X,F12.5,' + (',F12.5,')i')
 1010 FORMAT (6X,I3,12x,'Infinite',13X,'Infinite')
```

```
      END
C
C*****************************************************************
C                                                                *
C                      SUBROUTINE REPORT                         *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)  *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Report eigenvalues from non-IMSL extraction routines. *
C*****************************************************************
      SUBROUTINE REPORT (EIG,ITER,MES)
      REAL*8 EIG,F1,F2,STP,WSTP
      CHARACTER*(*) MES
      COMMON /SOLTIM/ TIMEON,TPOL,TIM(10),IROOT,TIMEIG(60)
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV

      IROOT=IROOT+1

      IF (ITER.GE.0) THEN
         IF (IWTWL) THEN
            IF (JMOD((IROOT-1),50).LT.1) WRITE (IOUT,2000)
            CALL WITWIL (EIG+WSTP,NSTEIG)
            WRITE (IOUT,2020) NSTEIG,IROOT,EIG,MES
         ELSE
            IF (JMOD((IROOT-1),50).LT.1) WRITE (IOUT,2010)
            WRITE (IOUT,2030) IROOT,EIG,MES
         END IF
      ELSE
         WRITE (IOUT,1000) IROOT,EIG,DSQRT(EIG)
      END IF

      RETURN
 1000 FORMAT (6X,I3,8X,E15.8,8X,F12.5)
 2000 FORMAT (1H1,41X'BFMTHD RESULTS'//
     1' Wittrick Computed',7x,'Eigenvalue',31x,'Message',/
     2' Williams   Eig.  ',7x,'(rads/sec)',/
     3' Counter    No.',/)
 2010 FORMAT (1H1,41X'BFMTHD RESULTS'//
     1'          Computed',7x,'Eigenvalue',31x,'Message',/
     2'            Eig.  ',7x,'(rads/sec)',/
     3'            No.',/)
 2020 FORMAT (1X,5X,I2,4X,I2,4X,F21.14,1X,A50)
 2030 FORMAT (12X,I2,4X,F21.14,1X,A50)
      END
C
C*****************************************************************
C                                                                *
C                      SUBROUTINE WITWIL                         *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)  *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Compute the Wittrick-Williams counter for nonlinear *
C               eigenvalue extractions.                          *
C*****************************************************************
      SUBROUTINE WITWIL (W,NSTEIG)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 A
      COMMON A(1)
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      COMMON /SOL/ NUMNP,NEQ,NWK,NUEST,MIDEST,MAXEST,MK
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      COMMON /CONPTS/ N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1               N110,N111,N112,N113,N114

      CALL CHKPOL (W,A(N6),A(N7),NPOL,NTPOL)
      CALL CLEAR  (A(N8),NWK)
      CALL CASSEM (W,A(N8),A(N101),A(N102),A(N103),A(N104),A(N105),
     1             A(N113))

      IF (IFPOL) THEN
         NSTEIG=-1
         RETURN
      END IF
```

```
      CALL COLSL (A(N8),A(N2),NEQ)
      CALL CNTSGN (A(N8),A(N2),NEQ,NSGN)
      NSTEIG = NTPOL + NSGN
      RETURN
      END

C*****************************************************************
C                                                                *
C                      SUBROUTINE COLSL                          *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|        (V. 4.1-45)  *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Upper-triangularize, without any pivotting, a mtrix *
C               stored in skyline mode.  Adapted from COLSOL     *
C*****************************************************************
      SUBROUTINE COLSL(A,MAXA,NN)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),MAXA(1)
C
   40 DO 140 N=1,NN
         KN=MAXA(N)
         KL=KN+1
         KU=MAXA(N+1) -1
         KH=KU-KL
         IF(KH)110,90,50
   50    K=N-KH
         IC=0
         KLT=KU
         DO 80 J=1,KH
            IC=IC + 1
            KLT=KLT -1
            KI=MAXA(K)
            ND=MAXA(K+1) - KI - 1
            IF(ND)80,80, 60
   60       KK=MIN0(IC,ND)
            C=0.
            DO 70 L=1,KK
   70          C=C+A(KI+L)*A(KLT+L)
               A(KLT)=A(KLT)-C
   80    K=K+1
   90    K=N
         B=0.
         DO 100 KK=KL,KU
            K=K - 1
            KI=MAXA(K)
            C=A(KK)/A(KI)
            B=B +C*A(KK)
  100    A(KK)=C
         A(KN)=A(KN) -B
  110    CONTINUE
  140 CONTINUE
      RETURN
      END

C*****************************************************************
C                                                                *
C                      SUBROUTINE CNTSGN                         *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|        (V. 4.1-45)  *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Count the number of negative signs on the main diago- *
C               nal of a matrix stored in skyline mode.          *
C*****************************************************************
      SUBROUTINE CNTSGN (A,MAXA,NEQ,NSGN)
      REAL*8 A
      DIMENSION A(1),MAXA(1)
      NSGN=0
      DO I=1,NEQ
         IF (A(MAXA(I)).LT.0.D0) NSGN=NSGN+1
      END DO
      RETURN
      END
C
C*****************************************************************
C                                                                *
C                      FUNCTION FCN                              *
```

```
C                                                                      *
C-----------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN        *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)       *
C             V.P.I & S.U.              |                              *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                *
C Date:       May 1985                  |                              *
C-----------------------------------------------------------------------*
C                                                                      *
C  DISCLAIMER:  This function was developed for research purposes      *
C               only.  It has not been optimized for general use.      *
C                                                                      *
C FUNCTION PURPOSE:                                                    *
C               Compute the determinant of the dynamic stiffness       *
C               matrix, compressed logarithmically and modified for    *
C               poles.                                                 *
C**********************************************************************
      REAL*8 FUNCTION FCN (X)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 A
      COMMON A(1)
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      COMMON /DEBUG/ ID1,ID2,ID3,ID4,ID5,ID6,ID7
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP

      CALL DETERM (X,DET,0)
      IF (IFPOL) RETURN
      CALL CHKPOL (X,A(N6),A(N7),NPOL,NTPOL)
      IF (NTPOL) THEN
         FCN=-DET
      ELSE
         FCN=DET
      END IF
      IF (ID1) WRITE (12,100) X,FCN
      RETURN
  100 FORMAT (1X,F20.14,2X,E40.30)
      END
C
C**********************************************************************
C                                                                      *
C                        SUBROUTINE CHKPOL                             *
C                                                                      *
C-----------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN        *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)       *
C             V.P.I & S.U.              |                              *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                *
C Date:       May 1985                  |                              *
C-----------------------------------------------------------------------*
C                                                                      *
C  DISCLAIMER:  This subroutine was developed for research purposes    *
C               only.  It has not been optimized for general use.      *
C                                                                      *
C SUBROUTINE PURPOSE:                                                  *
C               Count the number of poles below a set frequency.       *
C**********************************************************************
      SUBROUTINE CHKPOL (W,MULT,POL,NPO,NTP)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION POL(1),MULT(1)

      NTP=0
      N=1
      DO WHILE (W.GT.POL(N))
         NTP=NTP + MULT(N)
         N=N+1
         IF (N.GT.NPO) RETURN
      END DO
      RETURN
      END
C
C**********************************************************************
C                                                                      *
C                        SUBROUTINE DETERM                             *
C                                                                      *
C-----------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN        *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)       *
C             V.P.I & S.U.              |                              *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                *
C Date:       May 1985                  |                              *
C-----------------------------------------------------------------------*
C                                                                      *
C  DISCLAIMER:  This subroutine was developed for research purposes    *
C               only.  It has not been optimized for general use.      *
C                                                                      *
C SUBROUTINE PURPOSE:                                                  *
C               Compute the determinant of matrix stored in skyline    *
C               mode.                                                  *
C**********************************************************************
      SUBROUTINE DETERM (W,DET,IVECT)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*4 A
      COMMON A(1)
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
```

```
      COMMON /SOL/ NUMNP,NEQ,NWK,NUEST,MIDEST,MAXEST,MK
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      COMMON /CONPTS/ N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1               N110,N111,N112,N113,N114

      IF (IVECT) GOTO 200

      CALL CLEAR (A(N8),NWK)
      CALL CASSEM (W,A(N8),A(N101),A(N102),A(N103),A(N104),A(N105),
     1             A(N113))

      IF (IFPOL) RETURN
      CALL CVTSOR (A(N8),A(N9),A(N2),NWK,NEQ)
      DET=DETER3(NEQ,A(N9),A(N10))
      RETURN

  200 CALL SOLVEC (W,NEQ,A(N9),A(N10),A(N11))
      RETURN
      END
C
C*****************************************************************************
C                                                                             *
C                          SUBROUTINE CASSEM                                  *
C                                                                             *
C-----------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN               *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)              *
C             V.P.I & S.U.              |                                     *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                       *
C Date:       May 1985                  |                                     *
C-----------------------------------------------------------------------------*
C                                                                             *
C  DISCLAIMER:  This subroutine was developed for research purposes           *
C               only.  It has not been optimized for general use.             *
C                                                                             *
C SUBROUTINE PURPOSE:                                                         *
C .             Assemble the structure dynamic stiffness matrix.              *
C*****************************************************************************
      SUBROUTINE CASSEM (W,K,SN,C,S2,C2,CS,LM)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K,KA,KB
      REAL*4 A
      COMMON A(1)
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      COMMON /CONPTS/ N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1               N110,N111,N112,N113,N114
      COMMON /SOL/ NUMNP,NEQ,NWK,NUMEST,MIDEST,MAXEST
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      DIMENSION K(1),KA(3),KB(10),S(21),SN(1),C(1),S2(1),C2(1),CS(1),
     1          LM(6,1)

      W2=W*W
      DO I=1,NCEL
         CALL AXIALK (I,W,A(N111),A(N112),KA)
         IF (IFPOL) RETURN
         CALL BENDK  (I,W2,A(N108),A(N109),A(N110),A(N107),A(N106),KB)
         IF (IFPOL) RETURN
         S( 1) =  KA(1)*C2(I) + KB(1)*S2(I)
         S( 2) =  CS(I)*(KA(1) - KB(1))
         S( 3) = -KB(2)*SN(I)
         S( 4) =  KA(2)*C2(I) + KB(3)*S2(I)
         S( 5) =  CS(I)*(KA(2) - KB(3))
         S( 6) = -KB(4)*SN(I)
         S( 7) =  KA(1)*S2(I) + KB(1)*C2(I)
         S( 8) =  KB(2)*C(I)
         S( 9) =  S(5)
         S(10) =  KB(3)*C2(I) + KA(2)*S2(I)
         S(11) =  KB(4)*C(I)
         S(12) =  KB(5)
         S(13) = -KB(6)*SN(I)
         S(14) =  KB(6)*C(I)
         S(15) =  KB(7)
         S(16) =  S(1)
         S(17) =  S(2)
         S(18) = -S(3)
         S(19) =  S(7)
         S(20) = -S(8)
         S(21) =  S(12)
         CALL ADDBAN (A(N8),A(N2),S,LM(1,I),6)
      END DO

      CALL INCFEM (NWK,W2,A(N8),A(N3),A(N4))

      RETURN
      END
C
C*****************************************************************************
C                                                                             *
C                          SUBROUTINE AXIALK                                  *
C                                                                             *
C-----------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN               *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)              *
C             V.P.I & S.U.              |                                     *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                       *
C Date:       May 1985                  |                                     *
```

```
C-------------------------------------------------------------------*
C                                                                   *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use.   *
C                                                                   *
C SUBROUTINE PURPOSE:                                               *
C               Find the element axial stiffness matrix for an exact*
C               element.                                            *
C*******************************************************************
      SUBROUTINE AXIALK (N,OMEG,AXB,AXKF,K)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K,L
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      DIMENSION AXB(1),AXKF(1),K(3)

      B=AXB(N)*OMEG
      SINB=DSIN(B)
      IF (DABS(SINB).LT.1.E-30) THEN
         IFPOL=1
         RETURN
      END IF
      FAC=AXKF(N)*B/SINB
      K(1)=DCOS(B)
      K(3)=K(1)
      K(2)=-1.D0
      DO I=1,3
         K(I)=K(I)*FAC
      END DO
      RETURN
      END

C*******************************************************************
C                                                                   *
C                        SUBROUTINE BENDK                           *
C                                                                   *
C-------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN     *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)    *
C             V.P.I & S.U.              |                           *
C             Blacksburg, Virginia      | O.S.: VMS 4.1             *
C Date:       May 1985                  |                           *
C-------------------------------------------------------------------*
C                                                                   *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use.   *
C                                                                   *
C SUBROUTINE PURPOSE:                                               *
C               Compute the element bending stiffness matrix for an *
C               exact element.                                      *
C*******************************************************************
      SUBROUTINE BENDK (N,OMEG2,SF,TF,BF,MU,XL,K)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K,MU,L1,L2
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,F1,F2,STP,IWTWL,WSTP
      DIMENSION SF(1),TF(1),BF(1),MU(1),XL(1),K(10)

      A=BF(N)/MU(N)/XL(N)/XL(N)
      B=BF(N)*OMEG2
      T=TF(N)*OMEG2
      S=SF(N)*OMEG2
      SPT=S + T
      SMT=S - T

      RAD1=DSQRT(B + SMT*SMT/4.D0)
      A1=RAD1 - SPT/2.D0
      A2=A1 + SPT
      L1=DSQRT(A1)
      L2=DSQRT(A2)
      CLI=2.D0*RAD1

      CH=DCOSH(L1)
      SH=DSINH(L1)
      CS=DCOS(L2)
      SN=DSIN(L2)

      C0=(A2*CH + A1*CS)/CLI
      C1=(A2/L1*SH + A1/L2*SN)/CLI
      C2=(CH - CS)/CLI
      C3=(SH/L1 - SN/L2)/CLI

      U111=C0 - S*C2
      U112=XL(N)*(C1 - SPT*C3)
      U121=B/XL(N)*C3
      U122=C0 - T*C2
      U211=A*C2
      U212=A*XL(N)/B*( - S*C1 + (B + S*S)*C3)
      U221=A/XL(N)*(C1 - T*C3)

      DTU2=U211*U211 - U212*U221
      IF (DABS(DTU2).LT.1.E-30) THEN
         IFPOL=1
         RETURN
      END IF
      K(1)=U221*U111 - U211*U121
      K(2)=U221*U112 - U211*U122
      K(3)= - U221
```

```
      K(4)=U211
      K(5)=U211*U112 - U212*U122
      K(6)= - U211
      K(7)=U212
      K(8)=K(1)
      K(9)=U212*U121 - U211*U111
      K(10)=K(5)

      DO I=1,10
         K(I)=K(I)/DTU2
      END DO
      RETURN
      END
C
C*****************************************************************
C                                                               *
C                      SUBROUTINE INCFEM                        *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
C Date:       May 1985                  |                       *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C              Include finite element contributions.            *
C*****************************************************************
      SUBROUTINE INCFEM (NWK,W2,K,KFE,MFE)
      REAL*8 K,KFE,MFE,W2
      DIMENSION K(1),KFE(1),MFE(1)
      DO I=1,NWK
         K(I)=K(I)+KFE(I)-W2*MFE(I)
      END DO
      RETURN
      END
C
C*****************************************************************
C                                                               *
C                      SUBROUTINE CVTSQR                        *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
C Date:       May 1985                  |                       *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C              Convert a skyline-stored matrix into a square matrix. *
C*****************************************************************
      SUBROUTINE CVTSQR (A,AA,MAXA,NWK,N)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),AA(N,N),MAXA(1)

      DO I=1,N
         KA=MAXA(I)
         KB=MAXA(I+1)-1
         KC=KB-KA
         AA(I,I)=A(KA)
         KD=I-KC
         DO J=I-1,KD,-1
            KA=KA+1
            AA(J,I)=A(KA)
         END DO
         IF (KD.GT.1) THEN
            DO J=KD-1,1,-1
               AA(J,I)=0.D0
            END DO
         END IF
      END DO
      DO I=1,N
         DO J=I+1,N
            AA(J,I)=AA(I,J)
         END DO
      END DO
      RETURN
      END
C
C*****************************************************************
C                                                               *
C                      FUNCTION DETER3                          *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
```

```
C Date:       May 1985                  |                                  *
C-------------------------------------------------------------------------*
C                                                                         *
C  DISCLAIMER:  This function was developed for research purposes        *
C               only.  It has not been optimized for general use.         *
C                                                                         *
C FUNCTION PURPOSE:                                                       *
C               This function computes the determinant of a dynamic       *
C               stiffness matrix with logarithmic compression.            *
C*************************************************************************
      REAL*8 FUNCTION DETER3 (N,K,JP)
      REAL*8 K
      REAL*16 DET,ABSDET
      DIMENSION K(N,N),JP(N-1)

      CALL UPPDET (N,IP,NPD,K,JP)

      IF (NPD) THEN
         DETER3=0.D0
         RETURN
      END IF

      DET=1.Q0
      DO I=1,N
         DET=DET*QEXTD(K(I,I))
      END DO
      IF (IP) DET=-DET

      ABSDET=QABS(DET)
      IF (ABSDET.LT.2.7182818Q0) THEN
         DETER3=D8LEQ(DET)
      ELSE
         DETER3=D8LEQ(QLOG(ABSDET))
         IF (DET.LT.0.Q0) DETER3=-DETER3
      END IF
      RETURN
      END

C*************************************************************************
C                                                                         *
C                        SUBROUTINE CVTDIA                                *
C                                                                         *
C-------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN           *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)            *
C             V.P.I & S.U.              |                                 *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                   *
C Date:       May 1985                  |                                 *
```

```
C-------------------------------------------------------------------------*
C                                                                         *
C  DISCLAIMER:  This subroutine was developed for research purposes       *
C               only.  It has not been optimized for general use.         *
C                                                                         *
C SUBROUTINE PURPOSE:                                                     *
C               Convert a skyline-stored matrix into a diagonal stor-     *
C               age scheme.                                               *
C*************************************************************************
      SUBROUTINE CVTDIA (A,AA,DD,MAXA,NWK,N)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),AA(N,N),DD(1),MAXA(1)

      DO I=1,N
         KA=MAXA(I)
         KB=MAXA(I+1)-1
         KC=KB-KA
         AA(I,I)=A(KA)
         KD=I-KC
         DO J=I-1,KD,-1
            KA=KA+1
            AA(J,I)=A(KA)
         END DO
         IF (KD.GT.1) THEN
            DO J=KD-1,1,-1
               AA(J,I)=0.D0
            END DO
         END IF
      END DO
      IA=0
      DO I=1,N
         DO J=I,N
            IA=IA+1
            DD(IA)=AA(I,J)
         END DO
      END DO
      RETURN
      END

C*************************************************************************
C                                                                         *
C                        SUBROUTINE UPPDET                                *
C                                                                         *
C-------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN           *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)            *
C             V.P.I & S.U.              |                                 *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                   *
```

```
C Date:       May 1985                                                  *
C-------------------------------------------------------------------- *
C                                                                       *
C  DISCLAIMER:  This subroutine was developed for research purposes    *
C               only.  It has not been optimized for general use.      *
C                                                                       *
C SUBROUTINE PURPOSE:                                                   *
C               Upper triangularize a matrix with full pivotting for   *
C               a determinant calculation.                             *
C***********************************************************************
      SUBROUTINE UPPDET (N,IP,NPD,A,JP)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N,N),JP(N-1)

      NPD=0
      IP=0

C     This loop upper triangularizes matrix A:
      DO 100 I=1,N-1
C       + - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
C       Determine if pivotting is necessary:                          +
        II=0
        JP(I)=0
        AUX=0.D0
        DO J=I,N
          DO K=I,N
            IF (AUX.LT.DABS(A(J,K))) THEN
              AUX=DABS(A(J,K))
              II=J
              JJ=K
            END IF
          END DO
        END DO

        IF (II.EQ.0) THEN
          NPD=1
          RETURN
        END IF

C       +       now exchange the rows or columns if pivotting         +
        IF (II.NE.I) THEN
          IP=IP+1
          DO K=I,N
            AUX=A(I,K)
            A(I,K)=A(II,K)
            A(II,K)=AUX
          END DO
        END IF

        JP(I)=JJ
        IF (JJ.NE.I) THEN
          IP=IP+1
          DO J=1,N
            AUX=A(J,I)
            A(J,I)=A(J,JJ)
            A(J,JJ)=AUX
          END DO
        END IF
C                                                                       +
C               end of pivotting section.                               +
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
C
C       entry point if no pivotting is to be performed:
   20   CONTINUE
C
C       UPPER TRIANGULARIZE THE MATRIX WITH GAUSS ELLIMINATION
        DO 30 K=I+1,N
          IF (A(K,I).EQ.0.D0) GOTO 30
          A(K,I)=-A(K,I)/A(I,I)
          DO 29 L=I+1,N
            A(K,L)=A(K,L) + A(K,I)*A(I,L)
   29   CONTINUE
   30   CONTINUE
  100 CONTINUE
      IP=JMOD(IP,2)
      RETURN
      END

C***********************************************************************
C                                                                       *
C                        SUBROUTINE SOLVEC                              *
C                                                                       *
C-----------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN         *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)        *
C             V.P.I & S.U.              |                               *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                 *
C Date:       May 1985                  |                               *
C-----------------------------------------------------------------------*
C                                                                       *
C  DISCLAIMER:  This subroutine was developed for research purposes    *
C               only.  It has not been optimized for general use.      *
C                                                                       *
C SUBROUTINE PURPOSE:                                                   *
C               Solve for the eigenvector.                             *
C***********************************************************************
      SUBROUTINE SOLVEC (EIG,N,A,JP,X)
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /TAPES/ IELMNT,ILOAD,NSTIF,IIN,IOUT,IDEB,IBV,IGEOM,IEIGV
      DIMENSION A(N,N),X(N),JP(N-1)
      X(N)=1.D0
      DO 20 I=N-1,1,-1
         AUX=0.D0
         DO 10, J=I+1,N
            AUX=AUX + A(I,J)*X(J)
   10    END DO
         IF (A(I,I).EQ.0.D0) STOP 'STOP Eigenvector not computable'
         X(I)=-AUX/A(I,I)
   20 END DO

C     Unscramble the eigenvector:
      DO 40 I=N-1,1,-1
         IF (JP(I).NE.I) THEN
            AUX=X(I)
            X(I)=X(JP(I))
            X(JP(I))=AUX
         END IF
   40 END DO

C     Normalize the eigenvector:
      AUX=0.D0
      DO I=1,N
         IF (DABS(AUX).LT.DABS(X(I))) AUX=X(I)
      END DO
      DO I=1,N
         X(I)=X(I)/AUX
      END DO
      WRITE (IEIGV) EIG,(X(I),I=1,N)
      RETURN
 1000 FORMAT ((10E12.6))
      END
C*****************************************************************
C                                                               *
C                      SUBROUTINE PLOTB                         *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
C Date:       May 1985                  |                       *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C               This subprogram plots reconstructs mode shapes for *
C               exact beam elements and plots structure eigenvectors *
C               on a Tektronix graphic screen.                  *
C                                                               *
C SPECIAL NOTE: It needs PLOT10-Library routines.               *
C                                                               *
C*****************************************************************
      SUBROUTINE PLOTB (X,Y,FELM,CELM,XV,ID,HED,SI,CO)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER FELM,CELM
      REAL*4 A,TIMEON,TPOL,TIM,TIMEIG
      REAL*8 SI,CO,KS
      CHARACTER TODAY*9,HOUR*8
      COMMON A(1)
      COMMON /SOL/    NUMNP,NEQ,NWK,NUMEST,MIDEST,MAXEST,MK
      COMMON /DIM/    N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      COMMON /DEBUG/ ID1,ID2,ID3,ID4,ID5,ID6,ID7
      COMMON /TAPES/  IELMNT,ILOAD,NSTF,IIN,IOUT,IDEB,IVB,IGEOM,IEIGV
      COMMON /CONPAR/ NFEL,NCEL,NPOL,IFPOL,FREQ1,FREQ2,STEP
      COMMON /SOLTIM/ TIMEON,TPOL,TIM(10),NEIG,TIMEIG(60)
      COMMON /CONPTS/ N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1                N110,N111,N112,N113,N114
      COMMON /PLTPAR/ MAXENT,IFENT

      DIMENSION FELM(2,1),CELM(2,1),ID(3,1),X(1),Y(1),XV(1),HED(1)
      DIMENSION SI(1),CO(1),KS(15),Z(4),ZA(2),D(6),IPRESS(4)
      DATA IPRESS /' Pre',' ss <',' Retu','rn> '/

      REWIND (IGEOM)
      REWIND (IEIGV)

      READ (IGEOM) NUMNP,XMIN,XMAX,YMIN,YMAX
    2 DX=DABS(XMAX-XMIN)
      DY=DABS(YMAX-YMIN)
      FACTR=DX                          ! Find the global scaling
      IF (DY.GT.FACTR) FACTR=DY         ! factor for plots
      FACTR=0.1*FACTR
      XMIN=XMIN - FACTR                 ! Compute the actual max.
      YMIN=YMIN - FACTR                 ! and min.
      XMAX=XMAX + FACTR
      YMAX=YMAX + FACTR
      READ (IGEOM) (X(I),Y(I),I=1,NUMNP)
      READ (IGEOM) IELTYP,NUMFE
      IF (IELTYP.NE.2) THEN
         NUMCE=NUMFE
         NUMFE=0
```

```
      GOTO 12
    END IF
    DO 10 I=1,NUMFE
      READ (IGEOM) IAUX,FELM(1,IAUX),FELM(2,IAUX)
 10 END DO
    READ (IGEOM) IELTYP,NUMCE
    IF (IELTYP.NE.3) STOP 'STOP - Did not find Cont. elements as expec
   1ed in PLOTB'
 12 DO 15 I=1,NUMCE
      READ (IGEOM) IAUX,CELM(1,IAUX),CELM(2,IAUX)
 15 END DO
    CALL INITT (120)
    CALL DATE (TODAY)


    DO IEIG=1,NEIG
      READ (IEIGV) EIG,(XV(I),I=1,NEQ)
      IF (IO3) WRITE (17,700) IEIG,EIG,((I,XV(I)),I=1,NEQ)
      EIG2=EIG*EIG
      CALL PGDRW
      CALL MOVABS (275,740)
      CALL TIME (HOUR)
      CALL AOUTST (52,HED)
      CALL NEWLIN
      CALL ANMODE
      WRITE (6,210) SNGL(EIG),TODAY,HOUR
      CALL TWINDO (300,975,50,700)
      CALL DWINDO (SNGL(XMIN),SNGL(XMAX),SNGL(YMIN),SNGL(YMAX))

C     Plot the undeformed shape:
      DO I=1,NUMFE
        CALL MOVEA (SNGL(X(FELM(1,I))),SNGL(Y(FELM(1,I))))
        CALL DASHA (SNGL(X(FELM(2,I))),SNGL(Y(FELM(2,I))),34)
      END DO
      DO I=1,NUMCE
        CALL MOVEA (SNGL(X(CELM(1,I))),SNGL(Y(CELM(1,I))))
        CALL DASHA (SNGL(X(CELM(2,I))),SNGL(Y(CELM(2,I))),34)
      END DO

C     Plot the eigenvector:
      IFENT=0

C     Plot the continuum mode shape:
      DO N=1,NUMCE
        I=CELM(1,N)
        J=CELM(2,N)
        K=0
        DO M=1,2
          IAUX=CELM(M,N)
          DO MM=1,3
            K=K+1
            NE=ID(MM,IAUX)
            IF (NE.GT.0) THEN
              D(K)=XV(NE)
            ELSE
              D(K)=0.
            END IF
          END DO
        END DO

        S =SI(N)
        C =CO(N)

        CALL QINSTVC (N,EIG,D,S,C,Z,ZA)
        CALL ADDVEC (0,X(I),Y(I),D(1),D(2))
        CALL PLOTC (N,X(I),Y(I),C,S,IEIG,EIG,EIG2,
     1              A(N106),A(N111),A(N112),Z,ZA)
      END DO

C     Plot the finite element mode shape:
      DO N=1,NUMFE
        I=FELM(1,N)
        J=FELM(2,N)
        K=0
        DO M=1,2
          IAUX=FELM(M,N)
          DO MM=1,2
            K=K+1
            NE=ID(MM,IAUX)
            IF (NE.GT.0) THEN
              D(K)=XV(NE)*FACTR
            ELSE
              D(K)=0.D0
            END IF
          END DO
        END DO
        CALL ADDVEC (0,X(I),Y(I),D(1),D(2))
        CALL ADDVEC (1,X(J),Y(J),D(3),D(4))
      END DO
      CALL EXPVEC (EIG,A(N11),A(N12),A(N13),A(N14),A(N15),
     1             SNGL(FACTR))
      CALL MOVABS (1,730)
      CALL AOUTST (16,IPRESS)
      CALL TINPUT (IAUX)
    END DO
    RETURN
```

```
C
  200 FORMAT ('Press [RETURN]',$)
  210 FORMAT ('+',19x,'Frequency: ',F11.4,' rad/sec.',1x,
     1A9,' at ',A8)
  300 FORMAT (I5,4E15.8)
  400 FORMAT ((1X,10E12.6))
  500 FORMAT (3I5)
  600 FORMAT ((10E12.6))
  700 FORMAT ('1Eigenpair ',I2,/' Eigenvalue:',F20.13,/,' Eigenvector:'
     1/15x,'dof',15X,'displacement',(/,15X,I3,10X,F20.14))
      END
C
C*****************************************************************************
C                                                                            *
C                          SUBROUTINE INSTVC                                 *
C                                                                            *
C----------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN              *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)              *
C             V.P.I & S.U.              |                                    *
C             Blacksburg, Virginia      | O.S.: VMS 4.1                      *
C Date:       May 1985                  |                                    *
C----------------------------------------------------------------------------*
C                                                                            *
C  DISCLAIMER:  This subroutine was developed for research purposes          *
C               only.  It has not been optimized for general use.            *
C                                                                            *
C SUBROUTINE PURPOSE:                                                        *
C               Find the initial state vector at the beginning end of        *
C               an exact element.                                            *
C*****************************************************************************
      SUBROUTINE INSTVC (IEL,EIG,D,SN,CO,Z,ZA)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 A
      COMMON A(1)
      COMMON /DEBUG/ ID1,ID2,ID3,ID4,ID5,ID6,ID7
      COMMON /CONPTS/ N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1               N110,N111,N112,N113,N114
      DIMENSION UB(4,4),UA(2,2),D(6),Z(4),ZA(2),C(6)
      DIMENSION U2(2,2),U2I(2,2),U1(2,2),T(2),R(2),CV1(2),CV2(2)

      CALL QTRANM (1.D0,IEL,EIG,A(N108),A(N109),A(N110),A(N107)
     1           ,A(N106),A(N111),A(N112),UB,UA,1,U2I)

      IF (ID7) WRITE (14,1000) IEL,EIG,((UB(J,I),I=1,4),J=1,4)
      IF (ID7) WRITE (14,1010) ((UA(J,I),I=1,2),J=1,2)

C     Get the bending initial state vector:
C               C(2)                                C(5)
C
C
C       C(1)--(>0-------------------------------(-0-> C(4)
C              +                                +
C             C(3)                             C(6)

      DO I=1,4,3
         C(I)  =D(I)*CO   + D(I+1)*SN  ! U
         C(I+1)=D(I+1)*CO - D(I)*SN    ! -W
         C(I+2)=D(I+2)                 ! PHI
      END DO
      IF (ID7) WRITE (14,1020) ((I,C(I),I+1,C(I+1)),I=1,5,2)

      U1(1,1)=UB(1,1)
      U1(1,2)=UB(1,2)
      U1(2,1)=UB(2,1)
      U1(2,2)=UB(2,2)
      U2(1,1)=UB(1,3)
      U2(1,2)=UB(1,4)
      U2(2,1)=UB(2,3)
      U2(2,2)=UB(2,4)

      IF (ID7) THEN
         WRITE (14,1030) 'U2',((U2(J,I),I=1,2),J=1,2)
         WRITE (14,1050) ((U2I(J,I),I=1,2),J=1,2)
         WRITE (14,1100) U2(1,1)*U2I(1,1)+U2(1,2)*U2I(2,1),
     1                   U2(1,1)*U2I(1,2)+U2(1,2)*U2I(2,2),
     2                   U2(2,1)*U2I(1,1)+U2(2,2)*U2I(2,1),
     3                   U2(2,1)*U2I(1,2)+U2(2,2)*U2I(2,2)
      END IF
C
C        | M |   | Z(3) |   |  -1 |  [ | C(5) |   | U |  | C(2) | ]
C        |   | = |      | = | U   |  .[ |      | - |   |  |      | ]
C        | V |   | Z(4) |   |  2  |  [ | C(6) |   | 1 |  | C(3) | ]
C
      CALL MATVEC (2,U1,C(2),T)
      C(5)=C(5)-T(1)
      C(6)=C(6)-T(2)
      CALL MATVEC (2,U2I,C(5),Z(3))
      Z(1)=C(2)                        ! Z(1)=-W at i-1
      Z(2)=C(3)                        ! Z(3)=PHI at i-1

C     Find the axial initial state vector:
      ZA(1)= C(1)                                ! U at i-1
      ZA(2)= (C(4) - UA(1,1)*C(1))/UA(1,2)       ! N at i-1

C     Find the residual for the computed initial state vector:
      CALL MATVEC (2,U1,Z(1),CV1)
      CALL MATVEC (2,U2,Z(3),T)
```

```
      R(1)=CV1(1)+T(1)-C(5)
      R(2)=CV1(2)+T(2)-C(6)
      RESNOM=DABS(R(1)) + DABS(R(2))
      IF (ID7) THEN
         WRITE (14,1060) ((I,Z(I)),I=1,4)
         WRITE (14,1070) ((I,ZA(I)),I=1,2)
         WRITE (14,1080) ((I,R(I)),I=1,2)
      END IF
      IF (RESNOM.LT.1.D0) RETURN
      MAXITR=10
      IF (ID7) WRITE (14,1090) MAXITR
      CV2(1)=CV1(1)-C(5)
      CV2(2)=CV1(2)-C(6)
      DO WHILE (RESNOM.GT.1.D0)
         DO 200 I=1,2
  200       T(I)=R(I)-CV2(I)
         CALL MATVEC (2,U2I,T,Z(3))
         CALL MATVEC (2,U2,Z(3),T)
         R(1)=CV1(1)+T(1)-C(5)
         R(2)=CV1(2)+T(2)-C(6)
         RESNOM=DABS(R(1)) + DABS(R(2))
         IF (ID7) WRITE (14,1060) ((I,Z(I)),I=1,4)
         IF (ID7) WRITE (14,1080) ((I,R(I)),I=1,2)
         MAXITR=MAXITR-1
         IF (MAXITR.EQ.0) RETURN
      END DO
      RETURN

 1000 FORMAT ('1Initial state vector calculation for element',I3,' at '
     1,F10.4,' rads/sec.'//,' Bending Tansfer Matrix:',/,(10X,4(5X,
     2E18.10)))
 1010 FORMAT (/,' Axial transfer matrix:'/(10X,2(5X,E18.10)))
 1020 FORMAT (/,' Local displacements:',/,(15X,' C(',I1,')=',E20.14,
     110X,'C(',I1,')=',E20.14))
 1030 FORMAT (/,1X,A2,' matrix:'/,(10X,2(5X,E20.14)))
 1040 FORMAT (/,' Determinant of U2=',E20.14)
 1050 FORMAT (/' U2 Inverse matrix:'/,(10X,2(5X,E20.14)))
 1060 FORMAT (/,' Initial bending state vector:',/,(15X,' Z(',I1,')=',
     1E20.14))
 1070 FORMAT (/,' Initial axial state vector:',/,(15X,' ZA(',I1,')=',
     1E20.14))
 1080 FORMAT (/,' The residual vector for this element is:'/,(15X,
     1'R(',I1,')=',E20.14))
 1090 FORMAT (/,1X,20('*'),'ITERATE FOR INITIAL STATE VECTOR',20('*')
     1,/,15X,'Maximum number of iterations =',I3)
 1100 FORMAT (/' (U2)*(U2 Inverse) matrix:'/,(10X,2(5X,E20.14)))
      END
C
```

```
C*****************************************************************
C                                                                *
C                      SUBROUTINE PLOTC                          *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)  *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C              Reconstruct eigenvector for a continuum beam element. *
C*****************************************************************
      SUBROUTINE PLOTC (IEL,XI,YI,C,S,IEIG,EIG,EIG2,XL,AXB,AXKF,Z,
     1                  ZA)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 A
      COMMON A(1)
      COMMON /DEBUG/ ID1,ID2,ID3,ID4,ID5,ID6,ID7
      COMMON /CONPTS/ N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1                N110,N111,N112,N113,N114
      DIMENSION XL(1),AXB(1),AXKF(1),Z(1),ZA(1)
      DIMENSION UB(4,4),UA(2,2),ZOLD(4),ZNEW(4),ZAOLD(2),ZANEW(2)

C     Prepare the constants for plotting:
      NFIELD=45
      DL=XL(IEL)/FLOAT(NFIELD)
      FACTR=1./FLOAT(NFIELD)
      AXIALB=AXB(IEL)*EIG
      AXFAC=1./AXKF(IEL)

C     Get the transfer matrix for a small field
      CALL QTRANM (FACTR,IEL,EIG,A(N108),A(N109),A(N110),A(N107)
     1            ,A(N106),A(N111),A(N112),UB,UA,0,ZOLD)
      IF (ID5) WRITE (15,1001) EIG,IEL
      IF (ID6) WRITE (16,1030) IEL,EIG,((UB(J,I),I=1,4),J=1,4)
      IF (ID6) WRITE (16,1000) EIG,IEL
      IF (ID6) WRITE (16,1040) ((UA(J,I),I=1,2),J=1,2)

      DO 7 I=1,4
    7   ZOLD(I)=Z(I)
      DO 8 I=1,2
    8   ZAOLD(I)=ZA(I)
```

```
      IF (ID6) WRITE (16,1020) (ZOLD(III),III=1,4),(ZA(III),III=1,2)

      CUML=0.

      DO I=1,NFIELD
         CUML=CUML + DL

C        find the axial deformation:
         CALL MATVEC (2,UA,ZAOLD,ZANEW)
         ZAOLD(1)= ZANEW(1)
         ZAOLD(2)= ZANEW(2)

C        find the bending transverse deformation
         CALL MATVEC (4,UB,ZOLD,ZNEW)
         DO 11 III=1,4
   11       ZOLD(III)=ZNEW(III)

         IF (ID6) WRITE (16,1020) (ZOLD(III),III=1,4),
     1                            (ZAOLD(III),III=1,2)
         W=-ZNEW(1)
         U= ZANEW(1)

C        now find the global displacement:
         DELY=( U*S - W*C)
         DELX=( U*C + W*S)
         XP=XI + CUML*C
         YP=YI + CUML*S
         IF (ID5) WRITE (15,1010) CUML,DELX,DELY,XP,YP
         CALL ADDVEC (1,XP,YP,DELX,DELY)
      END DO
      RETURN

 1000 FORMAT (//' STATE VECTOR AT FREQUENCY',F15.7,' FOR ELEMENT ',I2,/
     1,5X,'DEFLECTION',10X,'SLOPE',10X,'MOMENT',13X,'SHEAR',10X,
     2'AXIAL DEF.',9X,'AXIAL LD.',/,
     37X,'(-W)',14X,'(PHI)',11X,'(M)',14X,'(V)',14X,'(U)',13X,'(N)')
 1001 FORMAT (1H1,'RESULTS AT FREQUENCY ',F15.7,' FOR ELEMENT ',I2,/,
     15X,'CUM LENGTH',10X,'DELTA X',10X,'DELTA Y',10X,'X-COOR',
     210X,'Y-COOR')
 1002 FORMAT (1H1,'RESULTS AT FREQUENCY ',F15.7,' FOR ELEMENT ',I2,/
     15X,'ZA(1)=',E20.14,'ZA(2)=',E20.14,//
     15X,'CUM LENGTH',9X,'AXIAL DEFORMATION')
 1010 FORMAT (1X,F15.5,4X,2(2X,E15.9),4X,2(2X,E15.9))
 1020 FORMAT (1X,6(E15.9,2X))
 1030 FORMAT ('1Transfer matrices used for element',I3,' at '
     1,F10.4,' rads/sec.'//,' Bending Tansfer Matrix:',/,(10X,4(5X,
     2E18.10)))
 1040 FORMAT (/,' Axial transfer matrix:'/(10X,2(5X,E18.10)))
      END
```

```
C*************************************************************
C                                                            *
C                    SUBROUTINE MATVEC                       *
C                                                            *
C------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|        (V. 4.1-45) *
C             V.P.I & S.U.              |                     *
C             Blacksburg, Virginia      | O.S.: VMS 4.1       *
C Date:       May 1985                  |                     *
C------------------------------------------------------------*
C                                                            *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use.  *
C                                                            *
C SUBROUTINE PURPOSE:                                        *
C               Multiply a matrix times a vector.            *
C*************************************************************
      SUBROUTINE MATVEC (N,A,B,C)
      REAL*8 A(N,N),B(N),C(N)
      DO 10 I=1,N
           C(I)=0.DO
           DO 10 K=1,N
   10         C(I)=C(I) + A(I,K)*B(K)
      RETURN
      END
C*************************************************************
C                                                            *
C                    SUBROUTINE MATMPL                       *
C                                                            *
C------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|        (V. 4.1-45) *
C             V.P.I & S.U.              |                     *
C             Blacksburg, Virginia      | O.S.: VMS 4.1       *
C Date:       May 1985                  |                     *
C------------------------------------------------------------*
C                                                            *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use.  *
C                                                            *
C SUBROUTINE PURPOSE:                                        *
C               Multiply two matrices.                       *
C*************************************************************
      SUBROUTINE MATMPL (N,M,NN,A,B,C)
```

```
      REAL*8 A(N,M),B(M,NN),C(N,NN)
      DO 10 I=1,N
         DO 10 J=1,M
            C(I,J)=0.D0
            DO 10 K=1,NN
   10          C(I,J)=C(I,J) + A(I,K)*B(K,J)
      RETURN
      END

C*****************************************************************
C                                                                *
C                      SUBROUTINE QTRANM                         *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|       (V. 4.1-45)   *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Compute the transfer matrix of a continuum beam. *
C*****************************************************************
      SUBROUTINE QTRANM (F,N,W,SF,TF,BF,MU,XL,AXB,AXBF,UB,UA,IU2FLG,
     1                   U2I)
      IMPLICIT REAL*16 (A-H,O-Z)
      REAL*8 F,W,SF,TF,BF,MU,XL,AXB,AXBF,UB,UA,U2I
      REAL*16 K,L1,L2
      DIMENSION SF(1),TF(1),BF(1),MU(1),XL(1),AXB(1),AXBF(1)
      DIMENSION UA(2,2),UB(4,4),U2I(2,2)

      OMEG=QEXTD(W)
      OMEG2=OMEG*OMEG
      FL=QEXTD(XL(N)*F)
      CF2=QEXTD(F*F)
      CF =OMEG2*CF2

      B=BF(N)*CF*CF2
      A=B/FL/FL/OMEG2/QEXTD(MU(N))
      T=QEXTD(TF(N))*CF
      S=QEXTD(SF(N))*CF
      SPT=S + T
      SMT=S - T

      RAD1=QSQRT(B + SMT*SMT/4.D0)
      A1=RAD1 - SPT/2.D0
      A2=A1 + SPT
      L1=QSQRT(A1)
      L2=QSQRT(A2)
      CLI=QEXTD(2.D0)*RAD1

      CH=QCOSH(L1)
      SH=QSINH(L1)
      CS=QCOS(L2)
      SN=QSIN(L2)

      C0=(A2*CH + A1*CS)/CLI
      C1=(A2/L1*SH + A1/L2*SN)/CLI
      C2=(CH - CS)/CLI
      C3=(SH/L1 - SN/L2)/CLI

      U211 = A*C2
      U212 = A*FL/B*( - S*C1 + (B + S*S)*C3)
      U221 = A/FL*(C1 - T*C3)
      U222 = U211

      UB(1,1) = DBLEQ( C0 - S*C2                      )
      UB(1,2) = DBLEQ( FL*(C1 - SPT*C3)               )
      UB(1,3) = DBLEQ( U211                           )
      UB(1,4) = DBLEQ( U212                           )
      UB(2,1) = DBLEQ( B/FL*C3                        )
      UB(2,2) = DBLEQ( C0 - T*C2                      )
      UB(2,3) = DBLEQ( U221                           )
      UB(2,4) =        UB(1,3)
      UB(3,1) = DBLEQ( B/A*C2                         )
      UB(3,2) = DBLEQ( FL/A*( -T*C1 + (B + T*T)*C3)   )
      UB(3,3) =        UB(2,2)
      UB(3,4) =        UB(1,2)
      UB(4,1) = DBLEQ( B/A/FL*(C1 - S*C3)             )
      UB(4,2) = DBLEQ( B/A*C2                         )
      UB(4,3) = DBLEQ( B/FL*C3                        )
      UB(4,4) =        UB(1,1)

      IF (IU2FLG) THEN                ! U2 inverse is needed
         DET=U211*U211 - U212*U221
         U2I(1,1) = DBLEQ( U222/DET )
         U2I(1,2) = DBLEQ(-U212/DET )
         U2I(2,1) = DBLEQ(-U221/DET )
         U2I(2,2) = DBLEQ( U211/DET )
      END IF

C     find the axial transfer matrix:
```

```
      B = AXB(N)*OMEG*QEXTD(F)
      FAC = QEXTD(F)/AXBF(N)
      UA(1,1) =  DBLEQ( QCOS(B)       )
      UA(2,2) =         UA(1,1)
      UA(1,2) =  DBLEQ( QSIN(B)*FAC/B )
      UA(2,1) =  DBLEQ(-B*QSIN(B)/FAC )

      RETURN
      END
C
C*****************************************************************
C                                                               *
C                     SUBROUTINE ADDVEC                         *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
C Date:       May 1985                  |                       *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C               Add vector points to plot buffer.               *
C*****************************************************************
      SUBROUTINE ADDVEC (IPF,X,Y,DX,DY)
      REAL*8 X,Y,DX,DY
      REAL*4 A
      COMMON A(1)
      COMMON /PLTPAR/ MAXENT,IFENT
      COMMON /DIM/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
      DIMENSION IA(1)
      EQUIVALENCE (A(1),IA(1))

      IA(N11+IFENT)=IPF
      A(N12+IFENT)=SNGL(X)
      A(N13+IFENT)=SNGL(Y)
      A(N14+IFENT)=SNGL(DX)
      A(N15+IFENT)=SNGL(DY)
      IFENT=IFENT+1
      IF (IFENT.GE.MAXENT)
     1 STOP 'STOP-Exceeded storage during vector plotting'
      RETURN
      END
```

```
C*****************************************************************
C                                                               *
C                     SUBROUTINE EXPVEC                         *
C                                                               *
C---------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45) *
C             V.P.I & S.U.              |                       *
C             Blacksburg, Virginia      | O.S.: VMS 4.1         *
C Date:       May 1985                  |                       *
C---------------------------------------------------------------*
C                                                               *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                               *
C SUBROUTINE PURPOSE:                                           *
C               Expand plot buffer for plotting.                *
C*****************************************************************
      SUBROUTINE EXPVEC (E,IFF,X,Y,DX,DY,FACTR)
      REAL*8 E
      COMMON /DEBUG/ ID1,ID2,ID3,ID4,ID5,ID6,ID7
      COMMON /PLTPAR/ MENT,IFENT
      DIMENSION IFF(1),X(1),Y(1),DX(1),DY(1)

      IF (ID4) WRITE (13,1000) E

      FD=ABS(DX(1))           ! FD is the maximum displacement
      DO I=1,IFENT
         IF (FD.LT.ABS(DX(I))) FD=ABS(DX(I))
         IF (FD.LT.ABS(DY(I))) FD=ABS(DY(I))
      END DO
      FD=FACTR/FD             ! FD is the scale factor to use

      DO I=1,IFENT
         IF (ID4) WRITE (13,1010) IFF(I),X(I),Y(I),DX(I),DY(I)
         DX(I)=DX(I)*FD
         DY(I)=DY(I)*FD
         X(I)=X(I)+DX(I)
         Y(I)=Y(I)+DY(I)
         IF (IFF(I)) THEN
            CALL DRAWA (X(I),Y(I))
         ELSE
            CALL MOVEA (X(I),Y(I))
         END IF
      END DO
      RETURN

 1000 FORMAT (1H1,'MODE SHAPE VECTOR AT FREQ.',F15.8,
```

```
     1/,4X,'PLOT FLAG',9X,'X',13X,'Y',13X,'DX',13X,'DY')
1010 FORMAT (I8,4(E15.8))
     END
```

```
C*****************************************************************
C                                                                *
C                      SUBROUTINE PGDRW                          *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)  *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Erase the screen and prepare it for plotting an  *
C               eigenvector.                                     *
C*****************************************************************
      SUBROUTINE PGDRW
      CALL NEWPAG
      CALL MOVABS (244,1)
      CALL DRWABS (244,779)
      CALL DRWABS (1023,779)
      CALL DRWABS (1023,1)
      CALL DRWABS (244,1)
      CALL MOVABS (270,30)
      CALL DRWREL (25,0)
      CALL DRWREL (-10,5)
      CALL MOVREL (0,-10)
      CALL DRWREL (10,5)
      CALL MOVREL (0,5)
      CALL ANCHO('X')
      CALL MOVABS (270,30)
      CALL DRWREL (0,25)
      CALL DRWREL (5,-10)
      CALL MOVREL (-10,0)
      CALL DRWREL (5,10)
      CALL BAKSP
      CALL ANCHO ('Y')
      CALL MOVABS (270,30)
      CALL LINEF
      CALL BAKSP
C     CALL ANCHO ('Z')
      CALL TSEND
      RETURN
      END
```

```
C*****************************************************************
C                                                                *
C                      SUBROUTINE QINSTVC                        *
C                                                                *
C----------------------------------------------------------------*
C Programmer: Jim Jara-Almonte          | Language: VAX FORTRAN  *
C Location:   Mechanical Engineering Dept.|         (V. 4.1-45)  *
C             V.P.I & S.U.              |                        *
C             Blacksburg, Virginia      | O.S.: VMS 4.1          *
C Date:       May 1985                  |                        *
C----------------------------------------------------------------*
C                                                                *
C  DISCLAIMER:  This subroutine was developed for research purposes *
C               only.  It has not been optimized for general use. *
C                                                                *
C SUBROUTINE PURPOSE:                                            *
C               Compute the initial state vector in Quad-Precission. *
C*****************************************************************
      SUBROUTINE QINSTVC (IEL,EIG,D,SN,CO,DZ,DZA)
      IMPLICIT REAL*16 (A-H,O-Z)
      REAL*4 A
      REAL*8 EIG,D,SN,CO,DZ,DZA
      COMMON A(1)
      COMMON /DEBUG/ ID1,ID2,ID3,ID4,ID5,ID6,ID7
      COMMON /CONPTS/ N101,N102,N103,N104,N105,N106,N107,N108,N109,
     1                N110,N111,N112,N113,N114
      DIMENSION UB(4,4),UA(2,2),D(6),DZ(4),DZA(2),C(6),Z(4),ZA(2)
      DIMENSION U2(2,2),U2I(2,2),U1(2,2),T(2),R(2),CV1(2),CV2(2)

      CALL QTRANM2 (1.D0,IEL,EIG,A(N108),A(N109),A(N110),A(N107)
     1           ,A(N106),A(N111),A(N112),UB,UA,1,U2I)

      IF (ID7) WRITE (14,1000) IEL,EIG,((UB(J,I),I=1,4),J=1,4)
      IF (ID7) WRITE (14,1010) ((UA(J,I),I=1,2),J=1,2)

C     Get the bending initial state vector:
C                C(2)                                  C(5)
C                 |                                     |
C       C(1)--(>0-------------------------------------(-0--> C(4)
C              +                                     +
C             C(3)                                  C(6)

      DO I=1,4,3
        C(I)  =QEXTD(D(I)*CO   + D(I+1)*SN  ) ! U
```

```
      C(I+1)=QEXTD(D(I+1)*CO - D(I)*SN    ) ! -W
      C(I+2)=QEXTD(D(I+2)                 ) ! PHI
    END DO
    IF (ID7) WRITE (14,1020) ((I,C(I),I+1,C(I+1)),I=1,5,2)

    U1(1,1)=UB(1,1)
    U1(1,2)=UB(1,2)
    U1(2,1)=UB(2,1)
    U1(2,2)=UB(2,2)
    U2(1,1)=UB(1,3)
    U2(1,2)=UB(1,4)
    U2(2,1)=UB(2,3)
    U2(2,2)=UB(2,4)

    IF (ID7) THEN
       WRITE (14,1030) 'U2',((U2(J,I),I=1,2),J=1,2)
       WRITE (14,1050) ((U2I(J,I),I=1,2),J=1,2)
       WRITE (14,1100) U2(1,1)*U2I(1,1)+U2(1,2)*U2I(2,1),
     1                 U2(1,1)*U2I(1,2)+U2(1,2)*U2I(2,2),
     2                 U2(2,1)*U2I(1,1)+U2(2,2)*U2I(2,1),
     3                 U2(2,1)*U2I(1,2)+U2(2,2)*U2I(2,2)
    END IF
C
C          | M |   | Z(3) |   |  -1| [ | C(5) |    |    | | C(2) | ]
C          |   | = |      | = U  .[   |      - U   |    | |      |
C           V     Z(4)      2  [  C(6)      1  C(3)  ]
C
    CALL QMATVEC (2,U1,C(2),T)
    C(5)=C(5)-T(1)
    C(6)=C(6)-T(2)
    CALL QMATVEC (2,U2I,C(5),Z(3))
    Z(1)=C(2)                          ! Z(1)=-W at i-1
    Z(2)=C(3)                          ! Z(3)=PHI at i-1

C   Find the axial initial state vector:
    ZA(1)= C(1)                                  ! U at i-1
    ZA(2)= (C(4) - UA(1,1)*C(1))/UA(1,2)         ! N at i-1

C   Find the residual for the computed initial state vector:
    CALL QMATVEC (2,U1,Z(1),CV1)
    CALL QMATVEC (2,U2,Z(3),T)
    R(1)=CV1(1)+T(1)-C(5)
    R(2)=CV1(2)+T(2)-C(6)
    RESNOM=QABS(R(1)) + QABS(R(2))
    IF (ID7) THEN
       WRITE (14,1060) ((I,Z(I)),I=1,4)
       WRITE (14,1070) ((I,ZA(I)),I=1,2)
       WRITE (14,1080) ((I,R(I)),I=1,2)
    END IF
    IF (RESNOM.LT.1.D0) GOTO 900
C     MAXITR=10
C     IF (ID7) WRITE (14,1090) MAXITR
C     CV2(1)=CV1(1)-C(5)
C     CV2(2)=CV1(2)-C(6)
C     DO WHILE (RESNOM.GT.1.D0)
C        CALL QMATVEC (2,U2I,R,T)
C        DO 200 I=1,2
C 200      Z(I+2)=Z(I+2) - T(I)
C        CALL QMATVEC (2,U2,Z(3),T)
C        R(1)=CV1(1)+T(1)-C(5)
C        R(2)=CV1(2)+T(2)-C(6)
C        RESNOM=QABS(R(1)) + QABS(R(2))
C        IF (ID7) WRITE (14,1060) ((I,Z(I)),I=1,4)
C        IF (ID7) WRITE (14,1080) ((I,R(I)),I=1,2)
C        MAXITR=MAXITR-1
C        IF (MAXITR.EQ.0) GOTO 900
C     END DO
  900 DO 901 I=1,4
  901    DZ(I)=DBLEQ(Z(I))
      DO 902 I=1,2
  902    DZA(I)=DBLEQ(ZA(I))
      RETURN

 1000 FORMAT ('1Initial state vector calculation for element',I3,' at '
     1,F10.4,' rads/sec.'//,' Bending Tansfer Matrix:',/,(10X,4(5X,
     2E18.10)))
 1010 FORMAT (/,' Axial transfer matrix:'/(10X,2(5X,E18.10)))
 1020 FORMAT (/,' Local displacements:',/,(15X,' C(',I1,')=',E20.14,
     110X,'C(',I1,')=',E20.14))
 1030 FORMAT (/,1X,A2,' matrix:'/,(10X,2(5X,E20.14)))
 1040 FORMAT (/,' Determinant of U2=',E20.14)
 1050 FORMAT (/' U2 Inverse matrix:'/,(10X,2(5X,E20.14)))
 1060 FORMAT (/,' Initial bending state vector:',/,(15X,' Z(',I1,')=',
     1E20.14))
 1070 FORMAT (/,' Initial axial state vector:',/,(15X,' ZA(',I1,')=',
     1E20.14))
 1080 FORMAT (/,' The residual vector for this element is:'/,(15X,
     1'R(',I1,')=',E20.14))
 1090 FORMAT (/,1X,20('*'),'ITERATE FOR INITIAL STATE VECTOR',20('*')
     1,/,15X,'Maximum number of iterations =',I3)
 1100 FORMAT (/' (U2)*(U2 Inverse) matrix:'/,(10X,2(5X,E20.14)))
      END
C
C*****************************************************************************
C                                                                            *
C                          SUBROUTINE QTRANM2                                *
```

```
C
C                                                                      *
C----------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte              | Language: VAX FORTRAN    *
C Location:   Mechanical Engineering Dept.  |           (V. 4.1-45)    *
C             V.P.I & S.U.                  |                          *
C             Blacksburg, Virginia          | O.S.: VMS 4.1            *
C Date:       May 1985                      |                          *
C----------------------------------------------------------------------*
C                                                                      *
C  DISCLAIMER:  This subroutine was developed for research purposes    *
C               only.  It has not been optimized for general use.      *
C                                                                      *
C SUBROUTINE PURPOSE:                                                  *
C               Find the transfer matrix for a continuum beam in       *
C               Quad-Precission.                                       *
C**********************************************************************
      SUBROUTINE QTRANM2 (F,N,W,SF,TF,BF,MU,XL,AXB,AXBF,UB,UA,IU2FLG,
     1                    U2I)
      IMPLICIT REAL*16 (A-H,O-Z)
      REAL*8 F,W,SF,TF,BF,MU,XL,AXB,AXBF
      REAL*16 K,L1,L2
      DIMENSION SF(1),TF(1),BF(1),MU(1),XL(1),AXB(1),AXBF(1)
      DIMENSION UA(2,2),UB(4,4),U2I(2,2)

      OMEG=QEXTD(W)
      OMEG2=OMEG*OMEG
      FL=QEXTD(XL(N)*F)
      CF2=QEXTD(F*F)
      CF =OMEG2*CF2

      B=BF(N)*CF*CF2
      A=B/FL/FL/OMEG2/QEXTD(MU(N))
      T=QEXTD(TF(N))*CF
      S=QEXTD(SF(N))*CF
      SPT=S + T
      SMT=S - T

      RAD1=QSQRT(8 + SMT*SMT/4.D0)
      A1=RAD1 - SPT/2.D0
      A2=A1 + SPT
      L1=QSQRT(A1)
      L2=QSQRT(A2)
      CLI=QEXTD(2.D0)*RAD1

      CH=QCOSH(L1)
      SH=QSINH(L1)
      CS=QCOS(L2)
      SN=QSIN(L2)

      C0=(A2*CH + A1*CS)/CLI
      C1=(A2/L1*SH + A1/L2*SN)/CLI
      C2=(CH - CS)/CLI
      C3=(SH/L1 - SN/L2)/CLI

      U211 = A*C2
      U212 = A*FL/B*( - S*C1 + (B + S*S)*C3)
      U221 = A/FL*(C1 - T*C3)
      U222 = U211

      UB(1,1) = ( C0 - S*C2                      )
      UB(1,2) = ( FL*(C1 - SPT*C3)               )
      UB(1,3) = ( U211                           )
      UB(1,4) = ( U212                           )
      UB(2,1) = ( B/FL*C3                        )
      UB(2,2) = ( C0 - T*C2                      )
      UB(2,3) = ( U221                           )
      UB(2,4) =       UB(1,3)
      UB(3,1) = ( B/A*C2                         )
      UB(3,2) = ( FL/A*( -T*C1 + (B + T*T)*C3)   )
      UB(3,3) =       UB(2,2)
      UB(3,4) =       UB(1,2)
      UB(4,1) = ( B/A/FL*(C1 - S*C3)             )
      UB(4,2) = ( B/A*C2                         )
      UB(4,3) = ( B/FL*C3                        )
      UB(4,4) =       UB(1,1)

      IF (IU2FLG) THEN                 ! U2 inverse is needed
         DET=U211*U211 - U212*U221
         U2I(1,1) = ( U222/DET )
         U2I(1,2) = (-U212/DET )
         U2I(2,1) = (-U221/DET )
         U2I(2,2) = ( U211/DET )
      END IF

C     find the axial transfer matrix:
      B = AXB(N)*OMEG*QEXTD(F)
      FAC = QEXTD(F)/AXBF(N)
      UA(1,1) = ( QCOS(B)      )
      UA(2,2) =       UA(1,1)
      UA(1,2) = ( QSIN(B)*FAC/B )
      UA(2,1) = (-B*QSIN(B)/FAC )

      RETURN
```

```
      END
C
C*************************************************************************
C                                                                        *
C                          SUBROUTINE QMATVEC                            *
C                                                                        *
C------------------------------------------------------------------------*
C Programmer: Jim Jara-Almonte            Language: VAX FORTRAN          *
C Location:   Mechanical Engineering Dept.          (V. 4.1-45)          *
C             V.P.I & S.U.                                               *
C             Blacksburg, Virginia        O.S.: VMS 4.1                  *
C Date:       May 1985                                                   *
C------------------------------------------------------------------------*
C                                                                        *
C  DISCLAIMER:  This subroutine was developed for research purposes      *
C               only.  It has not been optimized for general use.        *
C                                                                        *
C SUBROUTINE PURPOSE:                                                    *
C               Multiply a matrix times a vector in Quad-Precission.     *
C*************************************************************************
      SUBROUTINE QMATVEC (N,A,B,C)
      REAL*16 A(N,N),B(N),C(N)
      DO 10 I=1,N
           C(I)=0.QQ
           DO 10 K=1,N
   10           C(I)=C(I) + A(I,K)*B(K)
      RETURN
      END
```

APPENDIX C

Instructions for Using and Modifying DSTAP

As DSTAP was developed for research purposes only, it is not as user-friendly as other finite element codes. Therefore, the input of model data is somewhat more cumbersome than in production codes.

Program DSTAP expects to receive model data from the FORTRAN logical unit 25. The output from program DSTAP is sent to FORTRAN logical unit 26. The format of an input model is presented in subsection A below below. An example of an input file (FORTRAN logical unit 25) may be found in Appendix D. The format of the input file assigned to logical FORTRAN unit 25 is by logical line numbers. Logical line numbers are different from physical line numbers. That is, logical line numbers should not be associated with the corresponding lines in FORTRAN unit 25. Several references will be made to Ref. [9], since the static capabilities of DSTAP are the same as STAP. Please see Ref. [9] for further details.

A. Format for an Input file:

Line 1. Problem Title -- Format (20A4).

Note	Columns	Variable	Entry
None	1-80	HED	Enter the title for the model to be entered. It is useful to make this as unique as possible.

Line 2. Model Parameters -- Format (13I5).

Note	Columns	Variable	Entry
none	1-5	NUMNP	Total number of nodes in the model
1	6-10	NUMEG	Total number of element groups
2	11-15	NLCASE	Total number of load cases
none	16-20	MODEX	Execution mode: 0 - Check data only 1 - Solve the modeled problem
3	21-25	IVB	Eigenvalue analysis flag: 0 - Static analysis 1 - Eigenvalue analysis
4	26-30	IPLOTF	Mode reconstruction flag: 0 - do not plot eigenvectors 1 - reconstruct and plot eigenvectors
5,6	31-35	ID1	Determinant flag 0 - Do not save determinant 1 - Write determinant to unit 12
none	36-40	ID2	Unused
5	41-45	ID3	Eigenvector from PLOTB flag 0 - Vector is not saved 1 - Vector is written to unit 17
5	46-50	ID4	Scaled eigenvector flag (EXPVEC) 0 - Vector is not saved 1 - Vector is written to unit 13
5,7	51-55	ID5	Initial state vector flag 0 - Vector is not saved 1 - Vector is written to unit 15
5,8	56-60	ID6	Mode shape flag (PLOTC) 0 - Vector is not saved 1 - Vector is written to unit 16
5	61-65	ID7	Reconstruction state vector flag 0 - Vector is not saved 1 - Vector is written to unit 14

Notes:

1. DSTAP has been set-up so that this entry must be 2 whenever an exact element is used. The FEM-beam element group must be group 1 and the exact-beam element group 2.
2. This entry is ignored during eigenvalue analyses. In a static problem, this entry is used in the same manner as described by Ref. [9], page 219.
3. This entry must be present for eigenvalue problems.
4. This entry must be present for the reconstruction of eigenvectors. It is ignored in static problems.
5. ID1 through ID7 are debug flags, which were used during the debugging of DSTAP. Some of the information provided by the different flags is redundant.
6. When ID1 is set on, the function g(x), presented in the Progam Description section is written to logical unit 12.
7. The computed initial state vector is written to logical unit 15. This information may used to obtain the internal reactions of the mode shape, at

the master nodes.

8. The reconstructed mode shape of the exact-beam elements is written to unit 16. This information is the typical mode shape information obtained with a standard transfer matrix method program.

Line 3. Eigenvalue Problem Parameters -- Unformatted.
This line is not used in a static analysis.

Note	Type	Variable	Entry
1	INTEGER	METHOD	Linear eigenvalue extraction routine to use: 1 - Subspace Iteration 2 - EIGENR 3 - IMSL EIGZF
2	REAL*8	FREQ1	Lower end of the frequency range of interest
2	REAL*8	FREQ2	Upper end of the frequency range of interest
3	REAL*8	STEP	Step size for Incremental search and step-off size from eigenvalues.
none	INTEGER	IWTWIL	Wittrick-Williams flag 0 - Counter off 1 - Counter on
4	REAL*8	WSTEP	Step-off frequency from a computed eigenvalue for the computation of the the Wittrick-Williams counter.

Notes:

1. This is the extraction method to be used in purely FEM models. This is ignored whenever one or more exact elements are present. The Sub-Space iteration method is expained in Ref. [9] and was implemented by Ref. [58]. The EIGENR and EIGZF methods were described in the Program Description section of this dissertation.
2. FREQ1 and FREQ2 determine the frequency range of interest.
3. One third of this value is used as the interval size in the Incremental Search method and is used as a step-off after an eigenvalue is found.
4. At an eigenvalue, it is most likely that the dynamic stiffness matrix will be nonsingular. Thus, it is necessary to step-off from a computed eigenvalue to evaluate the Wittrick-Williams counter.

Line 4. Nodal Point Data - Format (4I5, 3F10.0, I5)
This line is repeated as necessary.

Note	Columns	Variable	Entry
1	1-5	N	Node number
1	6-10	ID(1,N)	Boundary constraints - 0=free, 1=fixed X - Translation boundary code
1	11-15	ID(2,N)	Boundary constraints - 0=free, 1=fixed Y - Translation boundary code

1	16-20	ID(3,N)	Boundary constraints - 0=free, 1=fixed Rotation boundary code
1	21-30	X(N)	X-coordinate of node N
1	31-40	Y(N)	Y-coordinate of node N
1	41-50	Z(N)	Z-coordinate of node N
1	51-55	KN	Node number increment for automatic node generation

Notes:
1. See Ref. [9], page 218, for details.

Lines 5 and 5a. Load Information (Omit for Eigenvalue Problems)
This line is repeated as necessary.

See Ref. [9], page 219, for this information. This section is only used in static analysis. In this type of problem, DSTAP operates in the same manner as STAP.

Line 6. FEM-beam Element group data - Format (3I5)
This line must be present whenever an exact element is used.

Note	Columns	Variable	Entry
none	1-5	NPAR(1)	Enter the number 2
1	6-10	NPAR(2)	Enter the number FEM-beam elements. This entry may be set to zero for purely exact-beam element models.
2	11-15	NPAR(3)	Number of different set of material/cross-sectional properties. May be set to zero for purely exact-beam element models.

Notes:
1. Set this entry to zero when no FEM-beam elements are present in the model.
2. This entry determines how many Lines 7 are present, see below.

Line 7. Material/Cross-Sectional Properties for FEM-Beam Elements - Unformatted
This line is repeated as necessary.

Note	Type	Variable	Entry
1	INTEGER	N	Number of property set.
none	REAL*8	E(N)	Modulus of elasticity.
none	REAL*8	AREA(N)	Cross-sectional area.
none	REAL*8	IXX(N)	Second area moment of inertia.
none	REAL*8	RHO(N)	Mass density.
2	REAL*8	SYT(N)	Yield strength. Set to zero for this version of DSTAP.

Notes:
1. The property set number must be entered sequentially, starting at 1, for each element group. No set numbers may be skipped.
2. The version of DSTAP presented here does not use the Yield Strength of the FEM-beam elements. Please set it equal to zero (0.0D0).

Line 8. FEM-Beam Element Information - Format (5I5
This line is repeated as necessary.

Note	Columns	Variable	Entry
1	1-5	M	FEM-Beam element number.
none	6-10	II	I^{th} node of the beam element.
none	11-12	JJ	J^{th} node of the beam element.
none	16-20	MTYPE	Material property set number. See Line 7 for details.
2	21-25	KG	Node generation Increment.

Notes:
1. The element numbers must be entered sequentially, starting with element number 1. The last ocurrence of Line 8 must have M equal to NPAR(2) of Line 6.
2. Please refer to Ref. [9], page 222, for details of KG.

Line 9. Exact-Beam Element group data - Format (3I5)
Not used in a static analysis

Note	Columns	Variable	Entry
none	1-5	NPAR(1)	Enter the number 3
1	6-10	NPAR(2)	Enter the number exact-beam elements. May be set to zero for purely FEM-beam elemnt models.
2	11-15	NPAR(3)	Number of different set of material/cross-sectional properties. May be set to zero for purely FEM-beam element models.

Notes:
1. Set this entry to zero when no exact-beam elements are present in the model.
2. This entry determines how many Lines 10 are present, see below.

Line 10. Material/Cross-Sectional Properties for Exact-Beam Elements-Unformatted
Not used in a static analysis

Note	Type	Variable	Entry
1	INTEGER	N	Number of property set.
none	REAL*8	E(N)	Modulus of elasticity.
2	REAL*8	G(N)	Modulus of rigidity. Enter a negative number for G(N)=infinity.

3	REAL*8	AREA(N)	Cross-sectional area.
none	REAL*8	IXX(N)	Second area moment of inertia.
none	REAL*8	RHO(N)	Mass density.

Notes:

1. The property set number must be entered sequentially, starting at 1, for each element group. No set numbers may be skipped.
2. Whenever the modulus of rigidity is entered as a negative number, DSTAP assumes that the modulus of rigity is infinite (neglects shear deformation) and assumes that the rotatory inertia is zero.
3. This second area moment of inertia is taken about the neutral axis of bending. For example, if the plane of bending lies on this page, then the second area moment of inertia is taken about a line coming out of this page.

Line 11. Exact-Beam Element Information - Format (5I5)
Not used in a static analysis.
This line is repeated as necessary.

Note	Columns	Variable	Entry
1	1-5	M	Exact-Beam element number.
none	6-10	II	I^{th} node of the beam element.
none	11-12	JJ	J^{th} node of the beam element.
none	16-20	MTYPE	Material property set number. See Line 9 for details.
2	21-25	KG	Node generation Increment.

Notes:

1. The element numbers must be entered sequentially, starting with element number 1. The last ocurrence of Line 11 must have M equal to NPAR(2) of Line 9.
2. Please refer to Ref. [9], page 222, for details of KG.

B. Running DSTAP:

Progarm DSTAP is located on the VAX cluster at the Virginia Tech Computing Center, in subdirectory DUA3:[JARAALJA.DISS]. Thus, a global DCL symbol DSTAP may be defined as:

DSTAP :== RUN DUA3:[JARAALJA.DISS]DSTAP <RETURN>

to avoid typing in the subdirectory dessignation everytime DSTAP must be run. Then, to actually run DSTAP the command: DSTAP §RETURN¶ at command level will invoke DSTAP.

C. Modifying DSTAP:

A command file LD.COM in the subdirectory DUA3:[JARAALJA.DISS] has been written to link all the necessary modules to generate program DSTAP. The list of modules needed to generate this version of DSTAP was presented in Appendix B. In summary, the main source code files are:

DUA3:[JARAALJA.DISS]DSTAP.FOR	The main program and function CPUSEC.
DUA3:[JARAALJA.DISS]STAP3.OLB	The routines from program STAP and the beam element routines.
DUA3:[JARAALJA.DISS]NEWEIG.FOR	The routines for the solution of the nonlinear eigenvalue problem.
DUA3:[JARAALJA.DISS]PLOTB2.FOR	The routines for reconstructing and plotting the eigenvectors.
IMSL-library	This commercial library contains the routines which allow the eigenvalue extraction using subprogram EIGZF. This library may not be modified.
Tektronix PLOT10 library	This commercial library contains the subroutines necessary to drive a Tektronix-type terminal in graphics mode. This library may not be modified.

Thus, after modifying and compiling the desired set of subroutines, a command @DUA3:[JARAALJA.DISS]LD <RETURN> at command level will conbine all the modules and create a new DSTAP.EXE file which will contain the modifications.

APPENDIX D

Sample Runs using Program DSTAP

The following pages present the input and output from the two exact-element models used in the first example of the Numerical Examples section. The input for the two-exact-element model of the fixed-fixed beam is presented in Fig. D-1. The corresponding DSTAP output is presented in Fig. D-2. Some of the lines in Fig. D-2 have been edited so that they fit in the prescribed space. None of the information from the output file has been changed in the editing process.

In the same manner, the input for the four-exact-element model is presented in Fig. D-3 and the corresponding output in Fig. D-4. Again, the output file has been edited to fit on the page; however, none of the information has been changed.

```
FIXED-FIXED BEAM COMPARISON WITH  2 C.E.
   3    2    0    1    1
   2    1. 100000.  301. 1 10.
   1    1    1    1     0.000     0.000     0.000    1     0.000
   2    0    0    0    12.000     0.000     0.000    1     0.000
   3    1    1    1    24.000     0.000     0.000    1     0.000
   2    0    0
   3    2    1
   1 0.300E+08 -11.5E6 0.125E+00 6.5104166666667E-04 7.304034314207753E-4 0.000E+00
   1    1    2    1    1
   2    2    3    1    1
```

Figure D-1. DSTAP Input for the Two-Exact-Element Model.

FIXED-FIXED BEAM COMPARISON WITH 2 C.E.

CONTROL INFORMATION

NUMBER OF NODAL POINTS	(NUMNP) =	3
NUMBER OF ELEMENT GROUPS	(NUMEG) =	2
NUMBER OF LOAD CASES	(NLCASE) =	0
SOLUTION MODE EQ.0, DATA CHECK EQ.1, EXECUTION	(MODEX) =	1
VIBRATION ANALYSIS EQ.0, STATIC ANALYSIS EQ.1, EXTRACT EIGENVALUES	(IVB) =	1
EIGENVECTOR RECONSTRUCTION FOR PLOTTING EQ.0, OFF - NO RECONSTRUCTION EQ.1, ON - RECONSTRUCT MODE SHAPES AND PLOT	(IPLOTF) =	0
METHOD USED IN EIGEN EXTRACTION.	(METHOD) =	2
STARTING FREQUENCY	(FREQ1) =	0.1000000E+01
ENDING FREQUENCY	(FREQ2) =	0.1000000E+06
STEP SIZE FOR FREQUENCY SEARCH	(STEP) =	0.3010000E+03
WITTRICK-WILLIAMS CHECK (1=ON, 0=OFF)	(IWTWL) =	1
WITTRICK-WILLIAMS STEP-OFF	(WSTP) =	0.1000000E+02

NODAL POINT DATA

INPUT NODAL DATA

NODE NUMBER	BOUNDARY CONDITION CODES X	Y	Z	NODAL POINT COORDINATES X	Y	Z
1	1	1	1	0.000	0.000	0.0
2	0	0	0	12.000	0.000	0.0
3	1	1	1	24.000	0.000	0.0

GENERATED NODAL DATA

NODE NUMBER	BOUNDARY CONDITION CODES X	Y	Z	NODAL POINT COORDINATES X	Y	Z
1	1	1	1	0.000	0.000	0.0
2	0	0	0	12.000	0.000	0.0
3	1	1	1	24.000	0.000	0.0

EQUATION NUMBERS

NODE NUMBER N	DEGREE OF FREEDOM X	Y	Z
1	0	0	0
2	1	2	3
3	0	0	0

Figure D-2. DSTAP output for the Two-Exact-Element Model.

ELEMENT GROUP DATA

NUMBER OF EQUATIONS . . . . . (NEQ) = 3
NUMBER OF MATRIX ELEMENTS . . . . . (NWK) = 6
MAXIMUM HALF BANDWIDTH . . . . . (MK) = 3
MEAN HALF BANDWIDTH . . . . . (MM) = 2

ELEMENT DEFINITION

ELEMENT TYPE . . . . . .(NPAR(1)) . .= 3
EQ.3, CONTINUUM BEAM ELEMENTS
NUMBER OF ELEMENTS. . . . . .(NPAR(2)) . .= 2

MATERIAL DEFINITION

NUMBER OF DIFFERENT SETS OF MATERIAL
AND CROSS-SECTIONAL CONSTANTS . . . .(NPAR(3)) . . = 1

SET No.	MODULUS OF ELASTICITY (E)	MODULUS OF RIGIDITY (G)	AREA (AREA)	MOMENT OF INERTIA (IXX)	DENSITY (RHO)
1	0.300000E+08	Infinite	0.125000E+00	0.651042E-03	0.730403E-03

ELEMENT INFORMATION /

ELEMENT NUMBER-N	NODE I	NODE J	MATERIAL SET NUMBER
1	1	2	1
2	2	3	1

Figure D-2. DSTAP output for the Two-Exact-Element Model (continued).

P O L E S (Infinity wraps)
(radians/second)

POLE No.	* UPPER BOUND	+ POLE	MULTIPLICITY
1	2273.45808	2272.45808	2
2	6265.11822	6264.11822	2
3	12281.17647	12280.17647	2
4	20300.75528	20299.75528	2
5	30325.33223	30324.33223	2
6	42392.16107	42391.16107	2
7	53057.63058	53057.63058	2
8	56389.22068	56388.22068	2
9	72428.53678	72427.53678	2
10	90472.76740	90471.76740	2
11	110521.91253	110520.91253	2

+ Natural frequencies of the fixed-fixed continuum elements, computed with an accuracy of .500E-01.
* Based on the natural frequencies of similar elements without shear deformation or rotatory inertia (Natural frequency plus one)

BFMTHD RESULTS

Wittrick Williams Counter	Computed Eig. No.	Eigenvalue (rads/sec)	Message
1	1	568.11452200995041	FAL - Root was found exactly F(
2	2	1566.02955636311677	FAL-The tolerance test was met
3	3	3070.04408270259415	FAL-The tolerance test was met
4	4	5074.93881958207260	FAL - Root was found exactly F(
5	5	7581.08305653635637	FAL - Root was found exactly F(
6	6	10588.45476796008461	FAL - Root was found exactly F(
7	7	14097.05516832095850	FAL - Root was found exactly F(
8	8	18106.88419377764967	FAL - Root was found exactly F(
9	9	22617.94184767444040	FAL - Root was found exactly F(
10	10	26528.81529027919487	FAL-The tolerance test was met
11	11	27630.22812977982403	FAL - Root was found exactly F(
12	12	33143.74303937236709	FAL - Root was found exactly F(
13	13	39158.48657111666944	FAL - Root was found exactly F(
14	14	45674.45873240057063	FAL - Root was found exactly F(
15	15	52691.65953725586132	FAL - Root was found exactly F(
16	16	53057.63058055839065	FAL-The tolerance test was met
17	17	60210.08919613237413	FAL - Root was found exactly F(
18	18	68229.74224404596498	FAL - Root was found exactly F(
19	19	76750.62564726539495	FAL - Root was found exactly F(
20	20	79586.44587083758597	FAL-The tolerance test was met
21	21	85772.84231593319964	FAL - Root was found exactly F(
22	22	95296.25798389694683	FAL - Root was found exactly F(

Figure D-2. DSTAP output for the Two-Exact-Element Model (continued).

SOLUTION TIME LOG IN SECONDS

FOR PROBLEM: FIXED-FIXED BEAM COMPARISON WITH 2 C.E.

The input of geometry	=	0.49
Computation of global F.E.M. matrices	=	0.23
Pole extraction of Continuum elements	=	0.05
Brute Force - Falsi extraction	=	61.86
T_O_T_A_L	=	62.63

ADDITIONAL TIMES OF INTEREST:,

Time to compute Eigenvalue number: 1.	=	1.65
Time to compute Eigenvalue number: 2.	=	1.78
Time to compute Eigenvalue number: 3.	=	2.02
Time to compute Eigenvalue number: 4.	=	2.11
Time to compute Eigenvalue number: 5.	=	2.52
Time to compute Eigenvalue number: 6.	=	2.43
Time to compute Eigenvalue number: 7.	=	2.49
Time to compute Eigenvalue number: 8.	=	2.61
Time to compute Eigenvalue number: 9.	=	2.69
Time to compute Eigenvalue number:10.	=	3.03
Time to compute Eigenvalue number:11.	=	1.57
Time to compute Eigenvalue number:12.	=	3.05
Time to compute Eigenvalue number:13.	=	3.21
Time to compute Eigenvalue number:14.	=	3.51
Time to compute Eigenvalue number:15.	=	3.38
Time to compute Eigenvalue number:16.	=	1.98
Time to compute Eigenvalue number:17.	=	3.29
Time to compute Eigenvalue number:18.	=	3.75
Time to compute Eigenvalue number:19.	=	3.75
Time to compute Eigenvalue number:20.	=	2.75
Time to compute Eigenvalue number:21.	=	2.63
Time to compute Eigenvalue number:22.	=	4.00

Figure D-2. DSTAP output for the Two-Exact-Element Model (continued).

```
FIXED-FIXED BEAM COMPARISON WITH  4 C.E.
   5    2    0    1    1
   2    1. 100000.  301. 1 10.
   1    1    1    1     0.000     0.000     0.000    1     0.000
   2    0    0    0     6.000     0.000     0.000    1     0.000
   5    1    1    1    24.000     0.000     0.000    1     0.000
   2    0    0
   3    4    1
   1 0.300E+08 -11.5E6 0.125E+00 6.5104166666667E-04 7.304034314207753E-4 0.000E+00
   1    1    2    1    1
   4    4    5    1    1
```

Figure D-3. DSTAP input for the Four-Exact-Element Model.

FIXED-FIXED BEAM COMPARISON WITH 4 C.E.

C O N T R O L I N F O R M A T I O N

NUMBER OF NODAL POINTS . . . . . (NUMNP) = 5

NUMBER OF ELEMENT GROUPS . . . . . (NUMEG) = 2

NUMBER OF LOAD CASES . . . . . (NLCASE) = 0

SOLUTION MODE . . . . . (MODEX) = 1
EQ.0, DATA CHECK
EQ.1, EXECUTION

VIBRATION ANALYSIS . . . . . (IVB) = 1
EQ.0, STATIC ANALYSIS
EQ.1, EXTRACT EIGENVALUES

EIGENVECTOR RECONSTRUCTION FOR PLOTTING . . (IPLOTF) = 0
EQ.0, OFF - NO RECONSTRUCTION
EQ.1, ON - RECONSTRUCT MODE SHAPES AND PLOT

METHOD USED IN EIGEN EXTRACTION. . . . . (METHOD) = 2

STARTING FREQUENCY . . . . . (FREQ1) = 0.1000000E+01

ENDING FREQUENCY . . . . . (FREQ2) = 0.1000000E+06

STEP SIZE FOR FREQUENCY SEARCH . . . . . (STEP) = 0.3010000E+03

WITTRICK-WILLIAMS CHECK (1=ON, 0=OFF) . . . .(IWTWL) = 1

WITTRICK-WILLIAMS STEP-OFF . . . . . (WSTP) = 0.1000000E+02

N O D A L P O I N T D A T A

INPUT NODAL DATA

NODE NUMBER	BOUNDARY CONDITION CODES X	Y	Z	NODAL POINT COORDINATES X	Y	Z
1	1	1	1	0.000	0.000	0.0
2	0	0	0	6.000	0.000	0.0
5	1	1	1	24.000	0.000	0.0

GENERATED NODAL DATA

NODE NUMBER	BOUNDARY CONDITION CODES X	Y	Z	NODAL POINT COORDINATES X	Y	Z
1	1	1	1	0.000	0.000	0.0
2	0	0	0	6.000	0.000	0.0
3	0	0	0	12.000	0.000	0.0
4	0	0	0	18.000	0.000	0.0
5	1	1	1	24.000	0.000	0.0

EQUATION NUMBERS

NODE NUMBER N	DEGREE OF FREEDOM X	Y	Z
1	0	0	0
2	1	2	3
3	4	5	6
4	7	8	9
5	0	0	0

Figure D-4. DSTAP output for the Four-Exact-Element Model.

ELEMENT GROUP DATA

NUMBER OF EQUATIONS . . . . . (NEQ) = 9

NUMBER OF MATRIX ELEMENTS . . . . . (NWK) = 36

MAXIMUM HALF BANDWIDTH . . . . . (MK) = 6

MEAN HALF BANDWIDTH . . . . . (MM) = 4

ELEMENT DEFINITION

ELEMENT TYPE . . . . . .(NPAR(1)) . .= 3
EQ.3, CONTINUUM BEAM ELEMENTS
NUMBER OF ELEMENTS. . . . . .(NPAR(2)) . .= 4

MATERIAL DEFINITION

NUMBER OF DIFFERENT SETS OF MATERIAL
AND CROSS-SECTIONAL CONSTANTS . . . .(NPAR(3)) . . = 1

SET No.	MODULUS OF ELASTICITY (E)	MODULUS OF RIGIDITY (G)	AREA (AREA)	MOMENT OF INERTIA (IXX)	DENSITY (RHO)
1	0.300000E+08	Infinite	0.125000E+00	0.651042E-03	0.730403E-03

ELEMENT INFORMATION /

ELEMENT NUMBER-N	NODE I	NODE J	MATERIAL SET NUMBER
1	1	2	1
2	2	3	1
3	3	4	1
4	4	5	1

TOTAL SYSTEM DATA

Figure D-4. DSTAP output for the Four-Exact-Element Model (continued).

P O L E S (Infinity wraps)
(radians/second)

POLE No.	UPPER BOUND *	POLE +	MULTIPLICITY
1	9090.83233	9089.83233	4
2	25057.47288	25056.47288	4
3	49121.70588	49120.70588	4
4	81200.02111	81199.02111	4
5	121298.32890	121297.32890	4

+ Natural frequencies of the fixed-fixed continuum elements, computed with an accuracy of .500E-01.
* Based on the natural frequencies of similar elements without shear deformation or rotatory inertia (Natural frequency plus one)

BRMTHD RESULTS

Wittrick Williams Counter	Computed Eig. No.	Eigenvalue (rads/sec)	Mes
1	1	568.11452200995051	FAL-The tolerance test was me
2	2	1566.02955636311660	FAL-The tolerance test was me
3	3	3070.04408270259376	FAL-The tolerance test was me
4	4	5074.93881958207805	FAL-The tolerance test was me
5	5	7581.08305653631123	FAL-The tolerance test was me
6	6	10588.45476796011121	FAL-The tolerance test was me
7	7	14097.05516832227977	FAL-The tolerance test was me
8	8	18106.88419379174047	FAL-The tolerance test was me
9	9	22617.94184761731458	FAL-The tolerance test was me
10	10	26528.81529027919532	FAL-The tolerance test was me
11	11	27630.22812963752040	FAL-The tolerance test was me
12	12	33143.74303986000541	FAL-The tolerance test was me
13	13	39158.48657828463820	FAL-The tolerance test was me
14	14	45674.45874491329778	FAL-The tolerance test was me
15	15	52691.65953973247724	FAL-The tolerance test was me
16	16	53057.63058055838974	FAL-The tolerance test was me
17	17	60210.08896276889118	FAL-The tolerance test was me
18	18	68229.74701399454716	FAL-The tolerance test was me
19	19	76750.63369342389524	FAL-The tolerance test was me
20	20	79586.44587083758597	FAL-The tolerance test was me
21	21	85772.74900103338587	FAL-The tolerance test was me
22	22	95296.09293674484616	FAL-The tolerance test was me

Figure D-4. DSTAP output for the Four-Exact-Element Model (continued).

```
                SOLUTION TIME LOG IN SECONDS

FOR PROBLEM:  FIXED-FIXED BEAM COMPARISON WITH  4 C.E.
The input of geometry . . . . . =      0.48
Computation of global F.E.M. matrices . . . . . =      0.25
Set-up for a Continuum-F.E. problem . . . . . =      0.01
Pole extraction of Continuum elements . . . . . =      0.04
Brute Force - Falsi extraction  . . . . . =    135.04
Post-processing of eigenvectors . . . . . =      0.01

                                    T_O_T_A_L =    135.83

ADDITIONAL TIMES OF INTEREST:,

Time to compute Eigenvalue number: 1. . . . . = 3.79
Time to compute Eigenvalue number: 2. . . . . = 3.88
Time to compute Eigenvalue number: 3. . . . . = 4.08
Time to compute Eigenvalue number: 4. . . . . = 4.53
Time to compute Eigenvalue number: 5. . . . . = 4.90
Time to compute Eigenvalue number: 6. . . . . = 4.98
Time to compute Eigenvalue number: 7. . . . . = 5.24
Time to compute Eigenvalue number: 8. . . . . = 5.74
Time to compute Eigenvalue number: 9. . . . . = 6.15
Time to compute Eigenvalue number:10. . . . . = 5.79
Time to compute Eigenvalue number:11. . . . . = 3.96
Time to compute Eigenvalue number:12. . . . . = 6.72
Time to compute Eigenvalue number:13. . . . . = 6.99
Time to compute Eigenvalue number:14. . . . . = 7.46
Time to compute Eigenvalue number:15. . . . . = 7.75
Time to compute Eigenvalue number:16. . . . . = 3.43
Time to compute Eigenvalue number:17. . . . . = 8.03
Time to compute Eigenvalue number:18. . . . . = 8.38
Time to compute Eigenvalue number:19. . . . . = 8.72
Time to compute Eigenvalue number:20. . . . . = 5.07
Time to compute Eigenvalue number:21. . . . . = 7.27
Time to compute Eigenvalue number:22. . . . . = 9.24
```

Figure D-4. DSTAP output for the Four-Exact-Element Model (continued).

The vita has been removed from the scanned document